DESIGN-BY-ANALOGY USING THE WORDTREE METHOD AND AN AUTOMATED WORDTREE GENERATING TOOL

A Thesis

by

EDGAR VELAZQUEZ ORIAKHI

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

May 2011

Major Subject: Mechanical Engineering

DESIGN-BY-ANALOGY USING THE WORDTREE METHOD AND AN AUTOMATED WORDTREE GENERATING TOOL

A Thesis

by

EDGAR VELAZQUEZ ORIAKHI

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

May 2011

Major Subject: Mechanical Engineering

ABSTRACT

Design-by-Analogy Using the WordTree Method and an Automated WordTree Generating Tool. (May 2011)

Edgar Velazquez Oriakhi, B.S, Prairie View A&M University

Chair of Advisory Committee: Dr. Julie Linsey

Design-by-Analogy is an approach that is widely embraced by engineers and designers seeking innovative designs. The identification of analogies for use in engineering design problems is usually a spontaneous action that is brought about by accident and not by a systematic design process applied during the idea generation stage of new product development. A Design-by-Analogy method developed to lead designers systematically to analogies that can be useful for solving design problems is the WordTree Method. The WordTree Method uses the semantic relationships between verbs, extracted from design problems, to lead engineers and designers to potentially useful analogies. The WordTree Method is a relatively new design method, and as with any new design method, there is room for improvement. In this thesis, a tool called WordTree Express (WTE) was developed to automate the generation of the database-based WordTrees used during the application of the WordTree Method. This tool (WTE) showed, from an experiment, that its implementation had a positive effect on the opinions of the engineers and designers who used it for solving a design problem. The effects found from surveying the participants suggested that the participants were more likely to apply the method in their future design problems with the WTE tool than when they applied the method without the WTE tool. Although the WTE tool did not show statistical significance ($p<0.1$) in increasing the number of analogies identified by the participants, compared to the non-automated method, it did enable the process of identifying analogies to be done faster. Tools designed to perform tasks faster and more efficiently usually tend to have a positive effect on its users. Different ontologies were studied for their value in the application to Design-by-Analogy in engineering. Recommendations for further work advancing the WordTree Method and contributions to Design-by-Analogy are presented in the future work section.

ACKNOWLEDGEMENTS

I would like to express my greatest sense of gratitude to my committee chair, Dr. Linsey for her guidance, encouragement and constant support throughout the course of this research. I would also like to thank my co-advisor, Dr. Peng for his guidance and support. Thanks to the members of my committee, Dr. McAdams and Dr. Johnson for their support.

I am grateful to all the students who participated in my study. This would not have been possible without them. I would like to thank my lab mates for their support and encouragement. I would also like to thank Texas A& M University and the Department of Mechanical Engineering for giving me the opportunity to be a part of such a great institution through the Pathways Fellowship Program.

Thanks to all my friends and colleagues and the department faculty and staff for making my time at Texas A&M University a great experience.

Finally, I would like to thank my mother and brothers for their constant encouragement and support throughout my time at Texas A&M University.

TABLE OF CONTENTS

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

CHAPTER I

INTRODUCTION: WORDTREE DESIGN METHOD

Engineers and designers are often faced with the need for designing innovative products; meeting such needs may sometimes require the application of one or more engineering design methods to stimulate ideas. Analogies can trigger breakthrough ideas in new product development (Schild et al., 2004). An example of Design-by-Analogy is the Velcro design from an analogy to burrs. Several procedures and methods exist which can be used to generate innovative ideas for product concepts based on analogies; such methods include synectics (Weaver & Prince, 1990), TRIZ (Altshuller, 1999) and biomimetics (Schild et al., 2004). Another design method based on analogy is the work of Linsey called the WordTree Design-by-Analogy Method (Linsey, 2007).

The WordTree Method systematically re-represents a design problem, assisting the designer in identifying analogies and analogous domains (Linsey, 2007). The WordTree Method is applied by the process shown in Figure 1. Key problem descriptors are identified from the design problem and used to create WordTrees that systematically re-represent the key functions to more abstract and domain specific terms resulting in analogies. Analogies and analogous domains are then identified for possible solutions to a design problem. Research of the analogies and a broader look into the identified analogous domains follows with newly created problem statements. Finally, ideas and concepts are generated.

____

This thesis follows the style of *Artificial Intelligence for Engineering Design, Analysis and Manufacturing.*

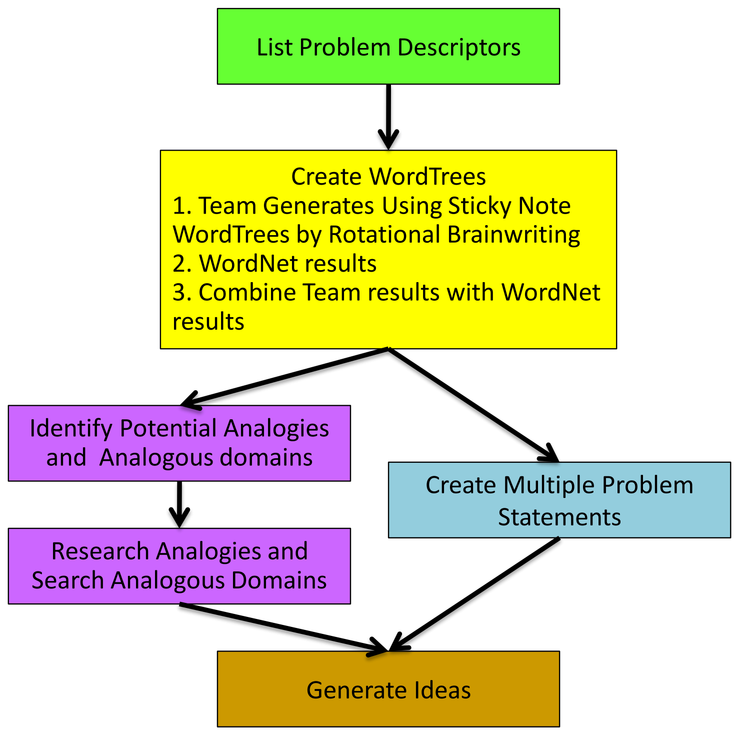

List Problem Descriptors
Create WordTrees
1. Team Generates Using Sticky Note WordTrees by Rotational Brainwriting
2. WordNet results
3. Combine Team results with WordNet results
Identify Potential Analogies and Analogous domains
Create Multiple Problem Statements
Research Analogies and Search Analogous Domains
Generate Ideas

Figure 1: WordTree Design-by-Analogy process

The WordTree Method is applied by identifying analogical relationships between a keyword function and other words/phrases in a WordTree diagram. WordTree diagrams are made using an online lexical database called WordNet (Princeton University, 2010) and from a team idea generation session where members write down words on sticky notes to make up the WordTree. From the WordTree diagrams, the designers look for analogous relationships between the keywords and the other words in the WordTree by skimming through all the words on the WordTrees. Sometimes relationships are found in distant domains from the original keywords for possible innovative solutions. The solutions from the WordTrees usually come from identified functional relationships between the keyword and other words that represent potential analogies. For example, Figure 2 shows a WordTree for the keyword "clean" that led the designers to the word "dump" in a distant part of the WordTree, and this resulted in an innovative solution for a cat litter box design where the analogy to a dump truck was used as a solution to the problem (Linsey, 2007) (Note: The diagram in Figure 2 was created for descriptive purposes only and does not accurately represent the WordTree used).

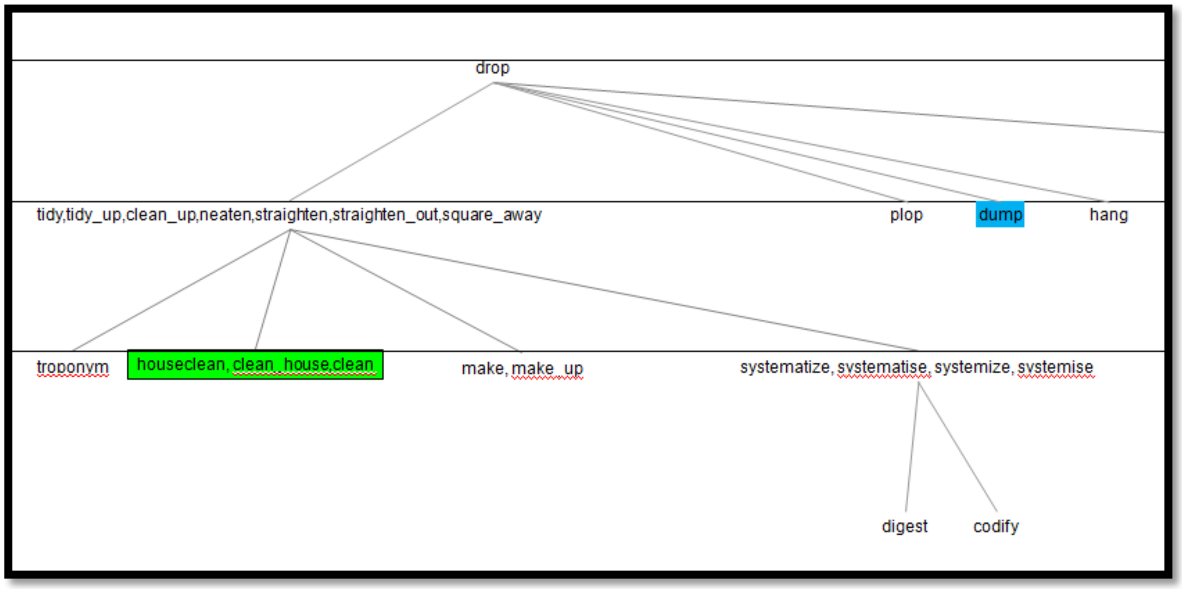

drop
tidy,tidy_up,clean_up,neaten,straighten,straighten_out,square_away
plop
dump
hang
troponym
houseclean, clean_house,clean
make, make_up
systematize, systematise, systemize, systemise
digest
codify

Figure 2: WordTree diagram for the word "clean"

This thesis is focused on the WordTree Method because it has been shown to be an effective tool for the identification of analogies and analogous domains. Although a prior study by Linsey et al., 2008, has shown positive results in the methods effectiveness, it also shows that the method needs an easier way to generate its WordNet-based WordTrees as the current method is very time-consuming and tedious. In this thesis, a major objective was to improve a significant part of the WordTree generation stage (i.e. the “WordNet results” shown in the yellow block of Figure 1) by developing an automated tool for generating the WordNet-based WordTrees. The tool was developed using Microsoft Visual Basic (VB) as the programming language and is called the “WordTree Express” (WTE). The WTE program works in combination with two other programs, Grapghviz (Ellson, J et al., 2010) and Inkscape (INKSCAPE, 2010), to accomplish its goal of creating WordTrees. Another objective of this thesis was to test the effectiveness of the WTE tool by performing a controlled study and comparing the results with the Linsey et al., 2008 study.

CHAPTER II

BACKGROUND

This chapter presents relevant work in the application of ontologies in engineering information organization and management as well as retrieval tools used in creative design analogies. Ontologies are very important for applying Design-by-Analogy because they represent a means to relate one concept to another.

What Are Ontologies?

Ontologies are an emerging means of knowledge representation to improve information organization and management, and they are becoming more prevalent in the domain of engineering design (Cross & Bathija, 2009). Artificial intelligence (AI) has borrowed the word ontology from philosophy, where it is defined as a systematic account of existence. According to Gruber, an ontology is an explicit specification of a conceptualization (Gruber, 1995).

Some ontologies can be applied to Design-by-Analogy by using the libraries of information they produce as a domain space for potential analogy search. Some important questions to ask are: Which ontologies are useful for Design-by-Analogy? How do we develop the right ontologies? These questions will be addressed with further investigation. According to Cross and Bathija, the task of creating new ontologies manually is not only tedious and cumbersome but also time consuming and expensive (Cross & Bathija, 2009). A known solution to the problem of creating new ontologies is reusing existing ones. The next section discusses an approach for reusing existing ontologies.

Approach for reusing domain-specific ontologies

One approach to reducing the cost of creating ontologies is to reuse an existing ontology mainly by extracting smaller application ontologies from larger, more general purpose ontologies (Cross & Bathija, 2009). An automated adaptation process was developed by Cross and Bathija that uses the architecture shown in Figure 3. The approach works by taking smaller ontologies from larger, more general ontologies and building upon them. The major algorithms developed for this adaptation process are those for bottom-up pruning and for matching a domain concept tree to an ontology concept in the extending phase (Cross & Bathija, 2009). The pruning algorithm incorporates techniques used for analogy evaluation because the objective is to prune concepts from the original domain that are not relevant to the new domain. The ontology obtained from the pruning phase represents the starting point for the extending phase. Concepts and the taxonomic relations relevant to this domain are added to the pruned ontology using the domain training corpus and the integration of several software resources.

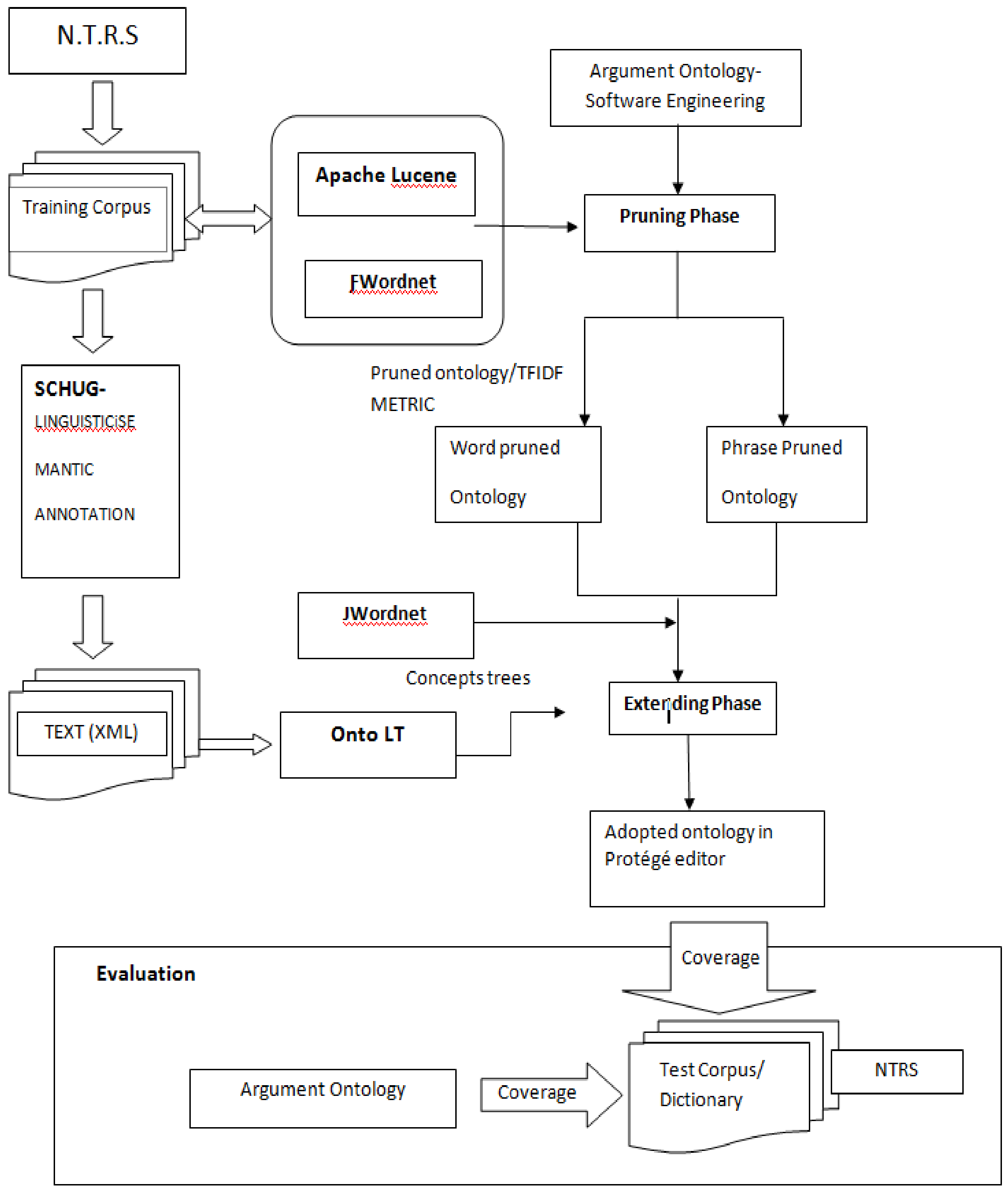


Figure 3: Ontology adaptation architecture

This approach was assessed experimentally by automatically adapting a design rationale ontology for the software engineering domain to a new one for the related domain of engineering design; the results produced an ontology that was comparable in quality to previous attempts to automate ontology creation (Cross & Bathija, 2009).

Which Ontologies Are Useful for Design-by-Analogy?

To answer the question of which ontologies are useful in Design-by-Analogy for engineering, it was necessary to study a wide range of ontologies. Ontologies for applications in engineering began with an overview of existing engineering ontologies. The following potentially useful ontologies were selected for discussion.

The PHYSSYS ontology

The PHYSSYS ontology is a formal ontology based on system dynamics theory as practiced in engineering modeling, simulation and design (Borst, 1997). The PHYSSYS ontology forms the basis for the open library for models of mechatronics components (OLMECO). Figure 4 shows the inclusion lattice of the PHYSSYS ontology.

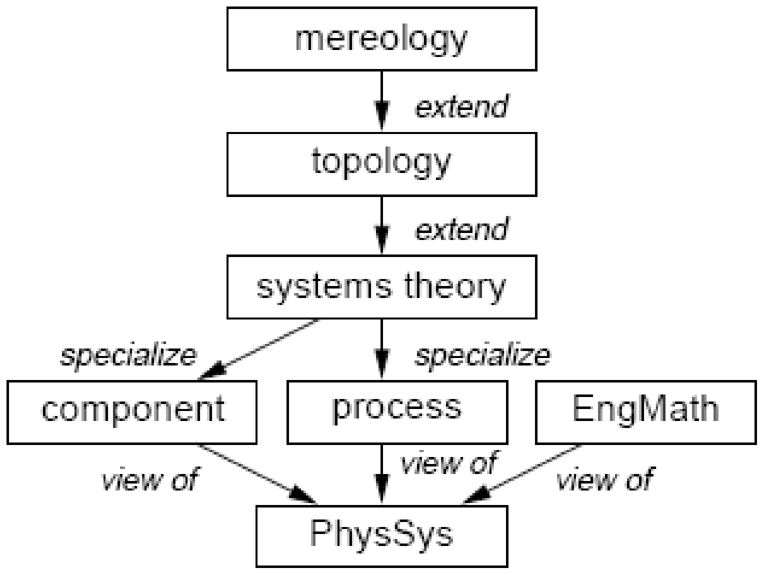


Figure 4: Inclusion lattice of the PysSys ontology

This ontology is important in answering the question of which ontologies are useful in Design-by-Analogy because it is made up of several engineering ontologies which highlight different viewpoints to consider. The PHYSSYS ontology consists of three engineering ontologies formalizing different viewpoints on physical devices: Mereological, Topological and Ontology of Systems Theory. There are three other ontologies also part of the PHYSSYS ontology: Component, Physical Process and Mathematical. All these ontologies are described as follows:

1. Mereological Ontology: Mereology means ‘science of parts’ and it defines the *part-of* relationships. Some examples of part-whole relationship are shown in Table 1.

Table 1: Examples of part-whole relationships

Whole	**Parts**
Body	organs
organism	cells
Device	components
House	roof, walls
Book	chapters

2. Topological Ontology: This ontology is based on the theory of the *is-connected* relation in general. Clarke's mereo-topology theory integrates mereological and topological concepts and relations into one (Borst, 1997).
3. Ontology of Systems Theory: This ontology defines the standard-theoretic notions such as system, subsystem, system boundary, environment, open/closed, etc.
4. Component Ontology: This ontology defines the structural view on physical systems engineers have, i.e. components that can have subcomponents and terminals.
5. Physical Process Ontology: This ontology specifies the behavioral view on physical systems Table 2 shows examples of physical domains.

Table 2: Some examples of physical domains

domain	**Stuff**	**flow**	**effort**
electrical	charge	current	voltage
mechanical	location	velocity	force
hydraulic	volume	volume flow	pressure

6. Mathematical Ontology: An example is the EngMath ontology (Gruber 1994). The EngMath ontology includes conceptual foundations for scalar, vector and tensor quantities, physical dimensions, units of measure, functions of quantities, and dimensionless quantities.

The PHYSYS ontology is broad and has many components that could make up a potentially rich design space. For Design-by-Analogy, the goal is to gain ideas by looking beyond existing products for useful analogies; to do that, there needs to be a driving force that connects a designer from one design to another useful design or idea that can be in the same domain or in a different domain. As described in Chapter I for the WordTree, this connection is based on a design's function. For the proper use of a design's function to make the connection, verbs abstracted from the design function are used as the individual connecting units from one design to an analogous design or idea. Careful review of all the ontologies within PHYSSYS suggests that it would make a large design space for engineering-specific constituents using the OLMECO library. PHYSSYS could be useful for identifying analogies between engineering products, but would leave out the very important analogies in nature. There is no known library developed with the PHYSSYS ontology that includes both engineering components and components from nature; this would have been an ideal search space for finding useful design analogies.

The YMIR ontology

The YMIR ontology specifies a taxonomy of concepts for engineering design which define the semantics of design knowledge in multiple engineering domains such as electrical engineering, mechanical engineering, and civil Engineering (Alberts & Dikker, 1992). YMIR represents two types of knowledge:

1. Synthesis knowledge: This knowledge is based on technical principles. Engineering design can be regarded as the problem of finding a configuration of physical elements in a single artifact that can perform a single function. The physical elements have particular geometrical and material properties called form that displays a certain behavior dependent of the form. The function is the required part of the combined behavior of a combination of elements (Alberts & Dikker, 1992).
2. Evaluation knowledge: the official design standards or codes that a design product has to adhere to. For instance, in the case of a bridge design, we might explicitly specify the technical function of a bridge in terms of the loads it has to transport to its fundaments. At the same time, however, we implicitly assume that the bridge will also meet the applicable safety standards, building and maintenance codes etc (Alberts & Dikker, 1992).

According to Alberts & Dikker, YMIR allows for the combination of the results from applying knowledge from different sources in the design process; this allows for stronger forms of integration between the different engineering domains (Alberts & Dikker, 1992). Stronger forms integration means that since codes and standards used in the different engineering domains are taken into consideration, there would be an easier access to its applicability by all the domains involved. The basis of the YMIR ontology is one that will be discussed in the future work section of this thesis. It possesses a characteristic that can be used in selecting the appropriate ontology development approach for a desired application. For example, if the goal was to develop ontology for the general domain of medicine as opposed to a specialization or sub-domain of medicine (e.g. pediatrics, dentistry, ophthalmology, veterinary etc.), information (both synthesis and evaluation knowledge) from all the sub-domains of medicine must be included in the development. The approach used in the YMIR ontology development can be used for future development of ontologies for Design-by-Analogy by incorporating information from the different engineering domains that would emphasize better forms of integration between the different domains. Although, this ontology is also engineering-specific and potentially useful, it would likely leave out the analogies found in nature which are very important.

The WordNet system and ontology

WordNet is an English language electronic dictionary accessible from the Internet. For a better understanding of the organization of the WordNet system, some key terms have been defined in Table 3.

Table 3: Key terms and their meaning

Key terms	**Meaning**
Collocation	A collocation in WordNet is a string of two or more words, connected by spaces or hyphens. Examples are: Man-eating shark, blue-collar, etc.
Domain	A topical classification to which a synset has been linked with a CATEGORY, REGION or USAGE pointer.
Group	Verb senses that are similar in meaning and have been manually grouped together.
Hypernym	The generic term used to designate a whole class of specific instances. **Y** is a hypernym of **X** if **X** is a (kind of) **Y**.
Hyponym	The specific term used to designate a member of a class. **X** is a hyponym of **Y** if **X** is a (kind of) **Y**.
Lemma	Lower case ASCII text of word as found in the WordNet database index files. Usually the **base form** of the word or collocation.
Lexical pointer	A lexical pointer indicates a relation between words in synsets (word forms).
Lexicographer file	Files containing the raw data for WordNet synsets, edited by lexicographers, that are put to the **grind** program to generate a WordNet database.
Lexicographer id	A decimal integer that, when appended onto **lemma**, uniquely identifies a sense within a lexicographer file.
Sense	A meaning of a word in WordNet. Each sense of a word is in a different **synset**.
Synset	A synonymous set; a set of words that are interchangeable in some context without changing the truth value of the preposition in which they are embedded.
Troponym	A verb expressing a specific manner elaboration of another verb. **X** is a troponym of **Y** if **to X** is **to Y** in some manner.

The WordNet system consists of lexicographer files. The lexicographer files organize nouns, verbs, adjectives and adverbs into groups of synonyms (synsets), and describe relations between synonym groups (Princeton University, 2010). Representations in WordNet are not on the level of individual words or word forms, but on the level of word meanings (lexemes) (Kamps & Marx, 2002). In other words, the meaning of an individual word (or word form) is characterized by listing other words or word forms that can be used to express it or replace it in a synonym set (synset), but the word meaning in WordNet is determined by its sets of synonyms (i.e. the synonym set it belongs to that defines the concept they describe). Meaning in WordNet is a structural notion: the meaning of a concept is determined by its position relative to the other words in the larger WordNet structure (Kamps & Marx, 2002). Each of the general WordNet groups are structured based on a hierarchical ordering with words describing more general concepts higher in the hierarchy and more specific ones lower.
The WordNet system is useful for identifying analogies because it presents a way to relate ideas from words based on their similarity in describing a concept; this attribute is powerful when searching for useful analogies based on semantics.

For the WordTree Method, only the semantic organization of verbs in WordNet is required because the method uses only the part of WordNet that identifies relationships between the descriptors (which are verbs) to other verbs as potential analogies. Figure 5 shows the relationship between a design and verbs. Designs are described by their functions while verbs are abstracted from designs' functions.

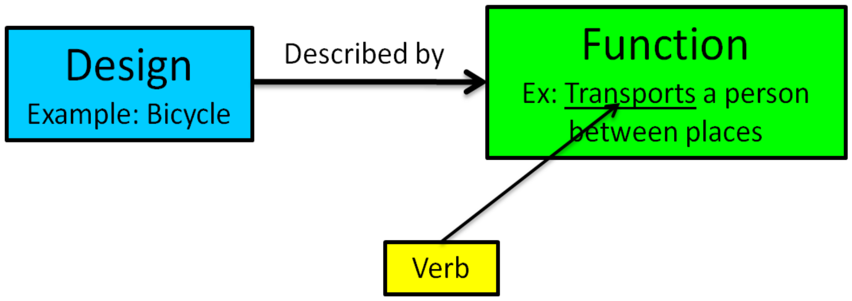


Figure 5: Relationship between a design and a verb

The WordNet system formalizes verb group relationships and distinguishes them with lexicographer file names as shown in Table 4. There are 15 verb group files (29 to 43) in the WordNet database whose members (synonym sets) are grouped based on a relationship to the concept of body, change, cognition, communication, competition, consumption, contact, creation, emotion, motion, perception, possession, social, stative, and weather. These groups or synsets and their relationships are important for retrieving analogies because they allow a designer to connect one idea from one verb to another relevant verb, from their shared concept description, for identifying potential analogies.

Table 4: WordNet database verb group files (Princeton University, 2010)

	Lexicographer file name	Description
29	verb.body	verbs of grooming, dressing and bodily care
30	verb.change	verbs of size, temperature change, intensifying, etc.
31	verb.cognition	verbs of thinking, judging, analyzing, doubting
32	verb.communication	verbs of telling, asking, ordering, singing
33	verb.competition	verbs of fighting, athletic activities
34	verb.consumption	verbs of eating and drinking
35	verb.contact	verbs of touching, hitting, tying, digging
36	verb.creation	verbs of sewing, baking, painting, performing
37	verb.emotion	verbs of feeling
38	verb.motion	verbs of walking, flying, swimming
39	verb.perception	verbs of seeing, hearing, feeling
40	verb.possession	verbs of buying, selling, owning
41	verb.social	verbs of political and social activities and events
42	verb.stative	verbs of being, having, spatial relations
43	verb.weather	verbs of raining, snowing, thawing, thundering

The WordNet Database

This section will cover the topics about WordNet that are needed to write a program code for the automation of WordNet-based WordTrees. The knowledge from a complete understanding of how the data is organized is needed because each time the database is queried by a program, the function in the program is always dependent on the position and interpretation of the queried data.

WordNet Database Organization

Information in WordNet is organized around logical groupings called synsets. Each synset consists of a list of synonymous words or collocations (e.g. "wink", "shake"), and pointers that describe the relations between one synset and other synsets. A word or collocation may appear in more than one synset, and in more than one part of speech. The words in a synset are grouped such that they are interchangeable in some context (Princeton University, 2010). Two kinds of relations are represented by pointers: lexical and semantic. Lexical relations hold between semantically related word forms; semantic relations hold between word meanings. These relations include (but are not limited to) hypernymy/hyponymy (super-ordinate/subordinate), antonymy, entailment, and meronymy/holonymy (Princeton University, 2010). Verbs are organized into hierarchies based on the hypernymy/hyponymy relation between synsets. Additional pointers are used to indicate other relations. As discussed previously, the WordTree Design-by-Analogy method uses files in the WordNet database composed of only verbs which include: the verb index file and the verb data file.

1. *The WordNet Verb Index File*

A line from the verb index file is shown in Table 5 to illustrate its format. The database format is in ASCII and can be viewed with an editor or text-based UNIX tool.

Table 5: A line from the verb index file illustrating its format

Abbreviate	v	2	4	@	~	$	+	2	0	00243900	00243749

In the field descriptions of the verb index file, **number** always refers to a decimal integer unless otherwise defined. The verb index file format is as follows:

[lemma pos synset_cnt p_cnt [ptr_symbol...] sense_cnt tagsense_cnt

synset_offset [synset_offset...]]

The meaning of each field in the index file is described in Table 6.

Table 6: Definition of fields in the index file (Princeton University, 2010)

Field	**From Table 5**	**Meaning**
Lemma	*abbreviate*	*Lower case ASCII text of word or collocation. Collocations are formed by joining individual words with an underscore (_) character.*
Pos	*v*	*Syntactic category:* **n** *for noun files,* **v** *for verb files,* **a** *for adjective files,* **r** *for adverb files.*
synset_cnt	*2*	*Number of synsets that lemma is in. This is the number of senses of the word in WordNet.*
p_cnt	*4*	*Number of different pointers that lemma has in all synsets containing it.*

Table 6 continued

Field	**From Table 5**	**Meaning**
Ptr_symbol	@ ~ $ +	*A space separated list of p_cnt different types of pointers that lemma has in all synsets containing it*
sense_cnt	*2*	*Same as synset_cnt above. This is redundant, but the field was preserved for compatibility reasons.*
tagsense_cnt	*0*	*Number of senses of lemma that are ranked according to their frequency of occurrence in semantic concordance texts.*
synset_offset	00243900 00243749	*Byte offset in **data.pos** file of a synset containing lemma. Each synset_offset in the list corresponds to a different sense of lemma in WordNet. synset_offset is an 8 digit, zero-filled decimal integer that can be used to read a synset from the data file.*

2. *The WordNet Verb Data File Format*

An example line from the verb data file is shown in Table 7 to illustrate the format. Similarly, the database format also in ASCII and can be viewed with an editor or text-based Unix tool.

Table 7: Sample of a verb data file field

00019182	29	v	01	reawaken	. . .	|	awaken once again

All the lines in the data file are in the format shown below. Integer fields are of fixed length, and are zero-filled to keep each column symmetric. It is important to know how every field in the database is formatted because it facilitates code-writing (when querying the databases) and prevents errors from mismatching fields.

[synset_offset lex_filenum ss_type w_cnt word lex_id [word lex_id...] p_cnt [ptr...] [frames...] | gloss]

The meaning of each field in the verb data file database is described in Table 8.

Table 8: Definition of fields in the verb data file

Field	**From Table 7**	**Meaning**
synset_offset	00019182	*8 digit decimal integer*
lex_filenum	29	*Two digit decimal integer corresponding to the lexicographer file name containing the synset. (29:Verb pertaining to body)*
ss_type	*v*	*One character code indicating the synset type: noun, verb, adjective, adjective satellite or adverb.*
w_cnt	*01*	*Two digit hexadecimal integer indicating the number of words in the synset.*
Word	*reawaken*	*ASCII form of a word as entered in the synset by the lexicographer.*
lex_id	*0*	*One digit hexadecimal integer that, when appended onto lemma, uniquely identifies a sense within a lexicographer file.*
p_cnt	*001*	*Three digit decimal integer indicating the number of pointers from this synset to other synsets.*
Ptr	@	*A pointer from this synset to another.*

Table 8 continued

Field	**From Table 7**	**Meaning**
Frames	00018813	*In the* **verb** *data file only, a list of numbers corresponding to the generic verb sentence frames for words in the synset.*
Gloss	|awaken once again	*Each synset contains a gloss. A gloss is represented as a vertical bar (|), followed by a text string. The gloss may contain a definition, one or more example sentences, or both.*

The HowNet ontology

HowNet is a bilingual lexical ontology for English and Chinese (Veale, 2005). HowNet and WordNet have a different view of semantic organization. In WordNet, rather than attempting to express the meaning of a word explicitly, WordNet instead differentiates words with different meanings by placing them in different synonym sets, and further differentiates these synsets from one another by assigning them to different positions in its taxonomy (Veale, 2005). In contrast, HowNet does not provide a human-oriented textual gloss for each lexical concept, but instead combines sememes from a less discriminating taxonomy to compose a semantic representation of meaning for each word sense (Veale, 2005). Research performed by Veale concluded that HowNet contains sufficient structure to realistically support both a taxonomic abstraction view *and* a structure-mapping view of analogy generation (Veale, 2005). For example in HowNet 手术刀 which is Chinese for “scalpel” (surgical knife) contains not just characters, but ideas. 手术 means “surgery” and 刀 means “knife” (Veale, 2005). This transparency in the makeup of words (etymology) allows for a broader scope of relations than the word “scalpel” as you would find in WordNet. This broader scope is expected to have an effect in analogy identification as it clearly allows an observer to make a broader connection between ideas.

Retrieval Systems and Visualization Tools

A retrieval system is a tool for people actively searching for information. Retrieved information can be applied to idea generation that focuses on Design-by-Analogy. Some retrieval tools access databases to obtain stored information for use in different applications. One of the applications, as is the focus of this thesis, is the WordTree Design Method which can use a retrieval tool for its application. Information used to stimulate creativity is sometimes stored in a repository. A repository is a place where knowledge is stored for later use. Most information can be reused and it is essential to save such information in an accessible location. Some repositories provide a wealth of knowledge (e.g. product designs, components, pictures, etc.) that could include potential analogy triggers that a user can use to jumpstart creativity during the idea generation stage. This section will discuss existing retrieval and visualization tools that can be applied to Design-by-Analogy.

VisualizeIT

VisualizeIT is a project seeking to identify a scientific basis and develop the supporting cyber infrastructure needed to facilitate, evaluate, and disseminate information-technology-enabled innovation methodologies that augment designer creativity (English et al., 2010). The visualization tool is used for design problems and works by the approach shown in Figure 6. The tool accesses a repository of stored design knowledge, and empirical grammar rules are used for retrieving the information and present them in the form of component flow graphs (CFG's). The user chooses a design problem from a list and is presented with a list of functional models. The user then selects a functional model and is presented with a list of clustering schemes. Finally, once the user selects a scheme he/she is presented with list of candidates for the proposed solution.

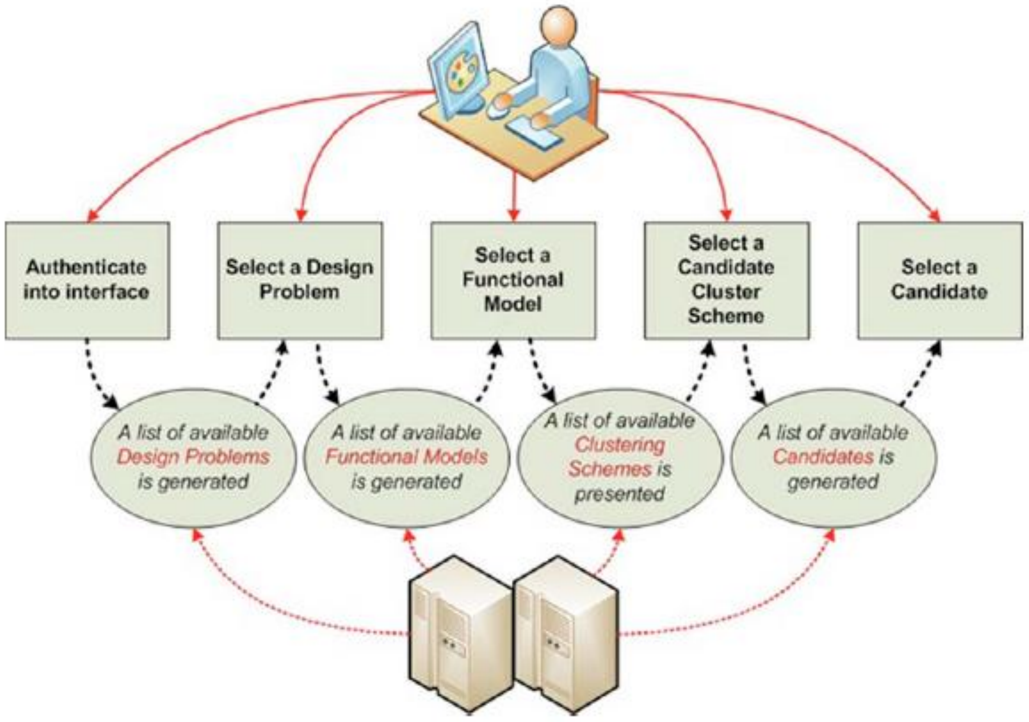

Authenticate into interface
Select a Design Problem
Select a Functional Model
Select a Candidate Cluster Scheme
Select a Candidate
A list of available Design Problems is generated
A list of available Functional Models is generated
A list of available Clustering Schemes is presented
A list of available Candidates is generated

Figure 6: Overview of the VisualizeIT approach

The VisualizeIT tool provides an alternate approach for the kind of information that can be stored when designing a repository in advancing the use of Design-by-Analogy for engineering design problems.

REBUILDER: *A Computer Aided Software Engineering (CASE) Tool for Analogy Retrieval*

REBUILDER is a Case-Based Reasoning (CBR) tool that uses several types of knowledge in the domain of computer software, including WordNet as the ontology (Gomes et al., 2006). Figure 7 shows the architecture of REBUILDER. There are four main modules: UML editor, knowledge base manager, knowledge base (KB), and the CBR engine. There are two user types: software designer and KB administrator. The software designer uses REBUILDER as a case tool, while KB administrator is responsible for keeping the KB updated and consistent. The KB consists of four parts: the case library, which stores the case of previous software designs; an index memory used for efficient case retrieval; a data type taxonomy; and WordNet, which is a general purpose ontology (Gomes et al., 2006).

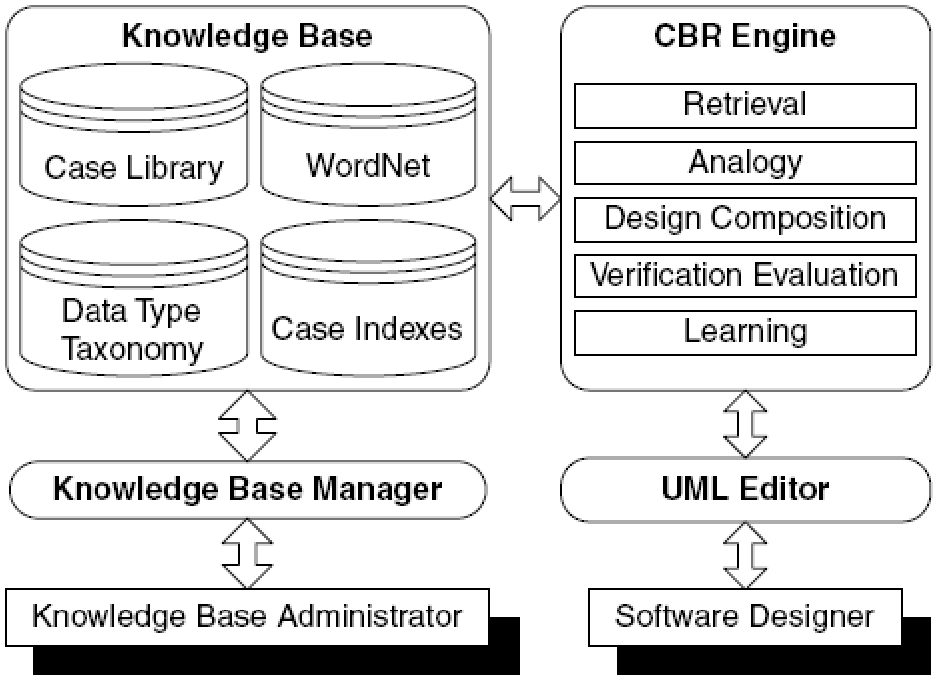


Figure 7: REBUILDER's architecture (Gomes et al., 2006)

REBUILDER uses analogical reasoning to suggest class diagrams to the designer. There are three steps to the analogy process: Identify candidate diagrams for analogy; map each candidate diagrams with the target diagram; create new diagrams, by knowledge transfer, between the candidate diagram and the target one (Gomes et al., 2006). From preliminary experiment results, it was inferred that semantic retrieval generates more useful class diagram, but they are less novel than diagrams using structural strategies (Gomes et al., 2006). In other words, the structural strategies are a predefined organization of class diagrams.

combinFormation (cF)

Research has shown that image and text knowledge representations are more effective than text only. Cognitive research by Glenberg shows that the combination of an image and a descriptive text promotes the formation of mental models (Glenberg & Langston, 1992). This was comparing combinFormation is a mixed initiative system for representing collections as compositions of image and text surrogates (Koh et al., 2007). A surrogate represents an information resource and enables access to that resource (Burke, 1999; Koh et al., 2007). The combinFormation mixed initiative process is shown in Figure 8.

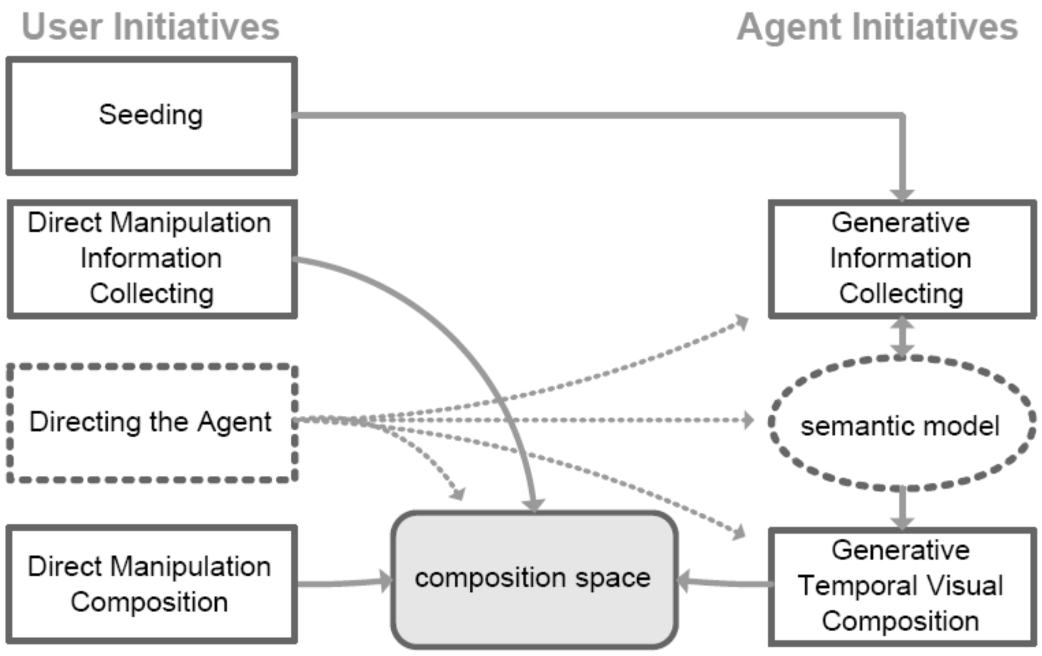
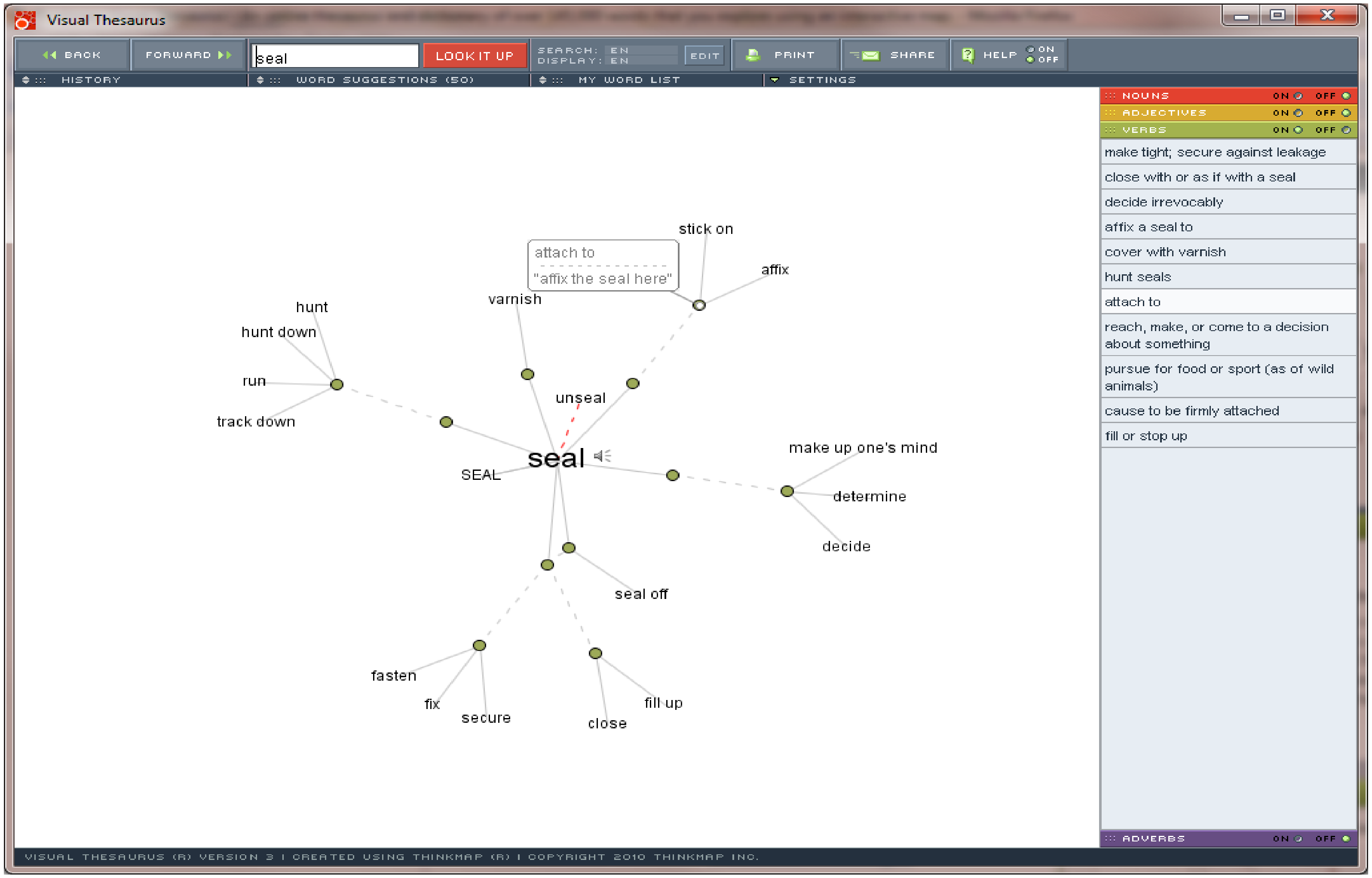

User Initiatives
Agent Initiatives
Seeding
Direct Manipulation Information Collecting
Directing the Agent
Direct Manipulation Composition
composition space
Generative Information Collecting
semantic model
Generative Temporal Visual Composition

Figure 8: combinFormation mixed initiative process

Through the composition space, the user and the agent engage in mixed initiatives. Through seeding, the user points the agent at particular information sources. Through direct manipulation information collecting, the user brings surrogates, and their underlying semantics, directly into the composition space and the model. Through direct manipulation and composition, the user changes how the composition looks in order to facilitate his/her own understanding of the information resources and their connections, and perhaps to communicate such understanding to others.

Visual Thesaurus

Visual Thesaurus (Thinkmap, 2010b), developed using Thinkmap software (Thinkmap, 2010a), is a visualization tool that enables a user to visualize relationships between synonyms in an interactive interface. Figure 9 shows an example display of Visual Thesaurus for the word seal. The thinkmap software is considered useful for applications involving the visualization of large amounts of information on a screen. This applicability could be used in displaying the output of an ontology-based library (such as WordTrees) for analogy search. Additional features such as those found in the Visual Thesaurus software could be applied to a WordTree generating tool. For example, displaying the definitions of words by simply clicking the words on the tree would be better than doing a separate search for the definition.



make tight; secure against leakage

close with or as if with a seal

decide irrevocably

affix a seal to

cover with varnish

hunt seals

attach to

reach, make, or come to a decision about something

pursue for food or sport (as of wild animals)

cause to be firmly attached

fill or stop up

Figure 9: Visual Thesaurus display for the word "seal" (Thinkmap, 2010a)

Chapter Conclusion

This chapter has presented relevant background for the proposed direction of this thesis. The goal of advancing the WordTree Design-by-Analogy method from its current state to a more applicable and sought after one begins with the understanding of its development. The WordTree Method uses WordNet-based WordTrees as its main resource for searching for, and identifying potentially useful analogies for solving design problems. The makeup of the WordNet-based WordTrees is governed by the WordNet ontology. This chapter has described the WordNet ontology and its related components. A study of other ontologies was done to provide insights for understanding ontology development, and also to recommend possible future directions. The YMIR ontology focused on using information from multiple engineering domains for its development; this provides stronger forms of integration between the different domains. This chapter has studied retrieval and visualization tools used in the context of information management and organization. The following chapters will discuss the development of an automated WordNet-based WordTree generating tool called the WordTree Express, and experiments designed to test the effectiveness of the tool.

CHAPTER III

WORDTREE EXPRESS PROGRAM DESCRIPTION

The WordTree Express (WTE) program is a computer program that automates the application of the WordTree Design-by-Analogy method. The WTE program works in conjunction with another program, Graphviz (Ellson, J et al., 2010), that reads and displays the output graph. WTE program was designed with the user experience held as the focal point for the design. The user of the program is expected to find its interface easy to understand and use. The program's design minimizes the number of steps the user takes to display the desired output from the program. This chapter lays out the foundation for the development of the WTE program. The topics covered are as follows:

- Goals of the WTE program
- WTE user interface layout
- WTE program code layout
- Challenges of the design
- Benefits of the design
- Tutorial

Goals of the WordTree Express Program

The WTE program was designed to possess attributes formed by the recognized user's need and expected program functions. The needs and desired functions include:

a. Simplicity: The need for making any design process as simple as possible is essential in industry where time is of the essence. Designers always prefer simplicity as it makes the design process more efficient.

b. Easy on the eyes: Some care was taken in designing the user interface to make the screen view more comfortable. Soft colors with well defined buttons and soft backgrounds were implemented to satisfy this attribute.

c. Provides the user with a selectable sense option: In order to develop a WordTree having more than one possible output, the program has to offer options to the user to produce the user's desired output. Implementing the checkbox options for the keyword senses was a needed feature for the user.

d. Generates readable WordTrees: The Graphviz program can display WordTrees, but has a display size limitation. Microsoft Office Visio and Inkscape are optional viewing programs for Scalar Vector Graphics (SVG) files and are capable of displaying the WordTrees without any limitation.

WordTree Express User Interface Layout

The WTE program has a total of three buttons as shown in Figure 10. The first button, "Search", runs a search query in the WordNet database of the typed keyword and displays the related senses in a checklist box format. The second button, "Create file & Start Graphviz", is used after a selection of a keyword sense is made; it generates a text file in a format that the Graphviz program can read, and simultaneously starts the Graphviz program for the user. This combined function with a single button was done to save time and make the program less complicated. Finally, there is a reset button that allows the user to clear all the fields and perform new keyword searches without having to restart the program or manually clearing the fields.

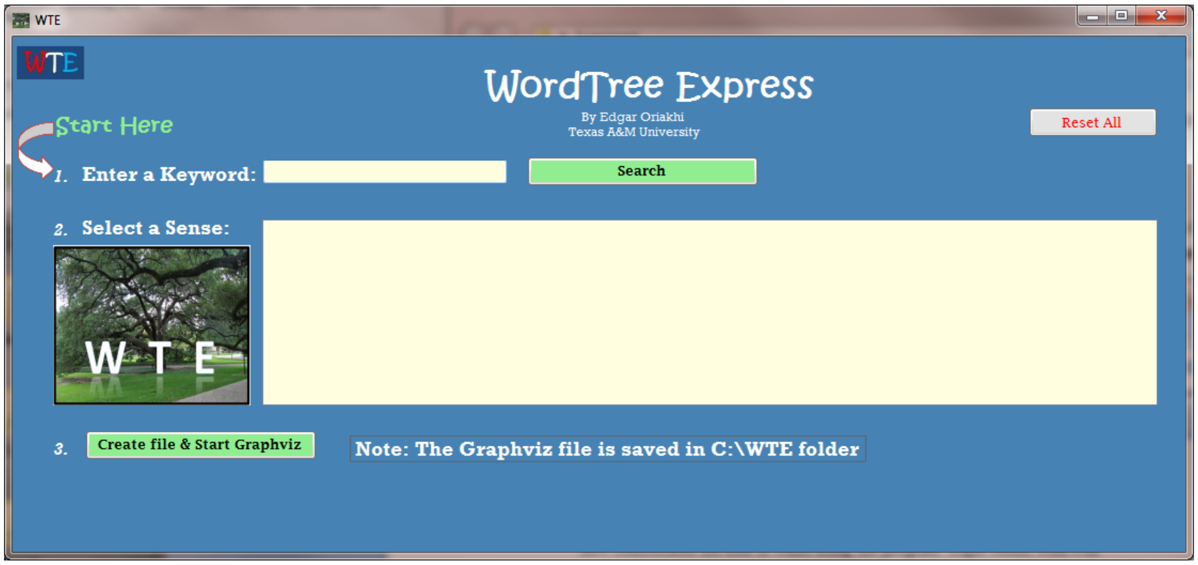


Figure 10: WordTree Express program layout

WordTree Express Program Code Layout

In the pre-programming planning phase of the WTE program, the following questions had to be answered:

1. How was WordNet database structured?
2. What information was needed from the WordNet database for the program?
3. What programming language was to be used for the coding?
4. What did the user need the program to do?
5. What was the layout suppose to look like?

To answer these questions, some research was done. The WordNet structure was studied and has been described in Chapter II. A conclusion from studying the WordNet database reveled that it contained an index data file folder and folders of words for the different parts of speech; for the application of the WordTree, only two folders would be needed. The needed folders for the application were the verb index data file and the verb data file. The next step was selecting the right programming language. Microsoft's Visual Basic (VB) was chosen because of its simple structure and the available resources. In determining what the program must do, a review of the non-automated approach was studied; the study revealed that the user must be able to perform a keyword search, have the ability to choose the desired sense of the keyword, and have the complete WordTree displayed. Finally, from the laid out user's need, a user interface was designed.

An overview of the major steps taken in the WordTree Express program development will now be discussed in detail.

Step 1: Creating form layout

Here the goal was to lay out an overall structure that would be filled in progression. Using visual basics, the overall layout is described and shown in Figure 11. All the necessary objects were placed on a new form (shown with colored dots). These objects included: 2 textboxes (purple), 1 check list box (red), 3 buttons (green), 9 labels (blue), and 3 pictures (orange).

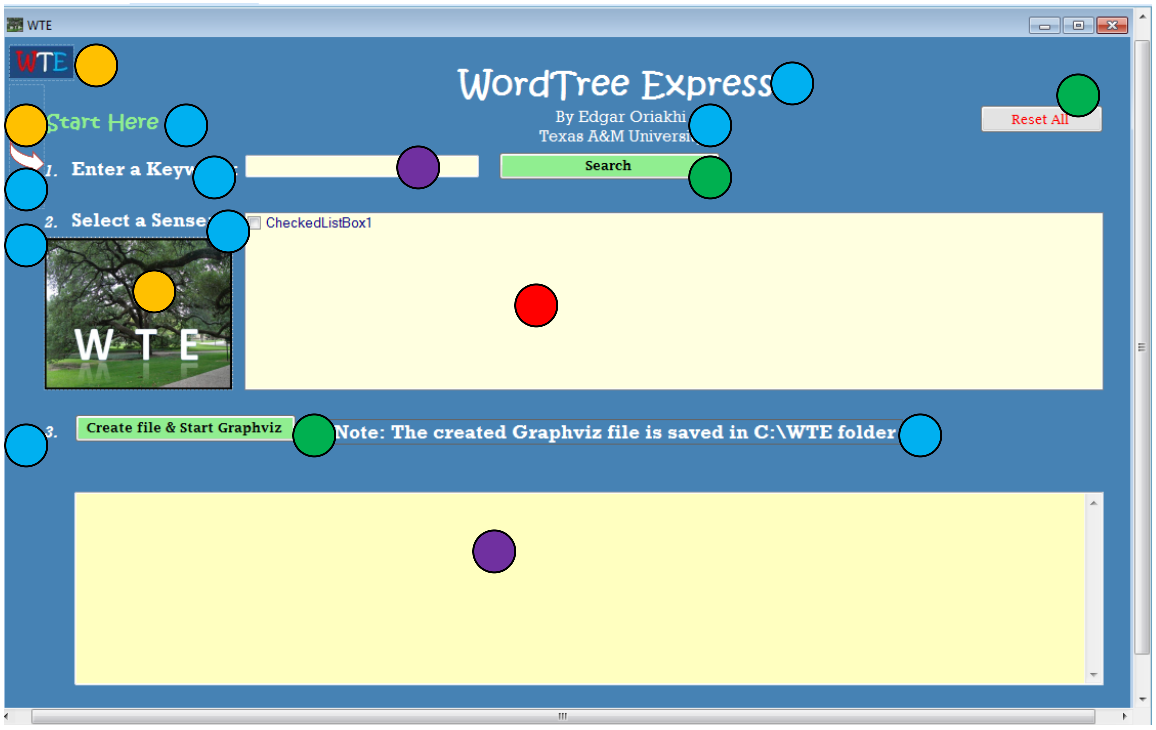


Figure 11: WTE program layout

Step 2: Importing the needed database files

As mentioned in the previous section, there are two files needed for the application of the WordTree Method: the index data file and the verb data file. In the program, these files were named index.txt and database.txt respectively. The code in VB to import the two files from their stated location is shown in Figure 12.

```
    Private Sub funcLoadFiles()
        Try
            Dim sr As System.IO.StreamReader = System.IO.File.OpenText("C:\WTE database\index.txt")
            Dim StrArray(), strLine() As String
            Dim intRow, intTotal As Integer
            Dim sData As String
            'string = contents of file
            sData = sr.ReadToEnd
            'fill array with data
            StrArray = Split(sData, ControlChars.NewLine)
            'close stream
            sr.Close()
            intTotal = StrArray.Length – 1
            For intRow = 0 To intTotal
                strLine = StrArray(intRow).Split(" ")
                _arrWord.Add(strLine(0))
                _arrMeaning.Add(StrArray(intRow))
            Next
            sr = System.IO.File.OpenText("C:\WTE database\database.txt")
            sData = sr.ReadToEnd
            'fill array with data
            StrArray = Split(sData, ControlChars.NewLine)
            sr.Close()
            intTotal = StrArray.Length – 1
            For intRow = 0 To intTotal
                strLine = StrArray(intRow).Split(" ")
                _arrDefNum.Add(strLine(0))
                _arrDefinition.Add(StrArray(intRow))
            Next
        Catch ex As Exception
        End Try
    End Sub
```

Figure 12: VB programming code for importing database files

Step 3: Creating the search query function

In creating the function to search the user's input keyword in the index data file and returning a list of sense options, the following code in Figure 13 was written. In creating the code some scenarios were identified that had to be accounted for in the coding. One was replacing the user's space with an underscore because multiple word phrases in the database are stored with underscores separating them. For example the phase "back away" is stored as "back_away" in the database file. For another scenario, a code was written so that if the input search keyword is not found in the database, the following message is shown to the user: "The word you typed does not exist in the database".

```
Private Sub funcSearch()
    Try
        File.Create("C:\WTE\" & txtSearch.Text & "")
        Dim arrDefined As New ArrayList
        Dim strSearch As String = txtSearch.Text
        Dim strMeaning, strWord As String
        Dim intRow, intNumRows, intPtrCnt, intNewCnt As Integer
        Do While (strSearch.IndexOf(Space(1)) >= 1)
            strSearch = strSearch.Replace(Space(1), "_") 'Replaces spaces with "_" in the input.
        Loop
        intNumRows = _arrWord.Count – 1
        For intRow = 0 To intNumRows
            If strSearch.ToLower = _arrWord(intRow).ToString.ToLower Then
                Exit For
            End If
```

Figure 13: VB code for the search function

```
            If intRow = intNumRows Then
                MsgBox("The word you typed does not exist in the database")
        End If
    Next
    strMeaning = _arrMeaning(intRow)
    Dim strMean() As String
    strMean = strMeaning.Split(" ")
    intPtrCnt = Cint(strMean(3))
    _arrSenses.Clear()
    For intNewCnt = (intPtrCnt + 6) To strMean.Length – 1
        strWord = strMean(intNewCnt).Trim
        If String.IsNullOrEmpty(strWord) = False Then
            _arrSenses.Add(strMean(intNewCnt))
            Console.WriteLine(strMean(intNewCnt))
        End If
        Dim strNdef, strWordW As String
        Dim intWordCnt, intRowN, intNumRowsN, intNewCntN As Integer
        Dim strDefinition As String
        intNumRowsN = _arrDefNum.Count – 1
        For intRowN = 0 To intNumRowsN
            If strMean(intNewCnt) = _arrDefNum(intRowN) Then
                Exit For
            End If
        Next
        strNdef = _arrDefinition(intRowN)
        Dim strNWord() As String
        strNWord = strNdef.Split(" ")
        intWordCnt = Cint(strNWord(3))
        _arrSensesW.Clear()
```

Figure 13 continued

```
            If strNWord(3) = “0a” Then
            strNWord(3) = 10
          ElseIf strNWord(3) = “0b” Then
            strNWord(3) = 11
          ElseIf strNWord(3) = “0c” Then
            strNWord(3) = 12
          ElseIf strNWord(3) = “0d” Then
            strNWord(3) = 13
          ElseIf strNWord(3) = “0e” Then
            strNWord(3) = 14
          ElseIf strNWord(3) = “0f” Then
            strNWord(3) = 15
          End If
          strWordW = Nothing
          For i = 4 To (strNWord(3) * 2) + 2
            strWordW += strNWord(i) & “(“ & strNWord(1) & strNWord(i + 1) & “)” & “,”
            i = i + 1
          Next
          strWordW = strWordW.Substring(0, strWordW.Length – 1)
          For intNewCntN = 0 To strNdef.Length – 1
            If strNdef(intNewCntN) = “|” Then
              strDefinition = strNdef.Substring(intNewCntN + 1)
              CheckedListBox1.Items.Add(strWordW & “ “ & “→” & “  “ & strDefinition)
            End If
          Next
        Next
        funcShowTree(Cint(_arrSenses(CheckedListBox1.SelectedIndex)))
      Catch ex As Exception
      End Try
    End Sub
```

Figure 13 continued

Step 4: Creating a function to display a message box to inform the user of the program status

After a set of options is presented to the user to make his/her selection of the keyword sense, the program will notify the user, when he/she presses the “Create file and start Graphviz” button, where the Graphviz WordTree file being created is stored. The code for the function is shown in Figure 14.

```
Private Sub funcGraphViz()
    Try
       Dim FILE_NAME As String = “C:\WTE\” & txtSearch.Text & “”
       If System.IO.File.Exists(FILE_NAME) = True Then
          Dim objWriter As New System.IO.StreamWriter(FILE_NAME)
          objWriter.Write(TextBox3.Text)
          objWriter.Close()
          MsgBox(“A new Text file named “ & “”” & txtSearch.Text & “”” & “ has been created
and saved in C:\WTE folder.” & ControlChars.NewLine & “Graphviz will now start. Please open the
new file to view the WordTree”)
       Else
          MsgBox(“File Does Not Exist”)
       End If
    Catch ex As Exception
    End Try
 End Sub
```

Figure 14: VB code for status message box

Step 5: Creating the function to generate the WordTree in a Graphviz readable format

This section was the main part of the coding and the raw code can be found in the Appendix B (The function was named: funcShowTree). The program was designed to read the database line by line and store the selected information in matrices it creates as needed. Information from the created matrices is later written to a hidden text box within the program. Based on the nature of the database structure, this section has multiple sub-functions. In the database, each word/phrase is represented by an 8-digit number. This function works by reading the user's selected keyword sense from the database.txt file then checks if the keyword has a hypernym in the related field (identified by an "@" symbol preceding it). If the keyword has a hypernym the program repeats the query as needed, for each hypernym found, until it gets to the top of the tree. Once the top-most word is found the function saves the word into a matrix and does a different sub-function. It starts to look for troponyms (identified by a "~" symbol preceding it). The term "children" is used in the code when referring to troponyms. For each of the children found, the function saves the word into a matrix and continues in a series of loops to find the troponyms of each child and so on. As all this is taking place, the function is also translating and storing the word version of each troponym (originally represented by numbers in the database).

The main problem in this section was differentiating words that were spelt the same way, but had different sense of use. This problem caused the final WordTree display to look disorganized by having lines between words cross each other, and for large WordTrees, almost impossible to read. The solution required adding identifiers to uniquely identify each word or phrase based on which synset and sense it belong to. As seen in the WordTree output, succeeding every word/phrase is a three or four digit number. The first two digits of the number represent the synset the words belong to, while the third and fourth digits represent the senses of the words. Adding these identifiers made the WordTrees to display properly.

Step 6: Creating the reset button

The function to clear all the fields in the form for performing new keyword searches was created using the code shown in Figure 15.

```
Private Sub ClearForm()
    For Each ctrl As Control In Me.Controls
      If TypeOf ctrl Is TextBox Then
        DirectCast(ctrl, TextBox).Text = String.Empty
      End If
    Next
    CheckedListBox1.Items.Clear()
  End Sub
```

Figure 15: VB code for the reset button

Step 7: Assigning functions to the buttons

This section was performed as needed during the code-writing. The functions were assigned trigger buttons, and some buttons performed multiple tasks such as running two or more different functions. For example the codes in Figure 16 were assigned to the "Create file and start Graphviz" button. The "search" button and the "reset" button were triggers for the functions shown in Figures 17 and 18 respectively.

```
  Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button2.Click
          Cursor.Current = Cursors.WaitCursor
      funcShowTree(CInt(_arrSenses(CheckedListBox1.SelectedIndex)))
      funcGraphViz()
      System.Threading.Thread.Sleep(2000)
      Dim p As New System.Diagnostics.Process
      p.StartInfo.FileName = "Gvedit.exe"
      p.Start()
          End Sub
```

Figure 16: VB function codes assigned to the "create file and start graphviz" button

```
Private Sub btnGenerate_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnGenerate.Click
     funcSearch()
 End Sub
```

Figure 17: VB search function code assigned to the "search" button

```
Private Sub BtcClear_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles BtcClear.Click
    ClearForm()
End Sub
```

Figure 18: VB reset function code assigned to the "reset" button

Benefits of the Design

It will only be fitting, at this point, to shed some light on the expected benefits of the WTE design as compared to the non-automated approach. A study of the precedent method revealed the following proposed benefits of the new design.

1. The user will not need to deal with the process of online search. All the full features of the program can be used offline as opposed to a limited offline feature in the case of the non-automated approach.
2. The user will not need to spend time sorting and selecting each word needed for the WordTree as this may lead to a limited range for idea generation and a waste of valuable time.
3. The user will not need to manually type the words in the WordTree, as is done in the non-automated approach; this also saves valuable time.

From the stated benefits, rewards from the program point to time savings, easy application process, and availability for use without an internet connection.

Tutorial for the WordTree Express

For optimal application, as is the case for almost every product, a tutorial (or user’s guide) was developed to assist the user in getting acquainted with the WordTree express program. The tutorial can be found in Appendix B; it covers the installation procedures and a step-by-step description of how to create a WordTree using the WTE program.

CHAPTER IV

EXPERIMENT: WORDTREE DESIGN METHOD

Overview

The WordTree Design Method was developed to assist engineers and designers in the idea generation stage of a Design-by-Analogy approach to design problems. A study by Linsey et al., on the effects of memory representation on analogy use supports the assertion that the form of concept representation is important in the cognitive analogy formation process (Linsey et al., 2008). The study was one of the drivers for the development of the WordTree Method. Furthermore, a controlled study of the WordTree Method showed that the method assisted engineers in identifying more analogies and altered their database search patterns which resulted in cross-domain solutions being found (Linsey, 2007). The study also showed that the method needed to provide a better support for the mapping of identified analogies into solutions (Linsey, 2007). In the controlled study of the WordTree Method, participants tended to identify large numbers of analogies, but then a high percentage did not inspire conceptual solutions (Linsey, 2007). Participants ranked the WordTree Method among the least valuable methods for their future use and also for design problems that required innovative solutions. There were 13 methods in total and the WordTree Method's ranking was similar to the TIPS/TRIZ, morph matrix, and 6-3-5 which were among the lowest scores while the other 9 methods ranked higher. It is also important to note that TIPS/TRIZ is a highly valued method by industry.

The method's developer suggested possible reasons for the outcome was from the methods presentation to the participants; that it could use some more powerful examples and strongly highlight the purpose of the method. The method's developer suggested that another reason could be because of some of the participants' lack of experience with the method or lack of skill in Design-by-Analogy. This thesis suggests that an automated WordTree generating tool to simplify part of the method's application (i.e. creating WordTrees), could be used in facilitating the teaching of the method and positively affect the users opinions about the method.

Research Questions and Hypothesis

The questions this thesis seeks to answers to are the following:

1. Does the WodrTree Express program affect engineering designers' opinions of the WordTree Method? Does simplifying the process of generating WordNet-based WordTrees have a positive effect on the opinions of engineers when asked to rate the value of the WordTree Method against other design methods for each of the following:
    a. A typical engineering design problem.
    b. A design problem that requires an innovative solution.
    c. How likely they would use the method in the future.
2. Does providing more comprehensive WordTrees (via WordTree Express) for chosen problem descriptors affect the number of analogies identified and used for conceptual solutions to the design problem?

3. What are some of the additional avenues for improvement to the WordTree Design-by-Analogy Method?

To investigate these research questions the following hypotheses were proposed:

Hypothesis 1: WTE, by simplifying the process of generating WordNet-based WordTrees, will increase designers' opinions of the WordTree Method.

Hypothesis 2: Using the WordTree Express program to create WordTrees will present the user with a more comprehensive WordTree; increasing the number of identified analogies.

This thesis investigated the research questions by first performing a repeated measures study of the participants. This was accomplished by surveying the participants who were taught different design methods including the WordTree Method without the use of an automated WordNet-based WordTree generating tool, and surveying the participants after the experiment in which they used the automated WordTree generation tool (WTE). Secondly, the study replicated the original WordTree Method control study with some minor modifications and compared both results. Table 9 summarizes the differences between the original WordTree control study and that done in this thesis.

Table 9: Difference between Linsey et al., 2008 study and current study

Linsey et al., 2008 WordTree Control Study	Current Study
Participant were undergrad students	Participants were graduate students
Senior capstone course during one 50 minute lecture	60 minutes graduate design course
Included re-writing problem statements	Did not include re-writing problem statements
Participant did not have to generate WordNet-based WordTrees, they were provided with them	Participants were asked to generate their WordNet-based WordTrees using WTE
10 participants in the WordTree condition and 10 in the control condition.	15 participants in the WTE condition

Method

Participants

The participants were graduate Mechanical Engineering students at Texas A&M University. All the participants were recruited from a graduate design class and were compensated for their participation in the experiment with extra credit in their class.

Procedure

The WordTree Method was taught to a graduate design course during a 60 minute lecture. Participants were recruited from the graduate design course after they had shown their understanding of the method from the results of an assignment on applying the WordTree Method to their project design problems. The participants were given extra credit for their participation and were told the amount of extra credit depended on their efforts and results. A total of 15 participants took part in the study. One of the participants was not an engineer, but a psychology graduate student taking the design course as an outside department course requirement. Most of the participants were PhD level students. The experiment procedure was as follows:

Step 1: Participants were shown to their sits, told that they could not monitor the time during the experiment, and were asked to turn off their cell phones and put their watches away. Participants were told the duration of the experiment was two hours and were given the consent forms to sign if they agreed to participate in the experiment.

Step 2: Participants were provided with a pre-experiment survey made up of Table 10, Table 11 and Table 12 which asked for their opinions about the WordTree Method and other design methods for different situations based on their experience with all the methods.

Table 10: Pre-experiment survey question 1

	Not at all	Somewhat	Useful	Very Useful
Overall, WordTree Method was:				
The WordTrees were:				
Listing analogies was:				
Listing analogous domains was:				
Writing new problem statements				

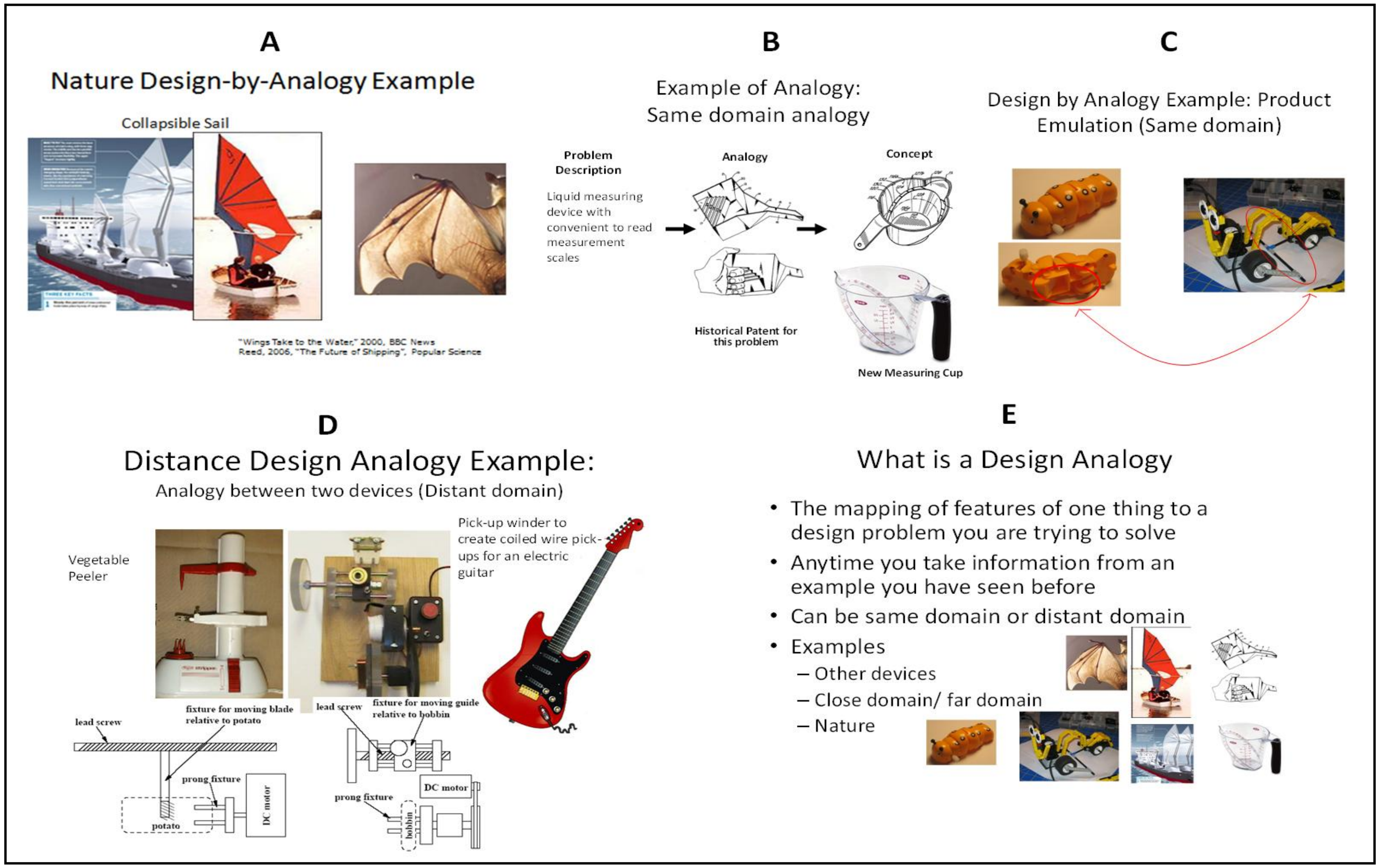
Table 11: Pre-experiment survey questions 2, 3 and 4 (Value for typical, innovative and future use respectively)

	Zero value	A little value	Medium value	High value	Extremely valuable	Can't remember
Background research / Literature review						
Mission Statement						
Quality Function Development (QFD, House of Quality)						
Black Box diagram						
Activity Diagram						
Function Structure						
Patent Search						
6-3-5						
Mind Maps						
TRIZ/TIPS						
Morph Matrix						
Pugh Charts						
WordTree Method						

Table 12: Pre-experiment survey question 5

	Strongly Disagree	Disagree	Neither Agree Nor Disagree	Agree	Strongly Agree
1. This method helped me to find analogies for my design problem.					
2. This method helped me to generate more ideas.					
3. This method helped me to generate more quality ideas					
4. This method was a waste of my time.					
5. The presentation of this method was easy to understand.					
6. This method was easy to use.					
7. I expect to use this method in the future.					
8. This method needs improvements.					
9. This method was useful.					
10. I like using the method.					
11. I expect to use this method in the future for design problems that require an innovative solution.					

Step 3: The participants were told that multiple color pens would be used to keep track of when items were written. Examples of analogies were shown to the participants using the PowerPoint slide shown in Figure 19.

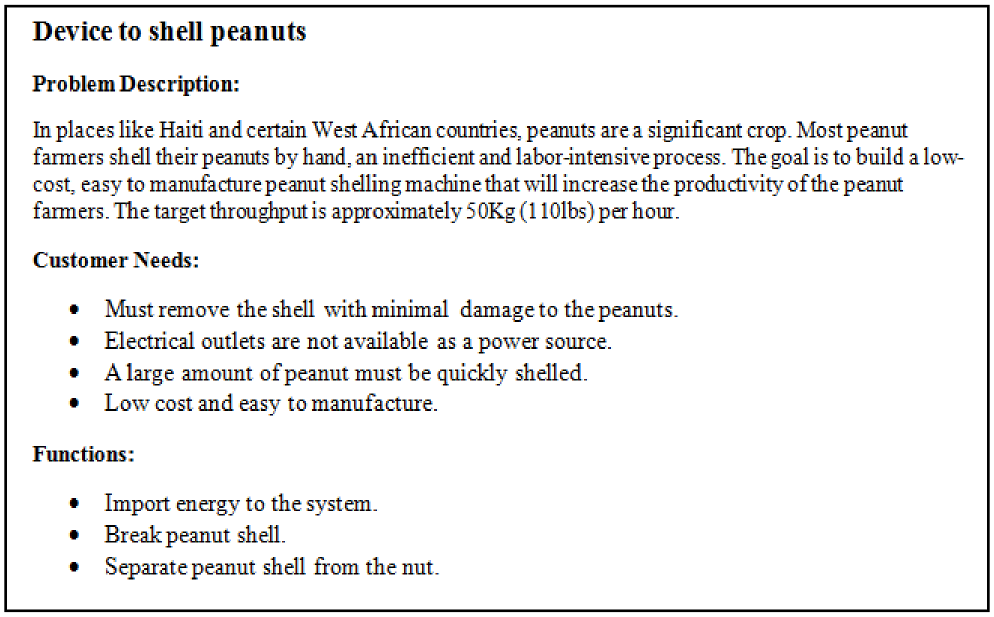

A
Nature Design-by-Analogy Example
Collapsible Sail
"Wings Take to the Water," 2000, BBC News
Reed, 2006, "The Future of Shipping", Popular Science
B
Example of Analogy: Same domain analogy
Problem Description
Liquid measuring device with convenient to read measurement scales
Analogy
Historical Patent for this problem
Concept
New Measuring Cup
C
Design by Analogy Example: Product Emulation (Same domain)
D
Distance Design Analogy Example:
Analogy between two devices (Distant domain)
Vegetable Peeler
Pick-up winder to create coiled wire pick-ups for an electric guitar
lead screw
fixture for moving blade relative to potato
prong fixture
potato
DC motor
lead screw
fixture for moving guide relative to bobbin
prong fixture
DC motor
bobbin
E

What is a Design Analogy

- The mapping of features of one thing to a design problem you are trying to solve
- Anytime you take information from an example you have seen before
- Can be same domain or distant domain
- Examples
  - Other devices
  - Close domain/ far domain
  - Nature

Figure 19: Analogy example slides shown to the participants

Step 4: Participants were provided with the design problem shown in Figure 20. The participants were told that the design problem was real and from the website thinksycle.org, and their solutions could be given to a design team working on the problem.



Device to shell peanuts

Problem Description:

In places like Haiti and certain West African countries, peanuts are a significant crop. Most peanut farmers shell their peanuts by hand, an inefficient and labor-intensive process. The goal is to build a low-cost, easy to manufacture peanut shelling machine that will increase the productivity of the peanut farmers. The target throughput is approximately 50Kg (110lbs) per hour.

Customer Needs:

- Must remove the shell with minimal damage to the peanuts.
- Electrical outlets are not available as a power source.
- A large amount of peanut must be quickly shelled.
- Low cost and easy to manufacture.

Functions:

- Import energy to the system.
- Break peanut shell.
- Separate peanut shell from the nut.

Figure 20: Design problem presented to the participants

Step 5: The participants were asked to create sticky note WordTrees for 20 minutes for the problem descriptors: shell, remove, separate, and import energy. A printout of the WordTree Method reminder was also given to the participants for reference. The method shown on the WordTree Method reminder was slightly modified to eliminate re-writing problem statements and geared towards an individual rather than a team. The WordTree Method reminder can be found in Appendix A. Figure 21 shows the method that was taught to the participants in class, while Figure 22 shows the method the participants were asked to use for the study.

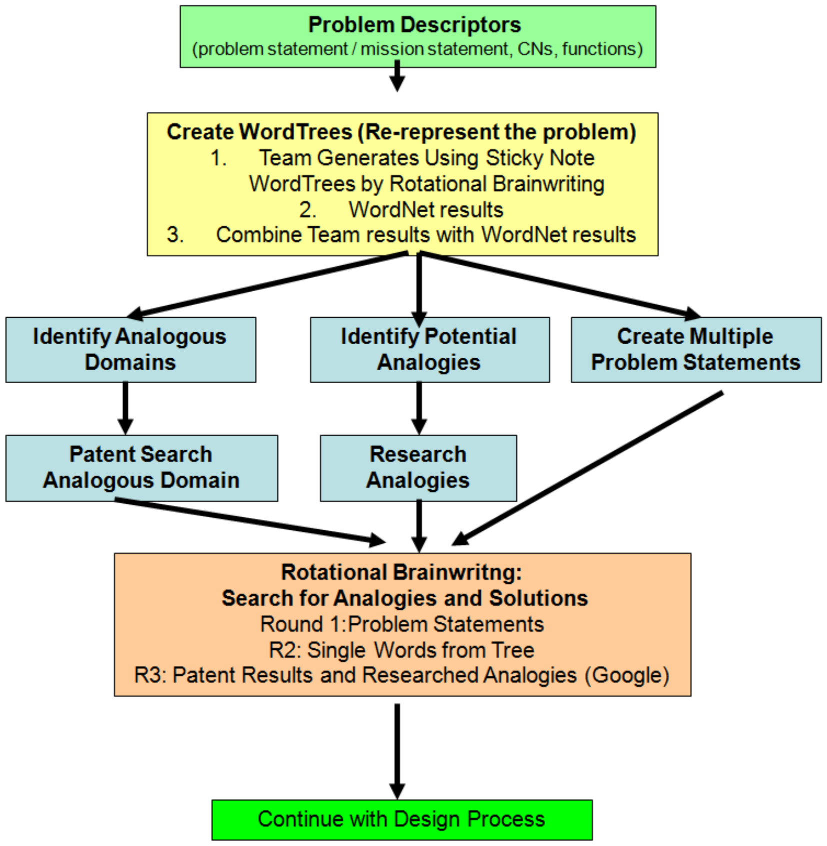

Problem Descriptors
(problem statement / mission statement, CNs, functions)
Create WordTrees (Re-represent the problem)
1. Team Generates Using Sticky Note WordTrees by Rotational Brainwriting
2. WordNet results
3. Combine Team results with WordNet results
Identify Analogous Domains
Identify Potential Analogies
Create Multiple Problem Statements
Patent Search Analogous Domain
Research Analogies
Rotational Brainwritng:
Search for Analogies and Solutions
Round 1:Problem Statements
R2: Single Words from Tree
R3: Patent Results and Researched Analogies (Google)
Continue with Design Process

Figure 21: WordTree Method as presented to the design class

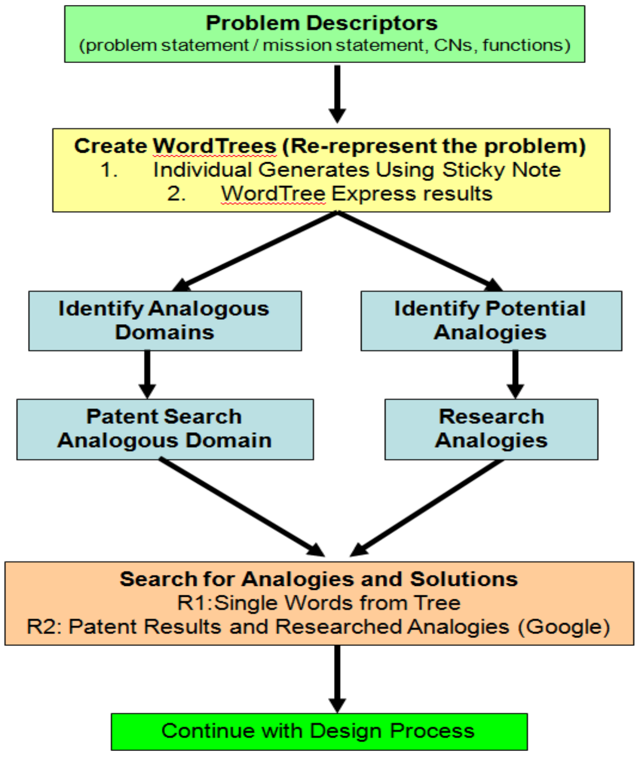

Problem Descriptors
(problem statement / mission statement, CNs, functions)
Create WordTrees (Re-represent the problem)
1. Individual Generates Using Sticky Note
2. WordTree Express results
Identify Analogous Domains
Identify Potential Analogies
Patent Search Analogous Domain
Research Analogies
Search for Analogies and Solutions
R1:Single Words from Tree
R2: Patent Results and Researched Analogies (Google)
Continue with Design Process

Figure 22: WordTree Method presented to the participants during the study

Step 6: Participants were asked to watch a recorded tutorial for the WordTree Express program and to use it to generated two WordTrees, one for the keyword "shell" and the other for the keyword "separate". The WTE program also dictated which sense of the keywords to use for their WordTrees. These were the two WordTrees presented to the participants in the control study. The participants were asked to circle all the words of interest on each WordTree that could lead to potential analogies. The participants had 30 minutes for this step which included 9 minutes for the tutorial video.

Step 7: Participants were asked to use 10 minutes to write down all the potential analogies they identified from all their WordTrees on a sheet of paper.

Step 8: The participants were provided with numbered sheets of paper to sketch and describe solutions to the peanut shelling problem. The total time for this activity was 60 minutes. Colors of the pens were changed during idea generation at the 15, 30, 40 and 45 minute marks. The participants were told that they could end the idea generation session at any time and were asked to fill out a questionnaire asking them why they decided to stop idea generation if they did. After participants filled out the questionnaire, they were provided with a sheet asking them to continue generating solutions as most people could still generate ideas even after they thought they ran out of ideas. After 45 minutes of idea generation, the participants were told that they could use the internet on the computers to assist them in generating ideas. They were told that it could be used to research the potential analogies they identified and to search for patents in the analogous domains. Web searches were optional and not required.

Step 9: In this step, the participants were again asked to note all the analogies they used for their final solutions and to describe how they searched the internet for solutions if they used that option; the participants were given a new analogy list sheet to fill.

Step 10: This was the final step in the experiment. The participants were provided with a post-experiment survey nearly identical to the first, but included questions specific to the WordTree Express program and a set of interview questions. The post-experiment survey included Tables 13 and 14 and the set of interview questions in Table 15.

Table 13: Post-experiment survey questions 2, 3 and 4 (value for typical, innovative and future use respectively)

	Zero value	A little value	Medium value	High value	Extremely valuable	Can't remember
Background research/Literature review						
Mission Statement						
Quality Function Development (QFD, House of Quality)						
Black Box diagram						
Activity Diagram						
Function Structure						
Patent Search						
6-3-5						
Mind Maps						
TRIZ/TIPS						
Morph Matrix						
Pugh Charts						

Table 13 continued

	Zero value	A little value	Medium value	High value	Extremely valuable	Can't remember
WordTree Method non-automated (Paper-based)						
WordTree Method automated (e.g. WordTree Express)						

Table 14: Post-experiment survey question 6

	Strongly Disagree	Disagree	Neither Agree Nor Disagree	Agree	Strongly Agree
I ran out of **time** before I ran out of ideas.					
I ran out of **ideas** before I ran out of time.					

Table 15: Interview questions

What did you like about the WordTree Method?
What steps in the WordTree Method were most useful?
How could the WordTree Method be improved?
What difficulties did you have when using the WordTree Method?
Where did you need more guidance from the WordTree Method?
What do you think of the WordTree Express program? How would you compare your experience when you made WordTrees manually to using this automated method?
Please state any additional comments you have about the experiment. Use the back of the paper if needed.

Metrics

Quantitative and qualitative measures were done in a similar fashion to the prior study. The metrics of interest include:

1. The number of analogies identified by the participants.
2. The number of ideas generated.
3. The percentage of analogies identified that were used to find solutions.
4. The number of participants who searched outside the domain of peanut shelling.
5. Opinions of the participants about the WordTree Method.

Metrics were scored by the experimenter. Analogies were calculated using two approaches. The first was from the number of analogies the participants listed on their analogy list sheets. It was noticed that many of the participants did not list all the analogies and analogous domains they identified in their WordTrees (i.e. the circled words). The second approach was done by counting the number of non-redundant analogies listed in either the analogy list sheets or those identified in the participants' WordTrees. Counting of analogies was done by the evaluator and a second evaluator with inter-rater agreement of 0.92 (Pearson's correlation). The search strategy used by the participants was scored from the search terms used that were outside the domain of peanut shelling. In this study the chosen criteria was similar to the 2007 study by Linsey et al., 2008. For example if a participant searched for "pod peas" or "pitting cherries" it would have be considered outside the domain, but if the search was for "peanut machine" or "universal nut sheller" as was the case in this study, it was considered within the domain of peanut shelling. The terms used for the evaluation came from those listed by the participants in the second analogy list sheet where they described their search strategy. Not all the participants chose to use the computer to search for analogies (only 9 of 14 used the computer) and some of those that used the computer did not list all the terms they searched for (i.e. 4 of 9 did not list anything); thus the search strategy for some participants could not be determined. The data for one of the participants who had a difficulty in understanding the meaning of the words on the WordTrees were not included in the analysis of the non-survey measures (i.e. the data was included for only the survey-related measures). This was a result of an observed low knowledge of the English vocabulary from being a native of a non-English speaking country. The decision to include the participant's data in the survey measures was made because the participant had applied the non-automated WordTree Method translating it into his native language; that makes him qualified for the study that was based on his opinion of the WTE program. Another participant whose field of study was not engineering was included in the data because it was not expected to have a significant influence on the participant's performance since the participant had equally been taught the required material needed to participate in the study.

Results and Discussion

The number of analogies identified by the participants is shown in Table 16. For scores by the evaluator, participants in the WTE study found significantly ($p<0.1$) more analogies on average than the control group in the Linsey et al., 2008 study; a pairwise t-test was used for the analysis ($t=0.46$, $p=0.001$).

Table 16: Number of analogies as scored by the participants and the evaluator

	Evaluator Scores (SD.)	**Raw Participant Scores (S.D.)**	**N**
Ave. Control (Linsey et al., 2008 study)	7.6 (4.8)	7.6 (4.8)	10
Ave. WordTree (Linsey et al., 2008 study)	23.3 (12.2)	15.6 (13.2)	10
Ave. WordTree (WTE)	29.4 (15.9)	15.6 (9.1)	14

Table 17 shows the percentage of identified analogies that were used to find solutions in both the Linsey et al., 2008 study and the current one. The average percentage of identified analogies that were used to find solutions was 42 percent for the Linsey et al., 2008 study and 22 percent for the WTE study. the decline was not statistically significant from a pairwise t-test The non-significance could be attributed to the large deviations from the mean as shown in the results for the minimum and maximum usage in the Linsey et al., 2008 study (15% and 64% respectively) and the WTE study (5% and 78% respectively). For the WTE study, the wide deviation from the average by the values of maximum and minimum percentage of analogies used for solutions could suggest that there were some participant who fully understood how to apply the identified analogies and others who did not know how to. A hypothesis from observing that some of the graduate students were non-English native speakers was that it had an influence on how well they interpreted each word on their WordTrees to find analogies resulting in poor performance.

Table 17: The percent of identified analogies used to find solutions

	Percentage of identified analogies that were used to find solutions (Linsey et al., 2008 study)	Percentage of identified analogies that were used to find solutions (WTE)
Ave. Usage	42%	22%
Min. Usage	15%	5%
Max Usage	64%	78%

Database search

In the experiment using the WordTree Express program only nine of the fourteen participants chose to use the internet to search for solutions. Four of the participants that used the internet did not present any results from their search and therefore did not bother to record the terms they searched for. This resulted in only five participant left to analyze search patterns. From the results shown in Table 18, all the five participants searched only within the peanut domain:

Table 18: Number of participants who searched outside the domain of peanut shelling

	Outside Peanut Shelling Domain	**Only Within Peanut Shelling Domain**
Control	0	4
WordTree (Linsey et al., 2008 study)	6	2
WordTree (WTE)	0	5

The result could be attributed to several possible factors such as: 1. the WordTree Express tool may be have had an effect causing the participants to fixate on the peanut shelling domain. 2. The part of the WordTree Method that stresses the importance of searching outside the principal domain during the research step of the WordTree Method may not have been understood clearly by the participants when they were taught the method. This was indicated by some the comments from in the interview questions presented to the participants. For example one of the participants answered to the interview question:

Question: How could the WordTree Method be improved?

Answer: "How do we draw the line between what is a design analogous domain and what is not? More clarification is needed. Also what good will the domains do for the engineer?"

From the collected data, nine participants used the internet to search for solution, so it is also possible that some of those that did not record their search terms may have searched for terms outside the peanut shelling domain.

Surveys

Figure 23 shows the result from comparing the Linsey et al., 2008 study, pre-experiment and post-experiment participant opinions on the usefulness of each step in the WordTree Method. As expected, the results from the Linsey et al., 2008 study and the pre-experiment survey from the WTE study are similar expect for "Listing Analogous Domains". In addition, the results show a favorable increase across the four questions asked, but only two of them were statistically significant between the pre-experiment and post-experiment for the WTE study: WordTrees (t= -2.10, p=0.05) and listing analogous domains (t= -3.1, p=0.007) (p<0.1 is significant). The results suggest that the participants were finding more value in the WordTree Method than they originally had prior to using the WordTree Express tool. The results show that for the question on listing analogous domains, participants opinions increased significantly between the pre-experiment and post experiment scores. This observation could suggest that the participants found some useful analogies from their WordTrees that were outside the peanut shelling domain; this was determined from the analogies they listed and used for solutions.

Given that the participants had low pre-experiment opinions about listing analogous domains, a hypothesis is that the participants didn’t fully value the importance of analogous domains at the time they were taught the WordTree Method possible because they did not learn it well, but saw the importance as they applied the method for the design problem in the experiment.

Figure 24 shows the results of the participant surveys on the value of different methods for a typical engineering design problem. The change in the pre-experiment and post-experiment scores across all the methods was insignificant except for the WordTree Method. Between the non-automated WordTree Method and the WordTree Method using WTE where a t-test showed statistical significance (t= -1.9, p= 0.07) for an increased opinion for the Method. This result suggests that the opinions of the participants on the value of the WordTree Method changed in a positive way from using the WordTree Express program.

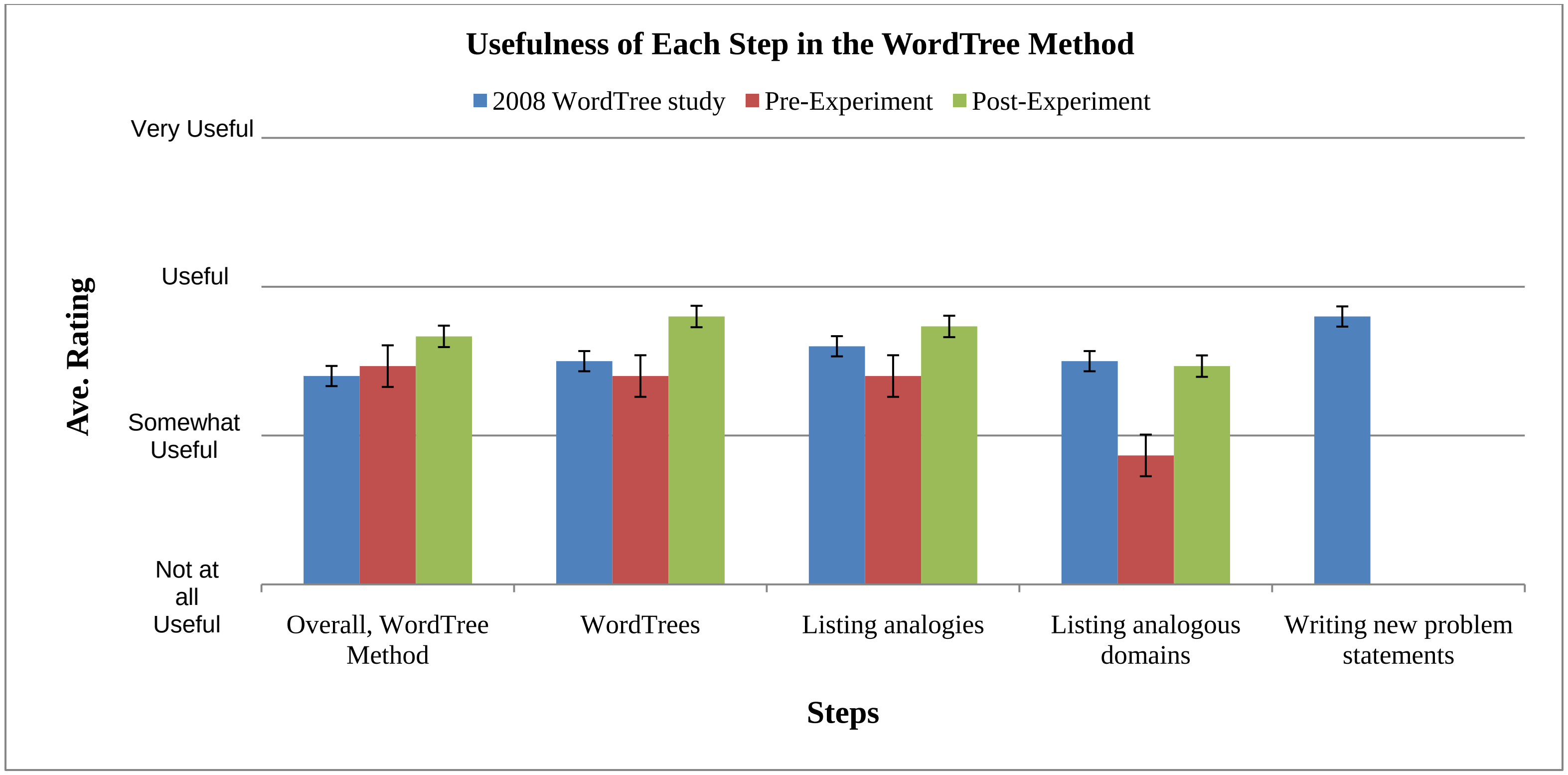


Figure 23: Usefulness of each step in the WordTree Method with standard error bars

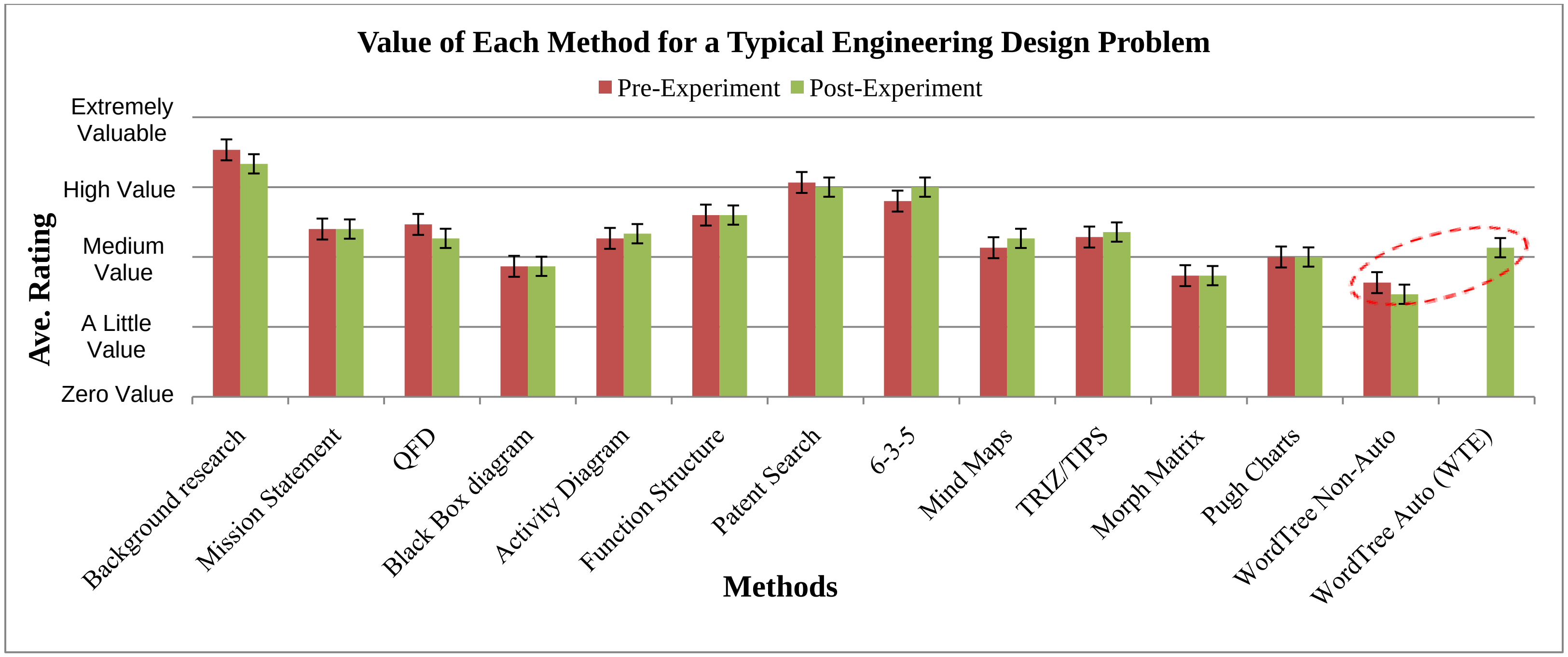


Figure 24: Participants were asked how valuable each method was for a typical engineering design problem

Figure 25 shows the results from the participant surveys on the value of different methods for a design problem that requires an innovative solution. The change in the pre-experiment and post-experiment scores across all the methods was statistically insignificant (for significance $p<0.1$) except for the QFD, function structure and 6-3-5 method ($t=-1.9$, $p=0.08$; $t=1.9$, $p=0.08$; and $t=2.4$, $p=0.03$, respectively). This result was strange because an effect was not expected for the either of the methods. A t-test comparing the post-experiment score for the non-automated WordTree Method and the WordTree Method with the WTE tool showed the difference to be statistically significant ($t=-3.6$, $p=0.003$); this result suggests that using the WTE tool caused the gap in participants' opinions to increase between the values for the WTE tool and the non-automated WordTree Method during the experiment.

The significance found in the 6-3-5 and Pugh methods between the Linsey et al., 2008 study and the current study could have resulted because the undergraduate students, on average, did not understand the method purpose compared to the graduate students. The methods in the Linsey et al., 2008 study and the current study were all taught by different professors with different except for the WordTree Method which could account for some of the differences.

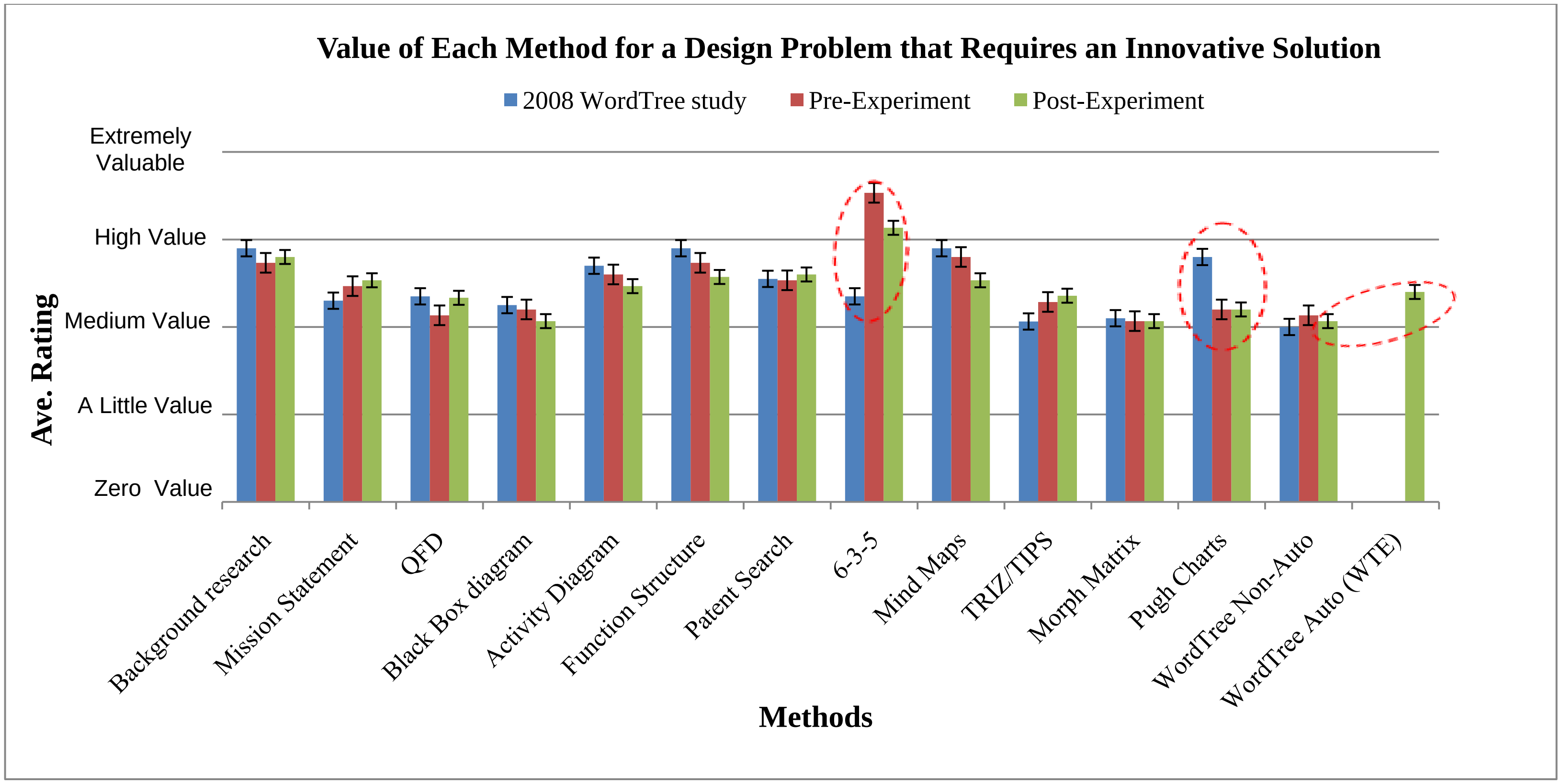


Figure 25: Participants were asked how valuable each method was for a design problem that required an innovative solution

Figure 26 shows the results from the participant surveys asking them how likely they were to use each method in the future. A t-test for change in the pre-experiment and post-experiment scores was significant ($p<0.1$) for the black box diagram and patent search ($t=-1.9$, $p=0.08$; $t=2.6$, $p=0.02$ respectively). The change in the WordTree Method (non-automated) and the WordTree Method (using the WTE) were also found to be statistically significant ($p<0.1$) from the t-test ($t=-1.9$, $p=0.08$; $t=-4.2$, $p=0.001$ respectively). The change found in the use of the black box was not expected, but a change in patent search could have resulted from being applied in the WordTree Method (as a step). The change found between the pre-experiment and post-experiment for the non-automated WordTree Method suggests that the participants are more willing to apply the method even if it was done manually in the future. A possible reason for this result is that using the WordTree Express tool may have caused some influence on their perception or understanding of the WordTree Method so that they are willing to use the non-automated method in the future.

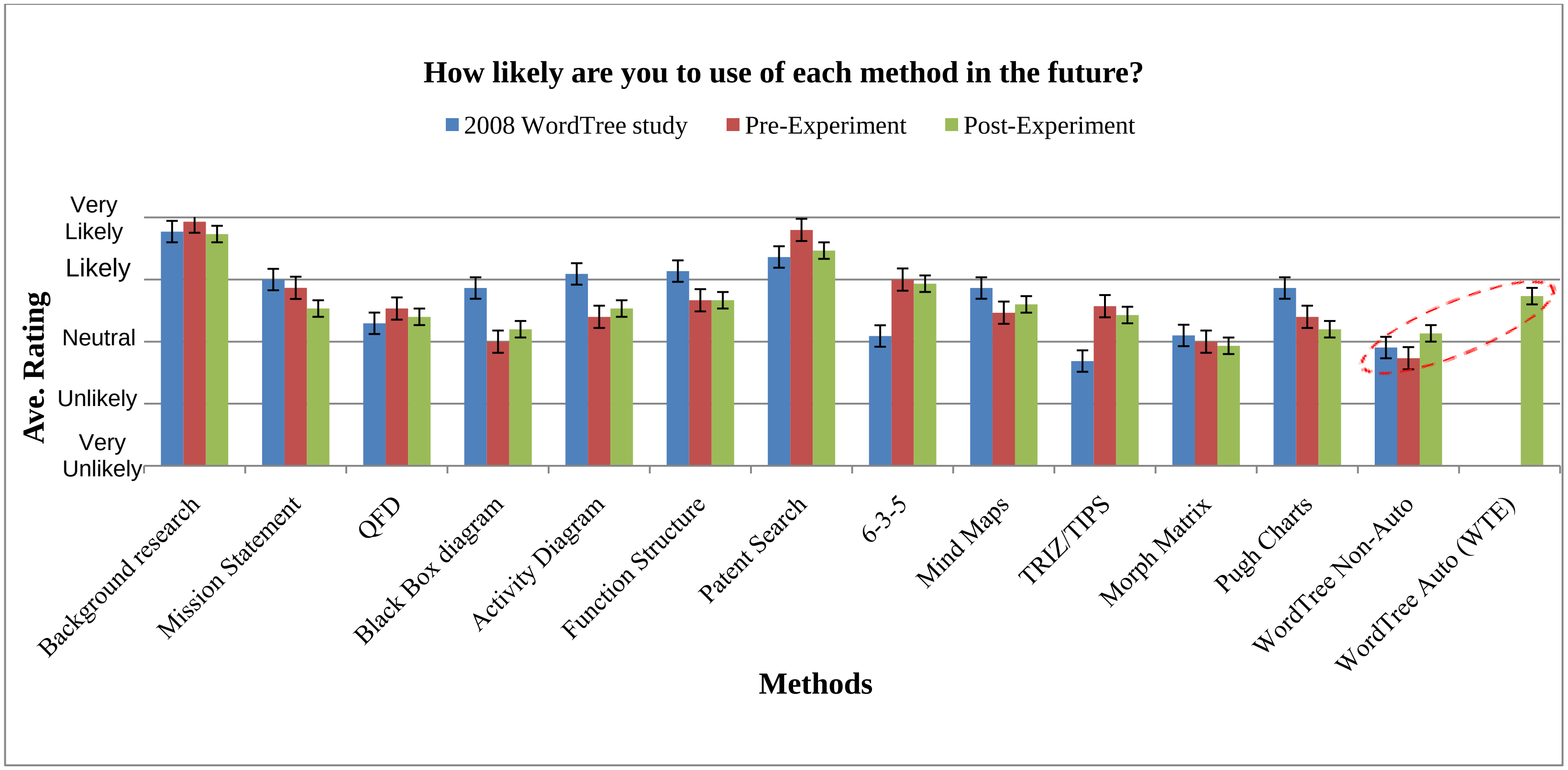
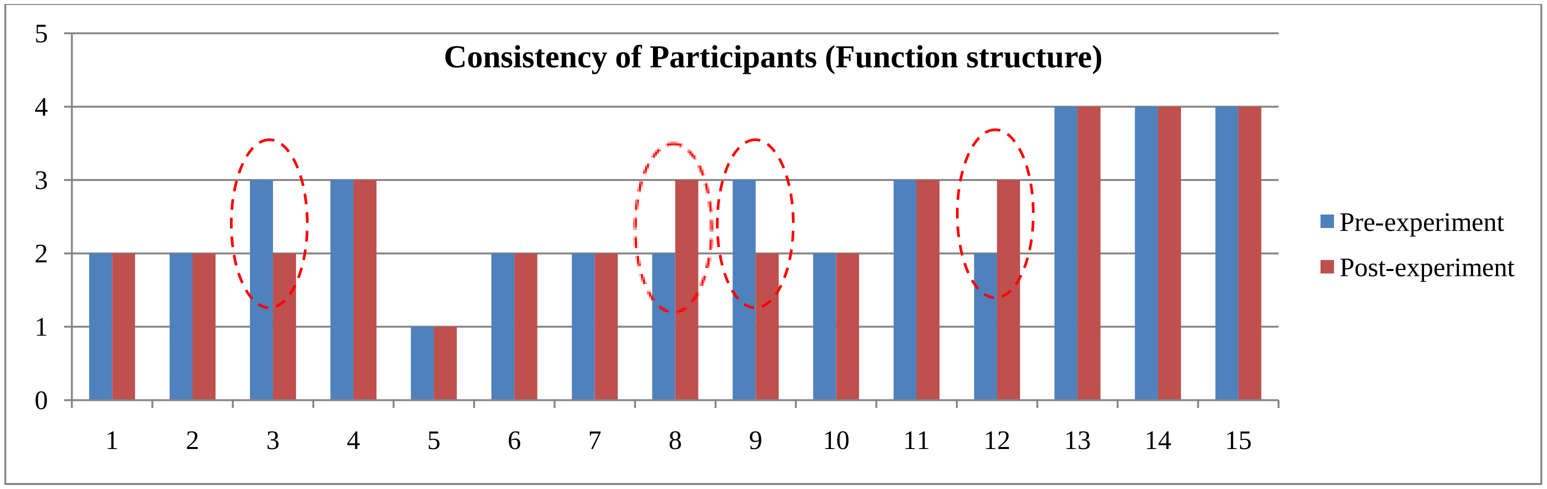


Figure 26: Participants were asked how likely they were to use each method in the future

In comparing the Linsey et al., 2008 study with the WTE study, there were statistical differences in the box diagram, activity diagrams, function structure, 6-3-5 and TIPS/TRIZ. These results could be attributed to the graduate versus undergraduate discrepancy in understanding each method's value or in how the methods were taught.

Evaluating the Participants' Scoring Consistency

The charts in Figure 27 show the consistency of the participants, for pre and post-experiment, in answering selected questions that were not expected to be influenced by the experiment. The questions were on the participants' opinions on value for a typical engineering design problem for function structures and QFD's respectively. These results are typical for the various methods. The charts show (from the dashed ovals) that some participants made some changes in scoring methods that were not targeted by the WTE tool for influence. This could signify that some of the changes that exist between the pre and post experiment scores for the WordTree Method may not be entirely a result of the WordTree Express tool's effect, but the participants' rating inconsistencies.



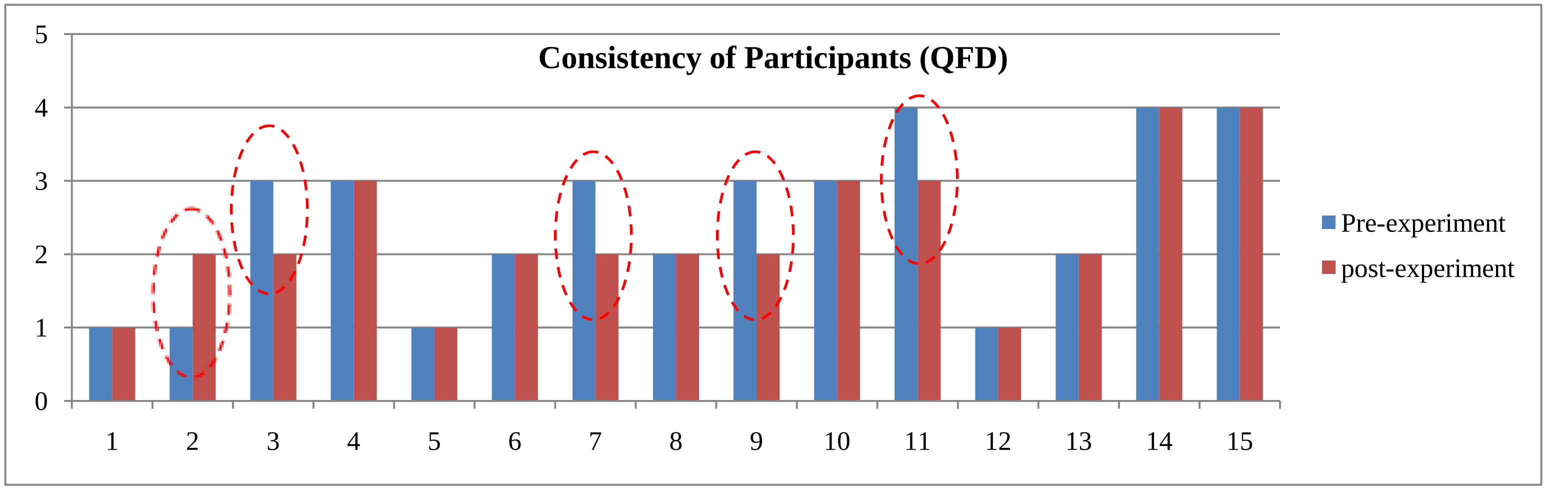


Figure 27: Two randomly selected charts to illustrate participant answering consistency

Evalation of the participants' scoring for the WordTree Method

Figure s 28, 29 and 30 show the pre-experiment and post-experiment score of each participant for the WordTree Method only. The results from observing how each participant scored the WordTree Method showed a trend in the post-experiment survey of opinions to be either equal or improved for every participant except for participant 1 and 8 in the question for typical engineering problems and 8 and 10 for the question on innovative designs. Further review of the participant's data show that participant 1 and 8 were among the participants that identified the most analogies, but used very few of them for solutions. Participant 10 was among those that identified the least number of analogies resulting in only a few solutions. This would possibly suggest that an element of frustration for the given design problem may have influenced their opinions about the WordTree Method. A review of each of the participants' data on the evaluation of the WordTree Method was done to check for consistency in how they responded to similar questions in the surveys. The results show that the participants were consistent in every question within a margin of error of +/- 1. This would mean that the effect of their opinions were not mostly out of inconsistency in scoring.

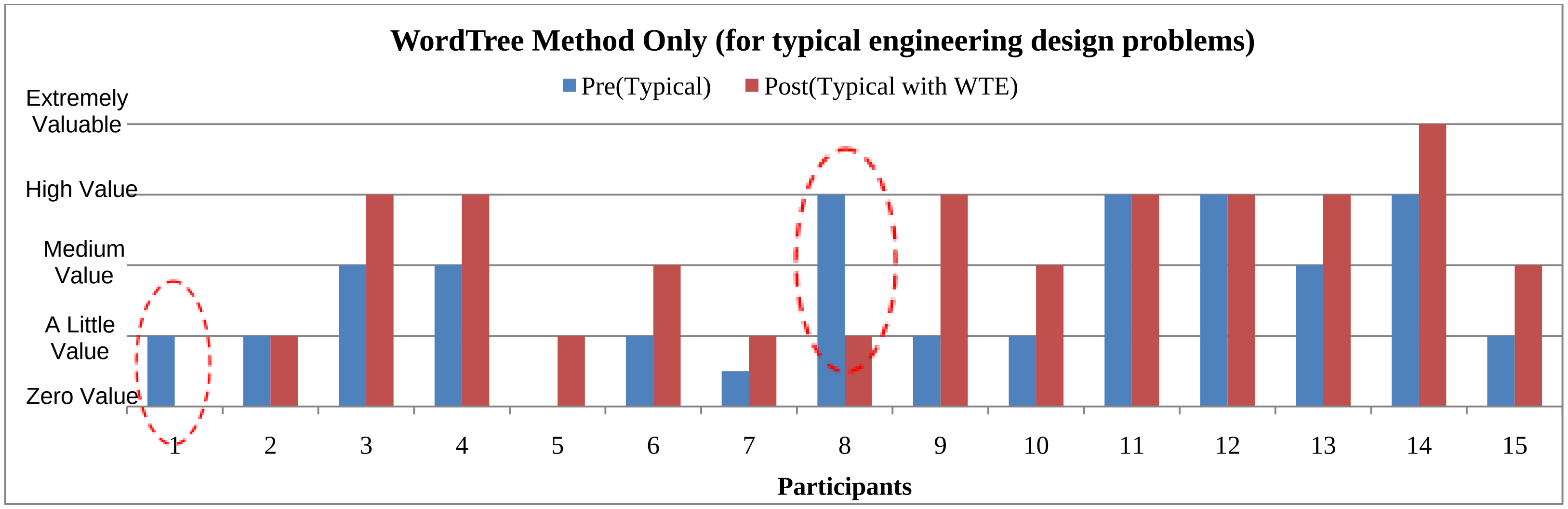


Figure 28: Participants' WordTree Method value score for a typical engineering design problem

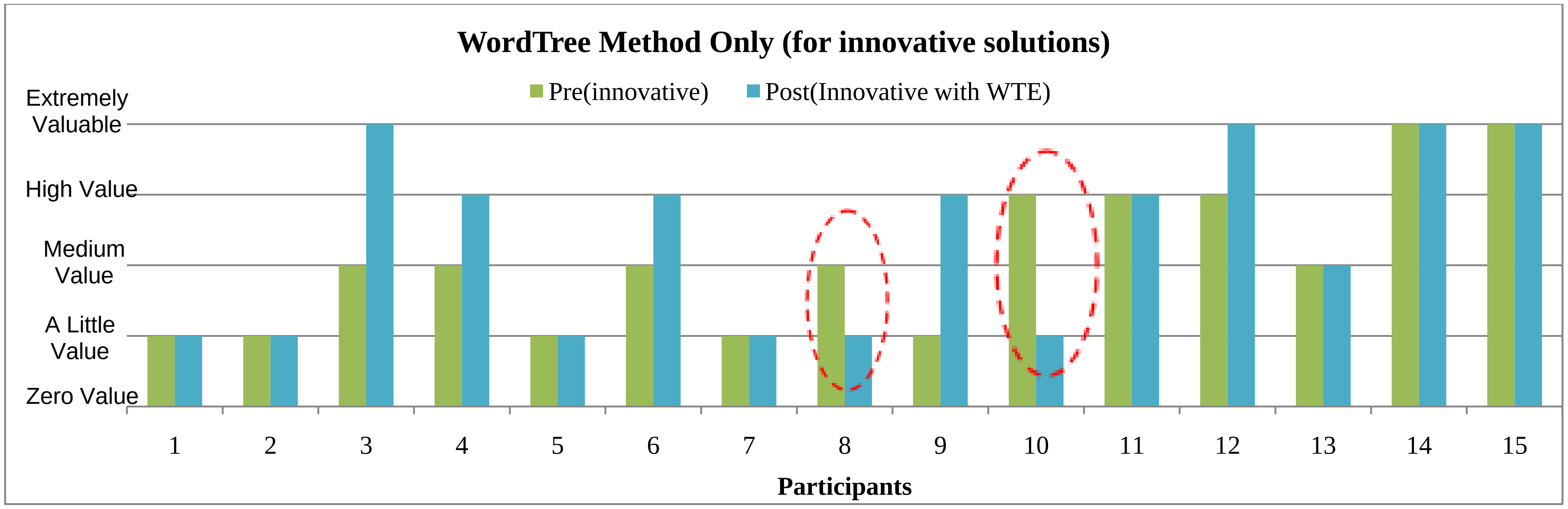


Figure 29: Participants' WordTree Method value score for a design that required an innovative solution

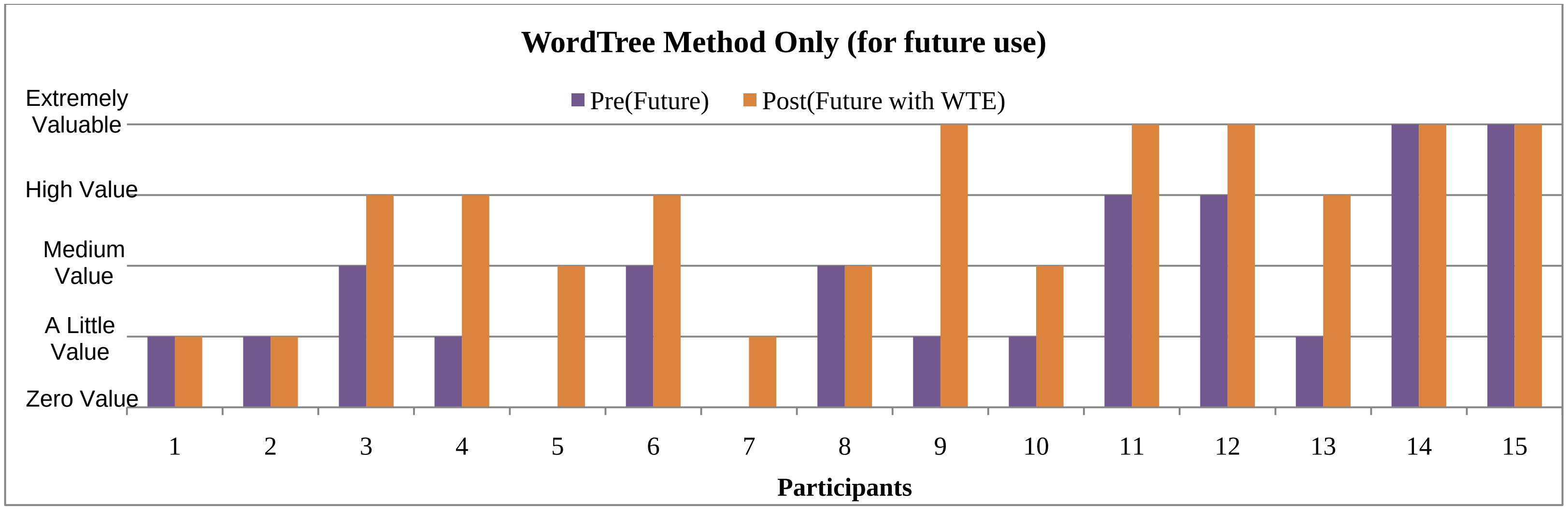


Figure 30: Participants' WordTree Method value score for how likely they would use it in the future

Evaluation of the WordTree Method

Figure 31 shows the results of the questionnaire (Table 19) for evaluating the WordTree Method and compares results from the Linsey et al., 2008 study, pre-experiment and post-experiment WTE study. The results comparing the pre-experiment and post-experiment showed statistically significant (p-value <0.1) positive effect in all the questions asked except for questions 7, 8 and11 where they remained statistically equal. The trend was higher scores for the post experiment (t and p-values are shown in Table 20). This effect showed that the WordTree Express program influenced the users' opinions in a positive way. Although, question 7 and 9 were expected to increase based on the questions comparing the different methods, the insignificance could be attributed to the question scale (0 to 4 in this one) rather than (0 to 5 in the methods value questions); this would influence an increase in standard error.

In comparing the result of the post-experiment with the Linsey et al., 2008 study, the result shows a positive change for all the questions with all statistically significant except for questions 7, 8 and11. The results suggest a possible effect from using the WordTree Express tool was positive.

In comparing the results of the Linsey et al., 2008 study to the pre experiment survey results, there were significant differences found in questions 1, 2, 5, 6, 10 and 11. These differences could be from a number of factors such as: graduate (WTE study) vs. undergraduate (Linsey et al., 2008 study), survey given before the experiment (WTE study) vs. after (Linsey et al., 2008 study), etc.

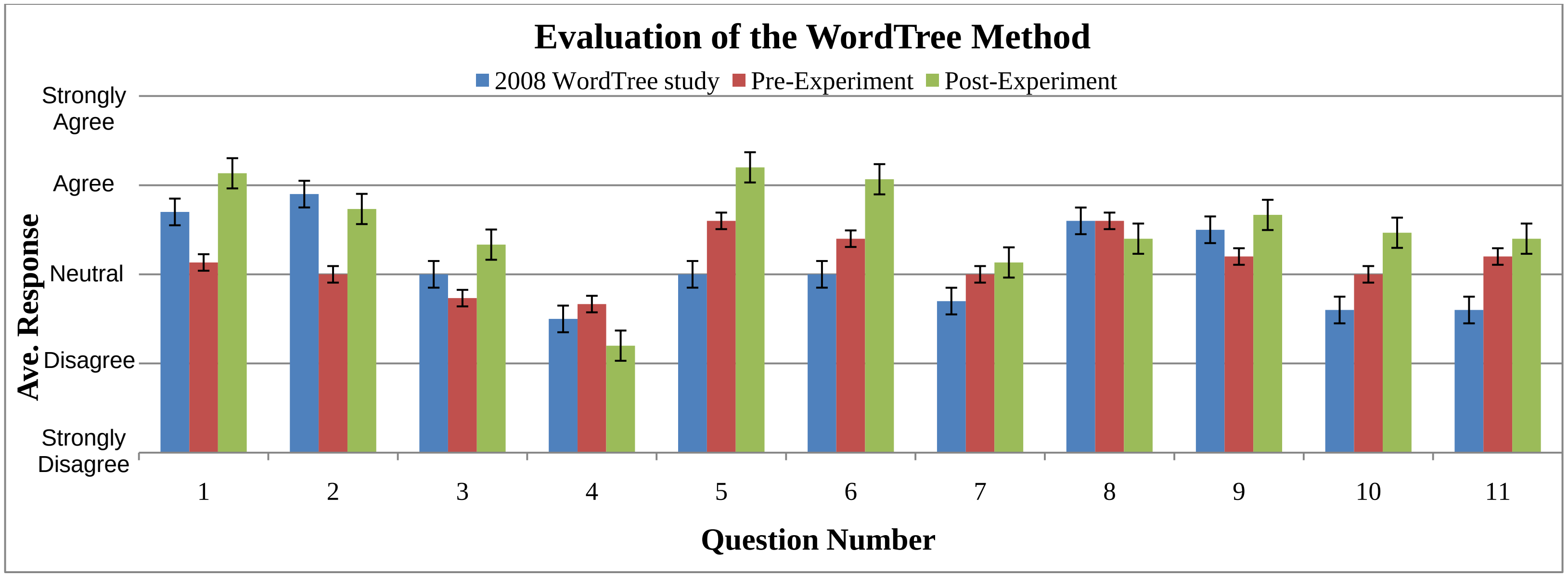


Figure 31: Participants evaluation of the WordTree Method

Table 19: Questions asked to evaluate the WordTree Method

1. This method helped me to find analogies for my design problem.	7. I expect to use this method in the future.
2. This method helped me to generate more ideas.	8. This method (does not) needs improvements.*
3. This method helped me to generate more quality ideas	9. This method was useful.
4. This method was (not) a waste of my time.*	10. I like using the method.
5. The presentation of this method was easy to understand.	11. I expect to use this method in the future for design problems that require an innovative solution.
6. This method was easy to use.	*Reversely scored. ( ) omitted in actual survey

Table 20: p-values comparing pre and post experiment questions

Questions	t	df	Sig. (2-tailed)
1	-3.9	14	.002
2	-2.8	14	.016
3	-3.2	14	.007
4	1.8	14	.089
5	-2.07	14	.057
6	-2.5	14	.027
7	-1.00	14	.33
8	1.00	14	.33
9	-2.8	14	.014
10	-2.8	14	.014
11	-1.0	14	.33

Addressing the Research Questions

Question 1: Does the WordTree Express program affect engineering designers' opinions of the WordTree Method? The WordTree Express program positively affected the opinions of the designers. The study showed a significant rise in value scores for the WordTree Method in two of the three measures taken: the question on the value of the WordTree Method for a typical engineering design problem and for how likely they would use the Method in the future The results shown in Figures 24 and 26 support the hypothesis that: WTE, by simplifying the process of generating WordNet-based WordTrees, will increase designers' opinions of the WordTree Method. Although the positive results seem to point at the use of the WordTree Express tool, a second factor to consider is that the participants knew at the time of the post experiment survey what they were being tested for and it may have biased their response.

Question 2: Does providing more comprehensive WordTrees (via WordTree Express) for the chosen problem descriptors affect the number of analogies identified and used for conceptual solutions to the design problem? The results from Table 16 showed that there was no significant increase in the number of analogies identified by the WordTree participants in both studies. This shows that the second hypothesis stating that using the WordTree Express program to create WordTrees will present the user with a more comprehensive WordTree; increasing the number of identified analogies was not satisfied. This result lead to the possibility that the pre-generated WordTrees given to the participants in the prior study was well put together using mostly relevant words to create them. Another possibility is that since the prior study had smaller WordTrees, it was easier to for the participants to identify the relevant potential analogies, while for the WTE participants a more demanding filtering was required from the participants because of the larger size of the WordTrees and this could have led to overlooked potential analogies. The percentage of analogies used for conceptual solutions on average were equal because there was no statistical significance (p-= 0.5). A possible explanation for the decrease in the percentage of analogies used is that the participants were graduate students rather than undergrads (as in the Linsey et al., 2008 study) which leads to the assumption that graduate students would tend to be more selective of the concepts they chose to present compared to the undergrads.

Question 3: What are some of the additional avenues for improvement to the WordTree Design-by-Analogy Method? From the observation of the participants during the experiment, the WordTrees needs to be further refined as many of the participants found it tedious to scroll through very large WordTrees. For example one participant answered to the following interview survey:

Survey: Please state any additional comments you have about the experiment. Use back of paper if needed.

Participant response: “Saving WordTree file is troublesome.” “WordTree generated tends to be horizontal, not easy to read.” “Easy to use”

A possible solution would be to prune the WordTrees using predefined criterion and storing the WordTrees in a depository for multiple uses.

Another area for improvement would be in enhancing the participants understanding of the Method. While some of the participants found the WordTree Method to be very useful, others did not seem to understand the concept of analogies and the need to search distant analogous domains for possible innovative solutions. This could be from not being presented with very effective examples of solutions using the WordTree Method during the lecture on the method. It is recommended to stress the method’s strength by challenging the students to generate ideas for a selected design problem in a class activity and presenting the students with the solution found using the WordTree Method. The lack of a full understanding of applying the WordTree Method could have suppressed the level of the positive results shown. Participants with the less favorable opinions came from those that identified either many or a few number of analogies; this supports the assertion that a lack of understanding of the method or a lack of skills in Design-by-Analogy will likely produce unfavorable results in the experiment.

CHAPTER V

CONCLUSION AND FUTURE WORK

Design-by-Analogy is becoming a more sought after approach for solving engineering design problems. Some of the best solutions to design problems are found in nature and prior solutions. The WordTree Method not only presents a way to lead an engineer or designer to useful analogies in nature, but also to other existing and useful non-natural analogies. This thesis has investigated the WordTree Method and has sought to foster advancing the state of the method to a more easily adapted method by engineers and designers. The first step to achieving this was to change designers' opinion about the method to a more positive one.

A computational tool for simplifying the process of generating WordNet-based WordTree was developed and has shown, by experiment, to be effective in significantly changing the opinions of engineers about the WordTree Method. The result is expected to help in the goal of making the WordTree Method a more sought after one. The results from the experiment showed that the WordTree Express tool allowed the users to identify a large number of analogies for the given design problem. It also showed that the participants' opinions on the WordTree Method positively changed across most of the survey questions asked. The result from the participants' opinion on using the WordTree Method in the future for each participant increased or remained the same; this was the basis of the study. The study showed that developing the WTE tool to foster the application of the WordTree Method made a positive impact that could be a contribution to the way the students are taught the method in the future.

This thesis has also done a thorough investigation on how the WordTree Method could benefit from other existing ontologies such as PHYSSYS, HowNet and YMIR ontologies. The PHYSSYS ontology provides a broad space for engineering specific analogies, but the current library (OLMECO) created using the PHYSSYS ontology limits the ability to search for analogies in nature. However, since the PHYSSYS ontology with the OLMECO library is more focused on engineering than WordNet, it would make finding relevant analogous products quicker. The HowNet ontology has an advantage over the WordNet ontology as its members (words) are clearly defined. For example in HowNet 手术刀 which is Chinese for “scalpel” (surgical knife) contains not just characters, but ideas. 手术 means “surgery” and 刀 means “knife” (Veale, 2005). This means that in HowNet for 手术刀 “surgery knife” you can relate to ideas in the domain of “medicine” (where “surgery” is found) and in the domain of “knifes” (where “knife” is found), but in WordNet “scalpel” would be found in the domain of “medicine” only. So a HowNet user can be provided with analogies in the “surgery domain” and “knives domain” rather than just the “medical tools” domain in WordNet. HowNet contains sufficient structure to realistically support both a taxonomic abstraction view and a structure-mapping view of analogy generation (Veale, 2005). In other words, using the example 手术刀, ”sufficient structure” means the unit (手术刀) is enough, as a word, to be classified by a taxonomy. The YMIR ontology which is made up of taxonomy of concepts that are used in different domains of engineering could be presented in a WordTree form for designers and engineers to use as an analogy search domain.

Contributions of This Thesis

1. Research work done in this thesis supported the development of the WordTree Express tool to foster the application of the WordTree design method. The WTE tool can serve as a tool to teach the WordTree Method.
2. The WTE tool has shown, from the experiment performed, that it had a positive effect on the opinions of engineers on the WordTree Method.
3. This thesis has researched some useful ontologies for their application in Design-by-Analogy such as the HowNet, YMIR and PYSSYS ontologies.

Future Work

For future work on the WordTree Method, some recommendations are proposed from the collection of insights gained from this thesis. The background research coupled with the results of the experiment and other significant observations lead to the following recommendations:

- Increase sample size experiment: A larger size experiment with more participants and a longer duration time would make for a better study. It was noticed that a large number of the participants did not have enough time to go through their second WordTree while identifying potential analogies. This could have had an effect on the number of useful analogies they identified. It is recommended that the experiment last for at least a three hour period and include a semi-formal interview of the participants to accurately account for their understanding and reasoning behind their opinions about the WordTree Method.

- Alternate ontology to WordNet: WordNet has proven to be very effective in leading to useful analogies, but as noted by Veale WordNet lacks the word transparency that could lead to a broader analogy space. This transparency is defined by the structure of a word (e.g. “surgical knife” in HowNet is represented by “scalpel” in WordNet) that makes more connections than the WordNet representation. In other words, a HowNet user can be provided with analogies in the “surgery domain” and “knives domain” rather than just the “medical tools” domain in WordNet.   HowNet contains sufficient structure to realistically support both a taxonomic abstraction view and a structure-mapping view of analogy generation (Veale, 2005). In other words, the words and pairs of words (i.e. Chinese language-based structure) you find in HowNet have enough structure to allow them to be classified by the HowNet ontology. So a recommendation would be to use HowNet to generate WordTrees and compare its effectiveness with WordNet results. The two results could also be combined for an even larger design space for analogy search.
- Improve user-interface: Including a more sophisticated user interface with more functions would make the WordTree Method even easier to apply. Functions such as those found in Visual Thesaurus where you can find definitions of the word by simply pointing to them on the tree rather than doing a new word search would be ideal. The software used for the development of Visual Thesaurus, thinkmap (Thinkmap, 2010a) should be considered as a developing tool for the new interface because it was designed specifically for the kind of application the

WordTree is (i.e. an application that displays large trees as outputs). The Thinkmap software will be beneficial because 1. it comes with a set of out-of-the-box configurations for solving common visualization problems, as well as visualization techniques for customizing data displays. 2. Visualizations can be built rapidly using an XML-based configuration language. 3. It comes with pre-configured building blocks including: Spider, Hierarchy, Clustering, and Chronology.

- combinFormation: As discussed in the background section, the work of Glenberg and Langston showed that when images are accompanied by descriptive texts they promote the formation of mental models than just texts alone (Glenberg & Langston, 1992). A recommendation would be to integrate the current WordTree Method with a program such as combinFormation to make identifying analogies easier for the user and to present images with the words.
- Color coding words on the WordTrees: Highlighting relationships between words on a WordTree could also be effective in teaching the importance of domains and how solutions can be found in distant domains. When it is visually clear to the student that two words belong to distinctly different domains it promotes a faster understanding of importance to search in other domains for solutions and would make students easily learn how to properly apply the method. Using different color for words that belong to different domains is a suggested approach.
- General use repositories: Useful analogies have made their way into design solutions from analogous products built on fundamental engineering principles

and concepts. The YMIR ontology is a taxonomy of concepts used in different engineering disciplines and could be used as a source of analogy identification if properly presented to designers and engineers. A proposed direction would be to design a repository of solutions, from using the WordTree Method and other design methods that could retrieve solutions from identifying function keywords (verb).

REFERENCES

Alberts, L., & Dikker, F. (1992). Integrating standards and synthesis knowledge using the YMIR ontology. In Artificial Intelligence in Design '94, J.S. Gero and F. Sudweeks (Eds.), Boston, Kluwer Academic Publishers, 517-53494.

Altshuller, G. (1999). The Innovation Algorithm: TRIZ, Systematic Innovation and Technical Creativity, Worchester, MA. Technical Innovation Center.

Borst, W., (1997). *Construction of engineering ontologies for knowledge sharing and reuse.* PhD Dissertation. University of Twente, Enschede, The Netherlands.

Burke, M. (1999). *Organization of Multimedia Resources,* Hampshire, UK: Gower.

Cross, V., & Bathija, V. (2009). Automatic ontology creation using adaptation. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing 24(01),* 127-141.

Ellson, J., Gansner, E., Hu, Y., & Bilgin, A. (2010). *Graphviz - Graph Visualization Software*. http://www.graphviz.org.

English, K., Naim, A., Lewis, K., Schmidt, S., Viswanathan, V., Linsey, J., McAdams, D.A., Bishop, B., Campbell, M.I., Poppa, K., Stone, R.B., & Orsborn, S. (2010). Impacting designer creativity through IT-enabled concept generation. *Journal of Computing and Information Science in Engineering 10(3)*, 031007-031010.

Glenberg, A., & Langston, W. (1992). Comprehension of illustrated text: Pictures help to build mental models. *Journal of Memory and Language 31(2)*, 129-151.

Gomes, P., Seco, N., Pereira, F.C., Paiva, P., Carreiro, P., Ferreira, J.L., & Bento, C. (2006). The importance of retrieval in creative design analogies. *Knowledge-based Systems* 19(7), 480-488.

Gruber, T. (1995). Toward principles for the design of ontologies used for knowledge sharing. *International Journal of Human Computer Studies 43(5)*, 907-928.

INKSCAPE. (2010). *Open Source Scalable Vector Graphics Editor*. http://www.inkscape.org.

Kamps, J., & Marx, M. (2002). Visualizing WordNet structure. ICGW02, *1st Int. Conf. on Global WordNet*, pp. 182–186. Mysore, India.

Koh, E., Hill, R., Kerne, A., Dworaczyk, B., Mistrot, J., Choi, H., Smith, S., Graeber, R., Caruso, D., & Webb, A. (2007). combinFormation: A mixed-initiative system for representing collections as compositions of image and text surrogates. JCDL07, *Proc. of 6th ACM/IEEE Joint conf. on Digital Libraries*, pp. 11-20. Chapel Hill, NC.

Linsey, J., Wood, K., & Markman, A. (2008). Modality and representation in analogy. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing 22(2)*, 85-100.

Linsey, J.S. (2007). *Design-by-Analogy and Representation in Innovative Engineering Concept Generation*. Ph.D. The University of Teaxas at Austin.

Princeton University (2010). About WordNet. *WordNet*. Princeton University. http://wordnet.princeton.edu

Schild, K., Herstatt, C., & Lüthje, C. (2004). *How to Use Analogies for Breakthrough Innovations*. Hamburg, Germany: Technical University of Hamburg, 63: 495-508.

Thinkmap (2010a). *The Thinkmap White Paper*. http://www.thinkmap.com.

Thinkmap (2010b). *Visual Thesaurus*. Desktop edition. Retrieved from http://www.visualthesaurus.com.

Veale, T. (2005). Analogy generation with HowNet. IJCAI05, *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence* 1148-1153. Edinburgh, Scotland.

Weaver, W., & Prince, G. (1990). Synectics: Its potential for education. *Phi Delta Kappan, 71(5)*, 378-388.

APPENDIX A

WORDTREE EXPRESS EXPERIMENT

PARTICIPANT MATERIALS

1. WordTree Express tutorial

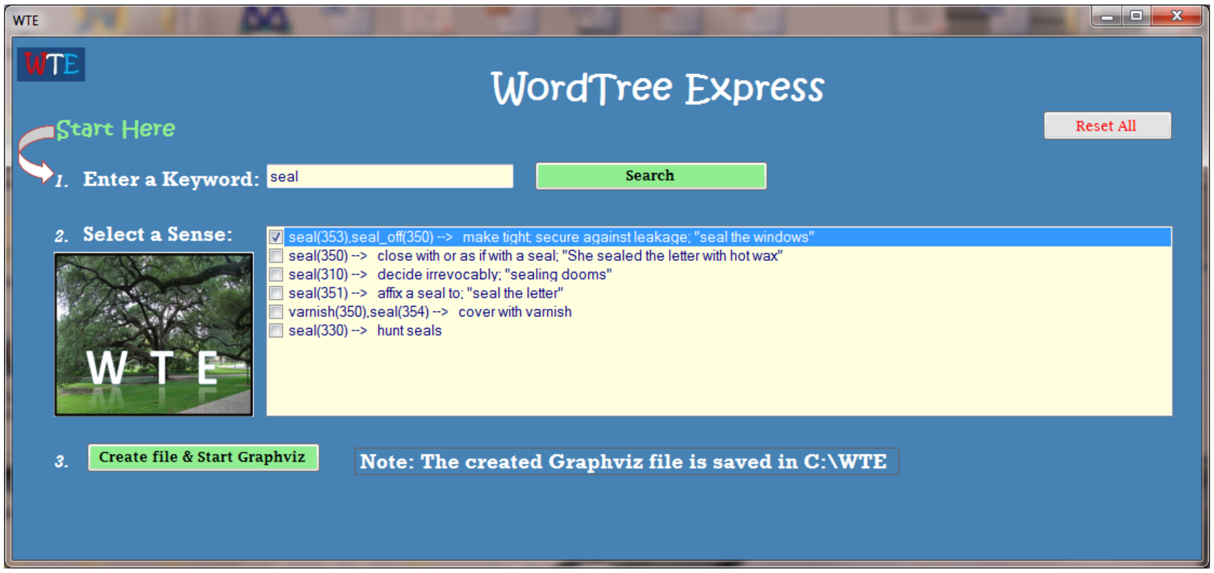
USING WordTree Express

1. Double click the WordTree Express shortcut icon on your desktop.

(WordTree Express program icon as seen on the desktop)

2. Type a keyword in the textbox, click search and select a sense.

   The program should look like this:



seal

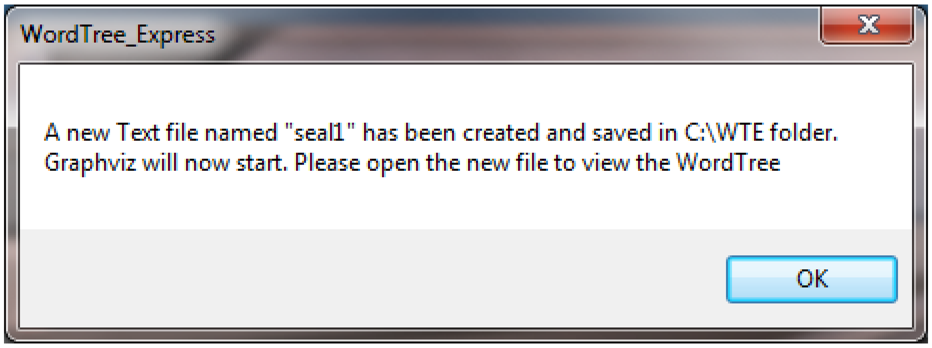
- [✓] seal(353),seal_off(350) --> make tight; secure against leakage; "seal the windows"
- [ ] seal(350) --> close with or as if with a seal; "She sealed the letter with hot wax"
- [ ] seal(310) --> decide irrevocably; "sealing dooms"
- [ ] seal(351) --> affix a seal to; "seal the letter"
- [ ] varnish(350),seal(354) --> cover with varnish
- [ ] seal(330) --> hunt seals

Note: The created Graphviz file is saved in C:\WTE

3. Click on the "**Create file & Start Graphviz**" button to create a Graphviz file and start the Graphviz program. The following popup message will be displayed to

let you know that a file has been saved as the keyword you typed in C:\WTE:

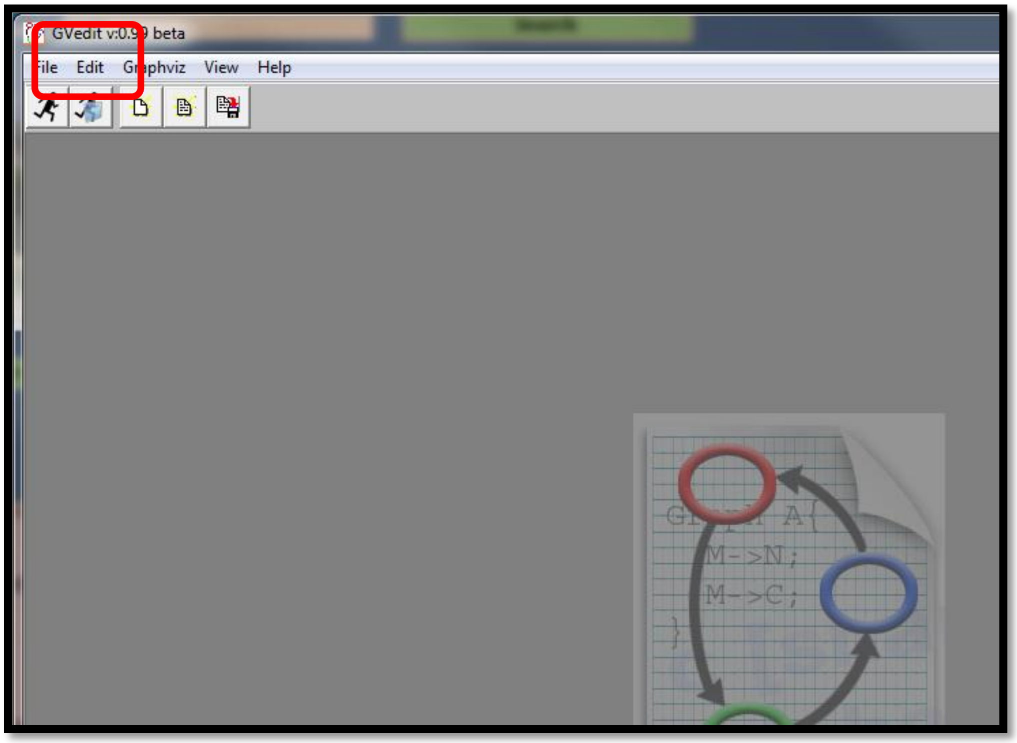
Click on OK

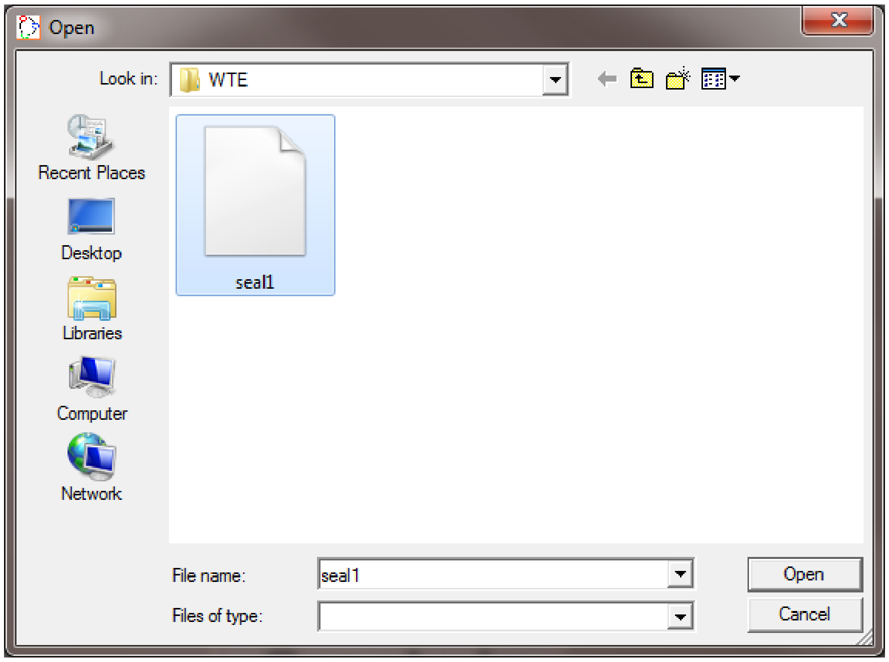


A new Text file named "seal1" has been created and saved in C:\WTE folder.
Graphviz will now start. Please open the new file to view the WordTree

*******If you don't see the message box before Graphviz starts, close Graphviz and click the "Create file & Start Graphviz" button again********

4. In the Graphviz program point to **file-->open** and select the created **Keyword** text file in C:\WTE.

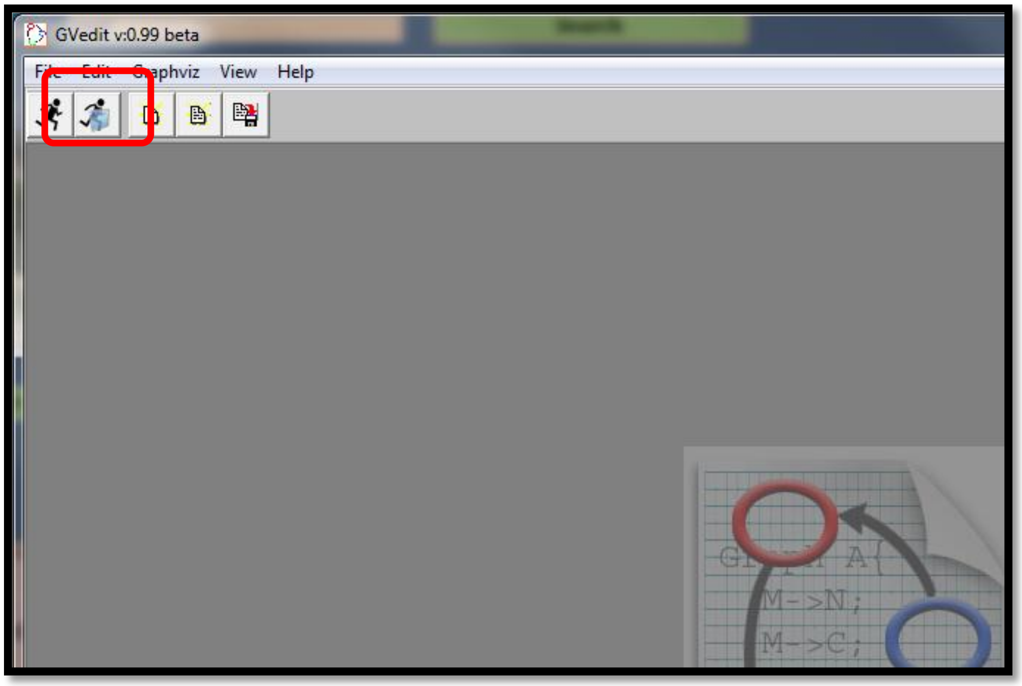


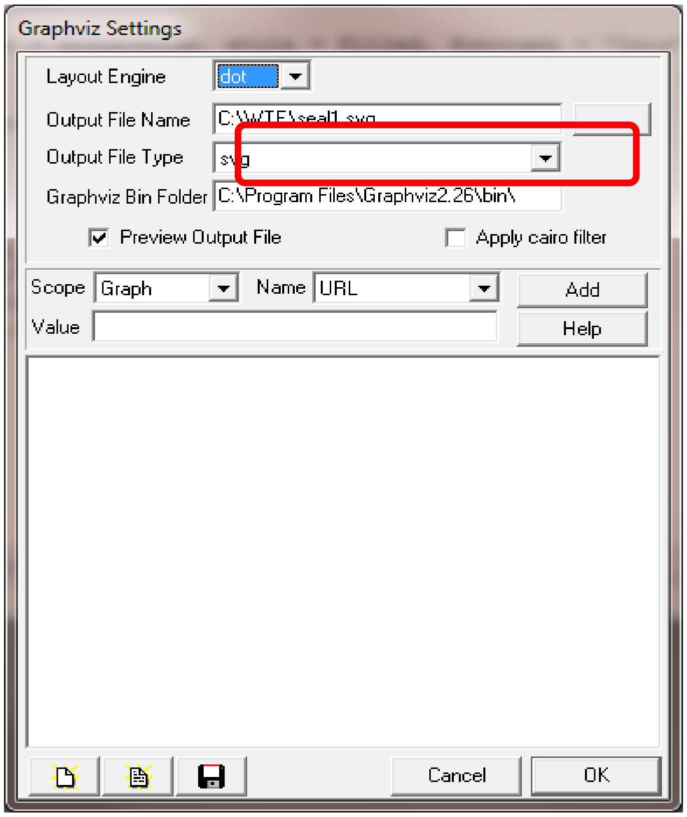


File name: seal1

Files of type:

5. Click on **run** and choose "**.svg**"(Scalable vector graphics) as the "**output file type**" in the popup window.




Graphviz Settings

Layout Engine: dot

Output File Name: C:\WTF\seal1.svg

Output File Type: svg

Graphviz Bin Folder: C:\Program Files\Graphviz2.26\bin\

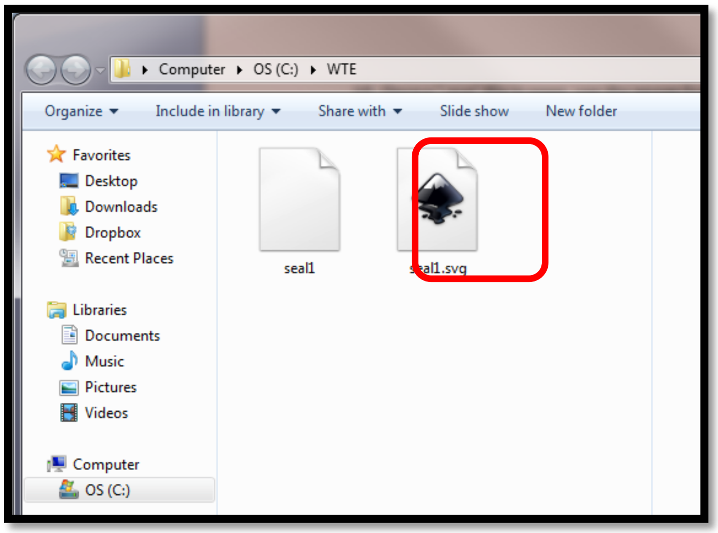
[✓] Preview Output File

[ ] Apply cairo filter

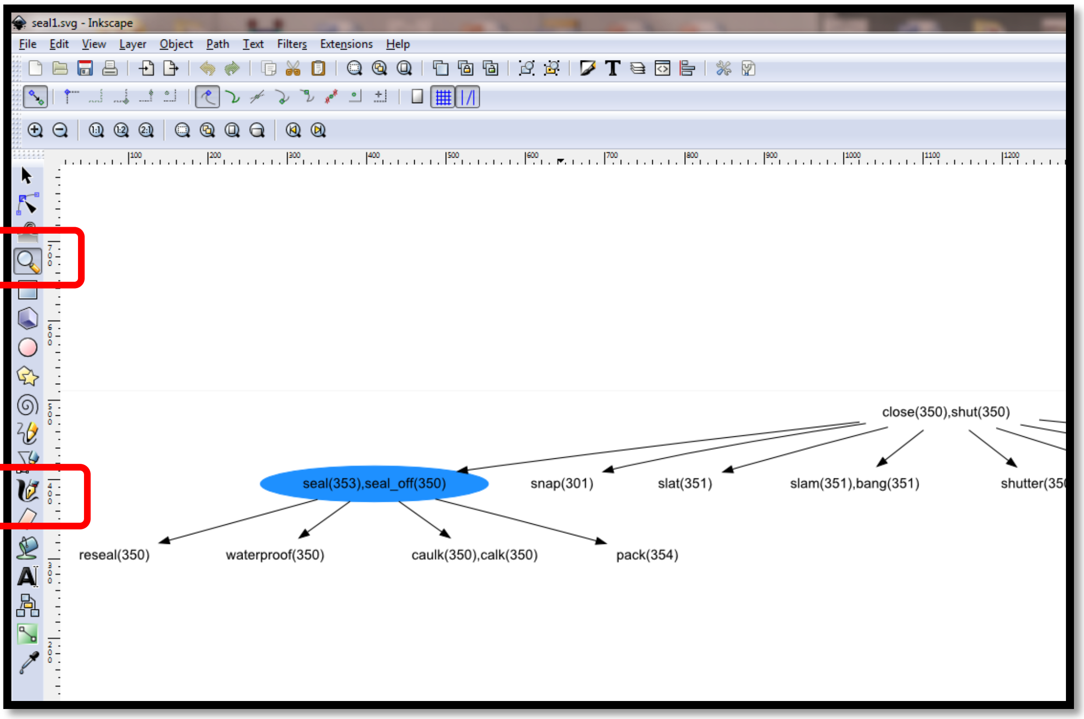
Scope: Graph Name: URL

Value:

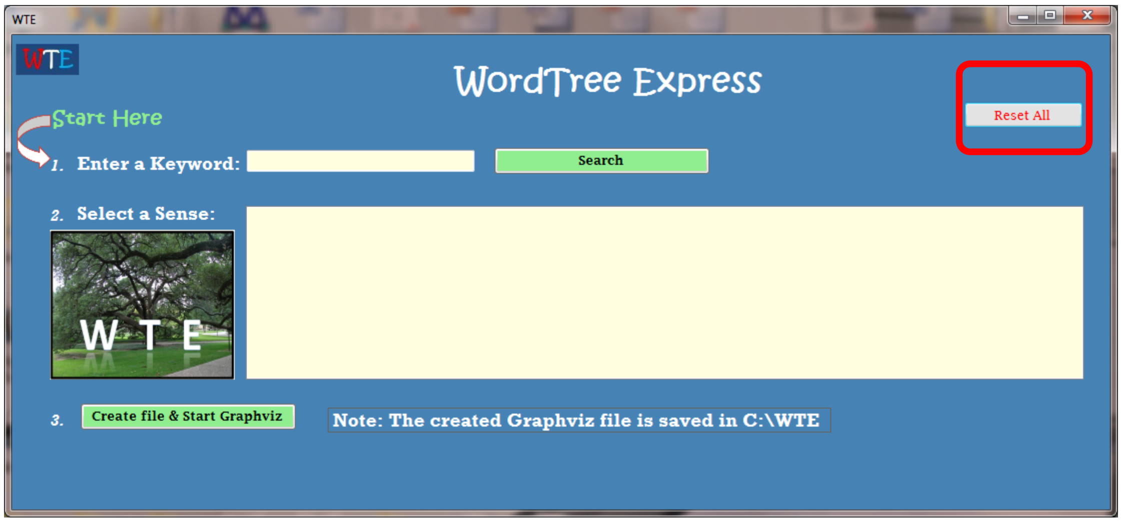
6. Next, **click OK** to save a scalable vector graphics version of the WordTree and to display the graphviz output as shown below.
7. **Close Graphviz** and **Open the WTE shortcut** on your desktop; double click the svg file you created.



8. Use the magnifying glass shown in the figure below to zoom out and in by left clicking with or without holding down the Ctrl button respectively.
9. Use the pen shown in the figure below to write on your WordTree.



10. To perform a new search using WordTree Express, click on the "**Reset All**" button and type a new keyword.



3. ***Design Problem***

Device to shell peanuts

Problem Description:

In places like Haiti and certain West African countries, peanuts are a significant crop. Most peanut farmers shell their peanuts by hand, an inefficient and labor-intensive process. The goal is to build a low-cost, easy to manufacture peanut shelling machine that will increase the productivity of the peanut farmers. The target throughput is approximately 50Kg (110lbs) per hour.

Customer Needs:

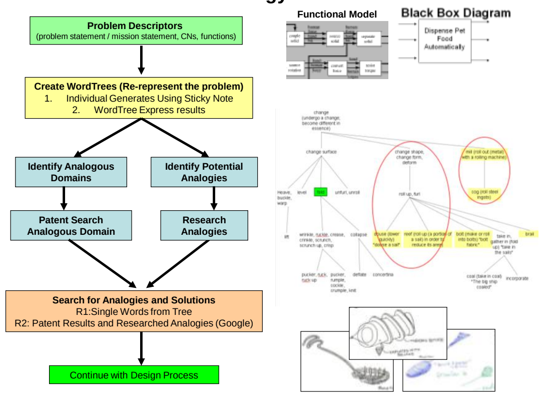
- Must remove the shell with minimal damage to the peanuts.
- Electrical outlets are not available as a power source.
- A large amount of peanut must be quickly shelled.
- Low cost and easy to manufacture.

Functions:

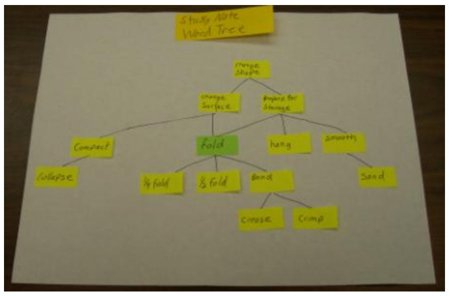
- Import energy to the system.
- Break peanut shell.
- Separate peanut shell from the nut.

4. WordTree Method Reminder

Modified WordTree Analogy Method Overview

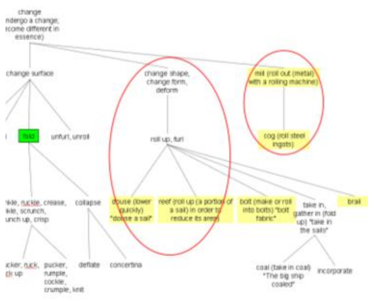

Problem Descriptors
(problem statement / mission statement, CNs, functions)
Create WordTrees (Re-represent the problem)
1. Individual Generates Using Sticky Note
2. WordTree Express results
Identify Analogous Domains
Identify Potential Analogies
Patent Search Analogous Domain
Research Analogies
Search for Analogies and Solutions
R1:Single Words from Tree
R2: Patent Results and Researched Analogies (Google)
Continue with Design Process
Functional Model
Black Box Diagram
Dispense Pet Food Automatically

Finished Sticky Note WordTree



Identify Analogous Domains & Analogies

- Identify Analogous Domains
  - Parallel braches
  - Multiple potential analogies in the same domain
- Identify potential analogies (frequently, words that are both nouns & verbs)
  - Unusual words / domain specific words
  - [e.g. douse (lower quickly) "douse a sail" and reef (roll up (a portion of a sail) in order to reduce its area)]
  - Pay close attention to the "leaves"
  - Once one analogy or useful word is found, others on same branch are likely candidates too.


change
ndergo a change, come different in essence)
change surface
change shape, change form, deform
mill (roll out (metal) with a rolling machine)
fold
unfurl, unroll
roll up, furl
cog (roll steel ingots)
ikle, ruckle, crease, ikle, scrunch, unch up, crisp
collapse
douse (lower quickly) "douse a sail"
reef (roll up (a portion of a sail) in order to reduce its area)
bolt (make or roll into bolts) "bolt fabric"
take in, gather in (fold up) "take in the sails"
brail
cker, ruck, ck up
pucker, rumple, cockle, crumple, knit
deflate
concertina
coal (take in coal) "The big ship coaled"
incorporate

Generating Ideas and Analogies

1. Words from WordTree
2. Researched Potential Analogies (Google) from list and search for patents in the analogous domains you identified

5. *Analogy Example*

Nature Design-by-Analogy Example

Collapsible Sail

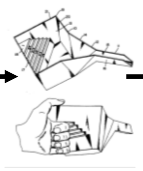
"Wings Take to the Water," 2000, BBC News
Reed, 2006, "The Future of Shipping", Popular Science

Example of Analogy: Same domain analogy

Problem Description

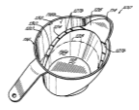
Liquid measuring device with convenient to read measurement scales

Analogy

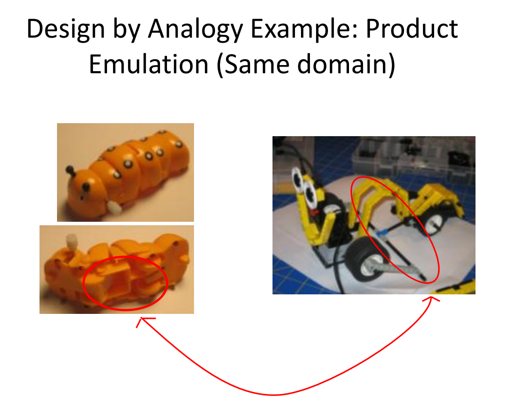


Historical Patent for this problem

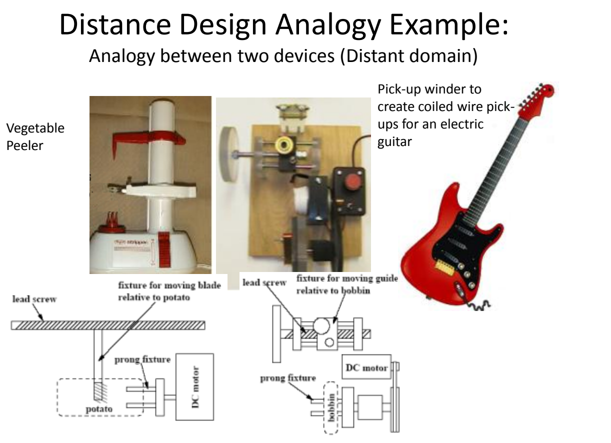
Concept



New Measuring Cup



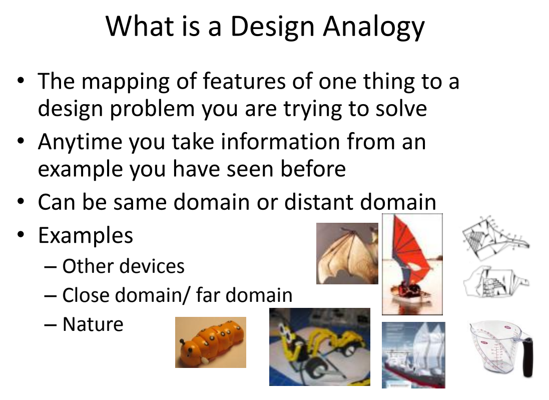
Design by Analogy Example: Product Emulation (Same domain)



Distance Design Analogy Example:

Analogy between two devices (Distant domain)

Vegetable Peeler
Pick-up winder to create coiled wire pick-ups for an electric guitar
lead screw
fixture for moving blade relative to potato
prong fixture
potato
DC motor
lead screw
fixture for moving guide relative to bobbin
DC motor
prong fixture
bobbin



What is a Design Analogy

- The mapping of features of one thing to a design problem you are trying to solve
- Anytime you take information from an example you have seen before
- Can be same domain or distant domain
- Examples
  - Other devices
  - Close domain/ far domain
  - Nature

3. Surveys

Pre-experiment Survey

Please answer the following questions for the WordTree Method.

	Not at all useful	Somewhat Useful	Useful	Very Useful
Overall, WordTree Method was:				
The WordTrees were:				

Listing analogies was:				
Listing analogous domains was:				
Writing new problem statements was:				

What is the value of each of the following for a **TYPICAL ENGINEERING DESIGN PROBLEM?**

	Zero value	A little value	Medium value	High value	Extremely valuable	Can’t remember
Background research / Literature review						
Mission Statement						
Quality Function Development (QFD, House of Quality)						
Black Box diagram						
Activity Diagram						
Function Structure						
Patent Search						
6-3-5						

Mind Maps						
TRIZ/TIPS						
Morph Matrix						
Pugh Charts						
WordTree Method						

What is the value of each of the following for a **DEIGN PROBLEM THAT REQUIRES AN INNOVATIVE SOLUTION?**

	Zero value	A little value	Medium value	High value	Extremely valuable	Can’t remember
Background research/Literature review						
Mission Statement						
Quality Function Development (QFD, House of Quality)						
Black Box diagram						
Activity Diagram						
Function Structure						
Patent Search						

6-3-5						
Mind Maps						
TRIZ/TIPS						
Morph Matrix						
Pugh Charts						
WordTree Method						

Assuming you are working as an engineer, how likely are you to use each of the following methods in the future?

	Very unlikely	Unlikely	Neutral	Likely	Very Likely	Can't remember
Background research/Literature review						
Mission Statement						
Quality Function Development (QFD, House of Quality)						
Black Box diagram						
Activity Diagram						
Function Structure						
Patent Search						

6-3-5						
Mind Maps						
TRIZ/TIPS						
Morph Matrix						
Pugh Charts						
WordTree Method						

Please answer the following questions for the WordTree Method:

	Strongly Disagree	Disagree	Neither Agree Nor Disagree	Agree	Strongly Agree
12. This method helped me to find analogies for my design problem.					
13. This method helped me to generate more ideas.					
14. This method helped me to generate more quality ideas					
15. This method was a waste					

of my time.					
16. The presentation of this method was easy to understand.					
17. This method was easy to use.					
18. I expect to use this method in the future.					
19. This method needs improvements.					
20. This method was useful.					
21. I like using the method.					
22. I expect to use this method in the future for design problems that require an innovative solution.					

Post-experiment Survey

Please answer the following questions for the WordTree Method including the WordTree Express program.

	Not at all useful	Somewhat Useful	Useful	Very Useful
Overall, WordTree Method was:				
The WordTrees were:				
Listing analogies was:				
Listing analogous domains was:				
Writing new problem statements was:				

What is the value of each of the following for a **TYPICAL ENGINEERING DESIGN PROBLEM?**

	Zero value	A little value	Medium value	High value	Extremely valuable	Can't remember
Background research/Literature review						
Mission Statement						
Quality Function Development (QFD, House of Quality)						

Black Box diagram						
Activity Diagram						
Function Structure						
Patent Search						
6-3-5						
Mind Maps						
TRIZ/TIPS						
Morph Matrix						
Pugh Charts						
WordTree Method non-automated (Paper-based)						
WordTree Method automated (e.g WordTree Express)						

What is the value of each of the following for a **DEIGN PROBLEM THAT REQUIRES AN INNOVATIVE SOLUTION?**

	Zero value	A little value	Medium value	High value	Extremely valuable	Can’t remember
Background						

research/Literature review						
Mission Statement						
Quality Function Development (QFD, House of Quality)						
Black Box diagram						
Activity Diagram						
Function Structure						
Patent Search						
6-3-5						
Mind Maps						
TRIZ/TIPS						
Morph Matrix						
Pugh Charts						
WordTree Method non-automated (Paper-based)						
WordTree Method automated (e.g WordTree Express)						

Assuming you are working as an engineer, how likely are you to use each of the following methods in the future?

	Very unlikely	Unlikely	Neutral	Likely	Very Likely	Can't remember
Background research/Literature review						
Mission Statement						
Quality Function Development (QFD, House of Quality)						
Black Box diagram						
Activity Diagram						
Function Structure						
Patent Search						
6-3-5						
Mind Maps						
TRIZ/TIPS						
Morph Matrix						
Pugh Charts						
WordTree Method non-						

automated (Paper-based)						▨
WordTree Method automated (e.g WordTree Express)						▨

Please answer the following questions for the WordTree Method including the WordTree Express program:

	Strongly Disagree	Disagree	Neither Agree Nor Disagree	Agree	Strongly Agree
23. This method helped me to find analogies for my design problem.					
24. This method helped me to generate more ideas.					
25. This method helped me to generate more quality ideas					
26. This method was a waste of my time.					
27. The presentation of this method was easy to					

understand.					
28. This method was easy to use.					
29. I expect to use this method in the future.					
30. This method needs improvements.					
31. This method was useful.					
32. I like using the method.					
33. I expect to use this method in the future for design problems that require an innovative solution.					

	Strongly Disagree	Disagree	Neither Agree Nor Disagree	Agree	Strongly Agree
I ran out of **time** before I ran out of ideas.					

I ran out of **ideas** before I ran out of time.					

How much engineering industrial work experience (experience not part of a class) do you have?

Full-time (35+ hrs/week) engineering work (internships or full-time work)

___________months

___________years

Part-time (less than 35 hrs/week) engineering work

___________hrs/week ___________months

___________years

Please answer the following

- Gender (check one):

Male	Female

- Age:

Years

- Years in Graduate school (check one):

1	2	3	4	Other

What did you like about the WordTree Method?

What steps in the WordTree Method were most useful?

How could the WordTree Method be improved?

What difficulties did you have when using the WordTree Method?

Where did you need more guidance from the WordTree Method?

What do you think of the WordTree Express program? How would you compare your experience when you made WordTrees manually to using this automated method?

Please state any additional comments you have about the experiment. Use the back of the paper if needed.

Thank you very much for your time.

EXPERIMENT SCRIPT

WordTree Idea Generation – Experimenter Script

Check list:

- Participant instruction packets (Sticky note and WTE)***
- Slides (projector setup)****
- WordTree Express tutorial
- Numbered papers (1-44)
- Sticky Notes
- Sticky Note instruction sheet
- Sticky Note Blanks (four 8.5X11)
- WTE instruction sheet
- Analogy list sheet 1
- Analogy list sheet 2
- Multiple color pens (black, blue, green, pink, maroon, light blue pen, light blue marker)***
- Computers for the participants (Tutorial screen up)
- Participant consent forms (2)
- Problem Statement sheet
- Surveys
- Stop watch
- Stapler
- Print out of WordTree Method Reminder***

- Tape

*****Make sure all files in WTE folder have been deleted*****

*****Make sure headphone are working*****

*****Test video playback*****

1. <u>Consent</u>

On the table:

- Participant consent forms
- BLACK pen
- 2 different color Sticky Notes (left)
- Sticky Note instruction sheet + 4 Blank sheets (left on top of sticky notes)
- WordTree Method Reminder (center)
- Problem statement (right)

On computer table:

- WTE tutorial hardcopy on participants left side (face down)
- WTE instruction sheet on participants right side (facedown)

When participants come, show them the work place.

*****Start stop watch*****

“Hello and thank you for taking time today to participate in this research study. Please turn off all cell phones. For this study, you are not supposed to monitor time using your watches or cell phones. So, please put your watches and cell phones in your back pack or the box on this table” (Show the box).

Check to make sure that the participants have no mobiles or watches with them.

“This study is evaluating different idea generation methods. Your task is to generate ideas and analogies for a design problem. The total time required for this study is 2 hours. Please read the consent form. You are not required to participate in this study and may end your participation at any time.”

Wait until all of the participants have finished reading to proceed with the experiment. Then say**,**

“If you agree to participate please sign the consent form and keep the second copy for your records.”

Wait for participants to sign the consent forms

Collect the consent forms

“Please put away your copy of the consent form”

2. <u>Pre-Experiment Survey (5 min)</u>

Place on the table

- Pre- Experiment survey

“Please fill out the given survey”

Collect the survey when finished

3. <u>Design problem (110 min)</u>

“OK, we are now beginning with the experiment. This experiment has multiple activities and the entire two hours will be required. Your effort will be compensated with extra credits for your design class or payment as discussed. You must agree to not discuss any aspects of this study with other mechanical engineering students in Texas A&M until after May 1, 2011 since this will bias the results. Are there any questions before we begin?”

Record the questions and answers in case of any.

Answer the questions if any.

“Multiple colors of pens are being used to keep track of when items are written. I will be asking you to switch colors periodically throughout the experiment.”

Show analogy examples and read from description

Place slides hardcopy on table

“Please look at the slides”

“You are being asked to generate ideas for a peanut shelling machine. Flip over the sheet on your right.”

Problem Description

In places like Haiti and certain West African countries, peanuts are a significant crop. Most peanut farmers shell their peanuts by hand, an inefficient and labor-intensive process. The goal is to build a low-cost, easy to manufacture peanut shelling machine that will increase the productivity of the peanut farmers. The target throughput is approximately 50Kg (110lbs) per hour.

Customer Needs:

- Must remove the shell with minimal damage to the peanuts.
- Electrical outlets are not available as a power source.
- A large amount of peanut must be quickly shelled.
- Low cost and easy to manufacture.

Functions include

- Import energy to the system.
- Break peanut shells.
- Separate peanut shells from the nut.

“This is a real problem from a website called Thinkcycle.org. Thinkcycle.org presents design needs from underserved populations. An efficient, low cost solution does not exist for this problem. Your ideas may be given to a design team working on this problem.”

“For this experiment you are being asked to use the WordTree Design-by-Analogy Method you were taught in class.”

<u>Activity 1 (20 min)</u>

Spend 20 minutes creating sticky note WordTrees for the following Key Problem Descriptors:

- **Shell**
- **Remove**
- **Separate**
- **Import Energy**

“A printout of the WordTree Method reminder is on your table for your reference”

“Flip over the stacks of papers on your table”

“Use the smaller stack of Sticky Notes for your keywords”

“Go ahead and start”

At 20 min

“Please stop the activity.”

*** Tape down sticky note WordTrees***

<u>Activity 2 (20 min)</u>

“Please move your chair to face the computer on your left and have a sit”

“For the next activity, you are required to generate WordTrees using a new software program called WordTree Express. To help you with this, we have recorded a short tutorial for you. Press the play button after you put on your headphones. When the tutorial finishes, take off your headphones”

“You may now put on your headphones and press the play button”

Allow tutorial to finish playing

“Please close the tutorial”

“Do you have any questions?”

Answer the questions if any

“Please raise your hand if you have any questions”

“Flip over the papers on the table”

“You are to generate WordTrees using WTE for the peanut shelling problem”

“You may refer to the hardcopy of the tutorial on your left as needed”

“To generate your WordTrees with WTE, use the 7th sense of the keyword “Shell” and the 5th sense of the keyword “Separate”.”

“Remember to circle all the words of interest on your WordTrees and save any changes you make to your WordTrees.”

“OK, you may now start.”

After 20 minutes

“Please stop the activity”

Activity 3 (10 min)

Place Analogy list sheet 1 on table

“Using the WordTrees you’ve generated, identify and list potential analogies and analogous domains. Write down every possible analogy or analogous domain, even if it is not directly from your WordTree or it is technically infeasible, wild, or crazy. You have ten minutes to do this”

Activity 4 (60 min)

Add to the table

- Numbered sheets (1-44)

- Maroon pen

"Now you are being asked to generate solutions and continue generating analogies for the peanut shelling problem. Use the single words from your WordTrees."

"The goal is to generate as many solutions as possible with as high of quality as possible and with as great of variety as possible. Technically infeasible, wild, non-standard and far out ideas are also encouraged. This helps to generate unique feasible solutions."

"Use sketches and words to describe your ideas."

"This session contains multiple tasks that will require the rest of the time. You may choose when to end the idea generation session and move to the next task. When you are ready to move to the next task please raise your hand."

"You can use the rest of the time for idea generation."

"Begin generating ideas by using the single words from your WordTrees."

"Remember, the amount of extra credit you will receive depends on your effort and performance."

"Go ahead and start"

Pen colors **"Switch to the X pen"**

Time	Color	Start	End
0-15	Maroon		
15-30	Light blue pen		
30-40	Orange pen		
40-45	Pink		

**computer			
45-60	purple		

Make sure I write down what time they want to end the activity

Hand out sheet on why they decided to stop idea generation

Hand out motivation sheet

"Remember you can use your WordTrees to help you generate ideas."

At 45 minutes

"If you want to use it, the computer has internet access and is available to assist you in solving the peanut shelling device problem. You can use it to research the potential analogies you identified in the previous activity and search for patents in the analogous domains. You do not need to use it. If you gain ideas from using the web, be sure to write down the reference information (the website address or other appropriate information)."

"Please raise your hand when you want to use the computer."

Make sure I write down what time they start using the computer

End of Activity

"Please stop all activities"

Add to the table

- Analogy list sheet 2

“Note any analogies you used to help you find solutions. Please go ahead and do this now”

If you used the computer to search for ideas, write down a short description of how you searched. What search engines did you use? What terms did you search for?

Collect all the papers

4. <u>Post Experiment Survey (5 min)</u>

Add to the table*

- Post-Experiment Survey

“This is the final part of the experiment. Please fill out the given survey”

“In the section on years in graduate school, indicate if years include Masters only or Masters and PhD”

Collect the surveys when finished

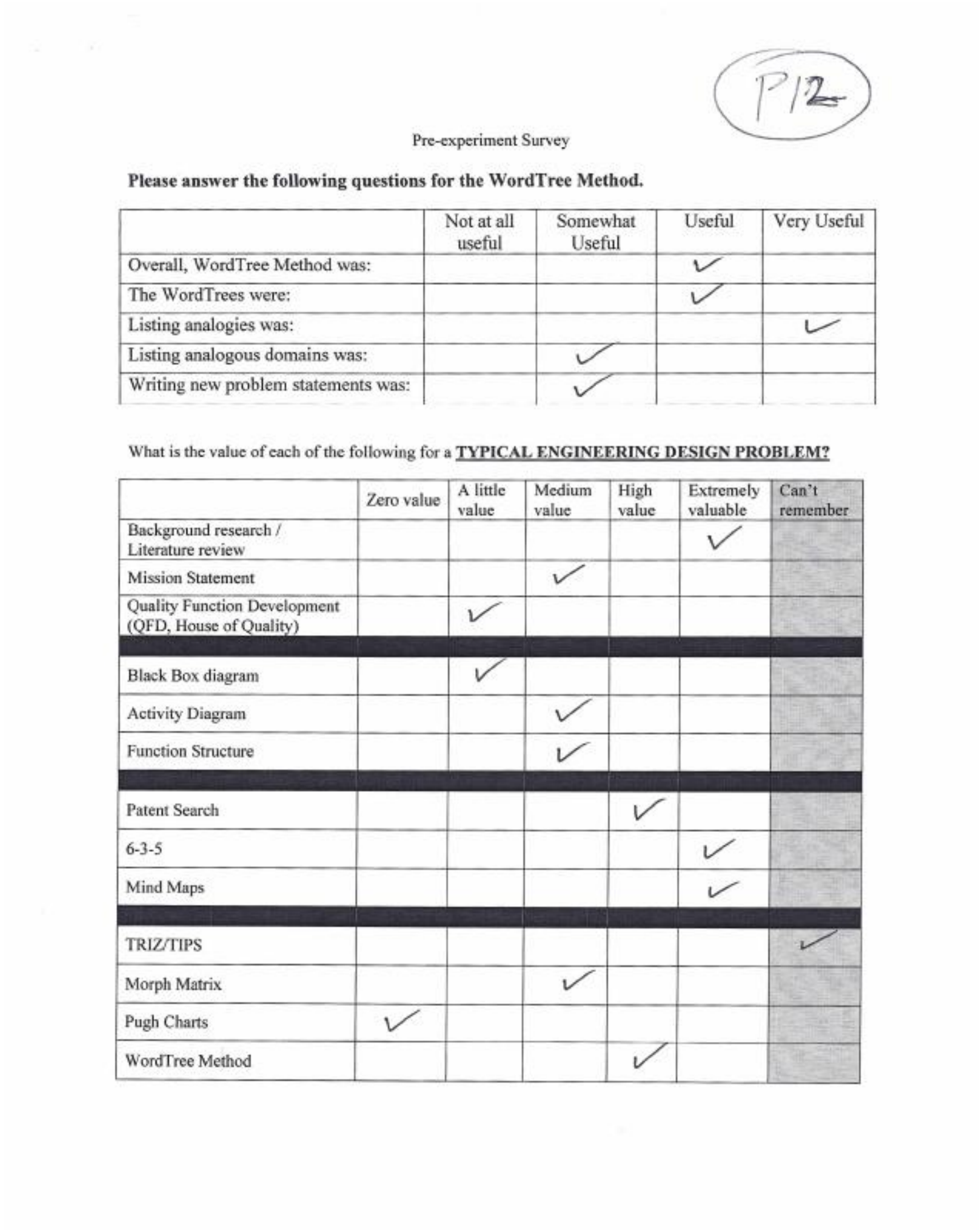
5. <u>Disbursement</u>

“Thank you for your participation. I will make sure that you receive your extra credit or payment for your participation. This concludes your portion of the study. Please remember to not discuss this study with your classmates until after May 1, 2011 since this will bias the data. If you have any questions about this study I can answer them at this time. “

Record the questions and answers in case of any.

APPENDIX B

SAMPLE SOLUTION



P12

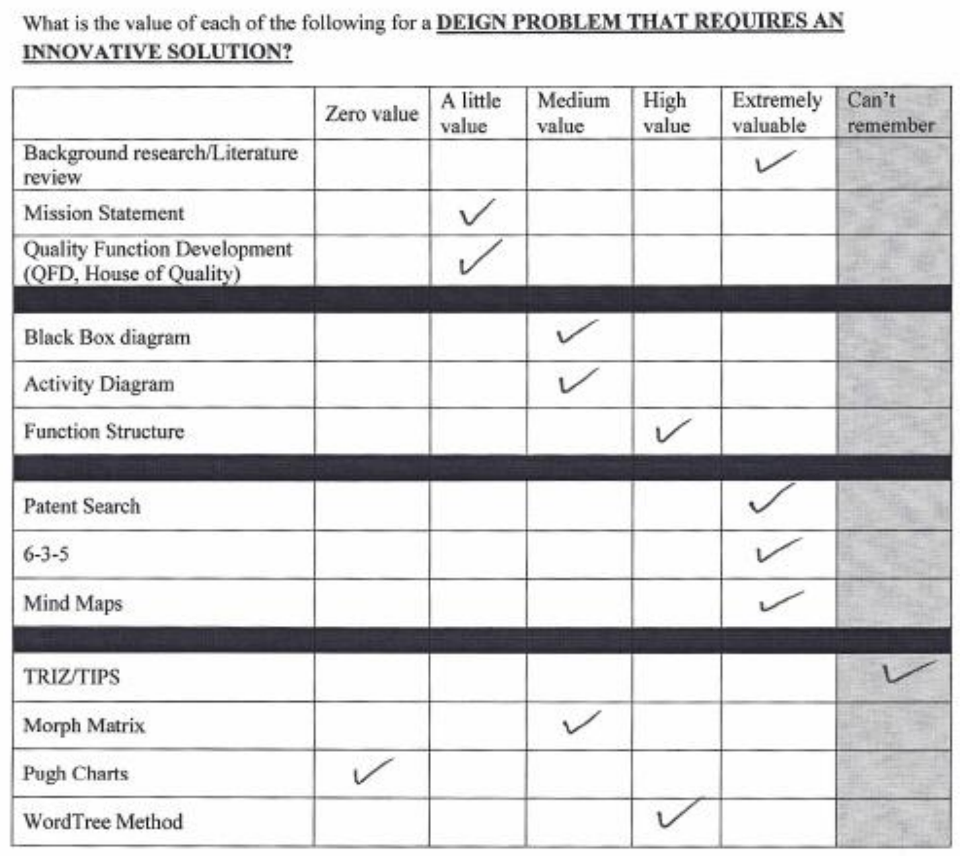
Pre-experiment Survey

Please answer the following questions for the WordTree Method.

	Not at all useful	Somewhat Useful	Useful	Very Useful
Overall, WordTree Method was:			✓	
The WordTrees were:			✓	
Listing analogies was:				✓
Listing analogous domains was:		✓		
Writing new problem statements was:		✓		

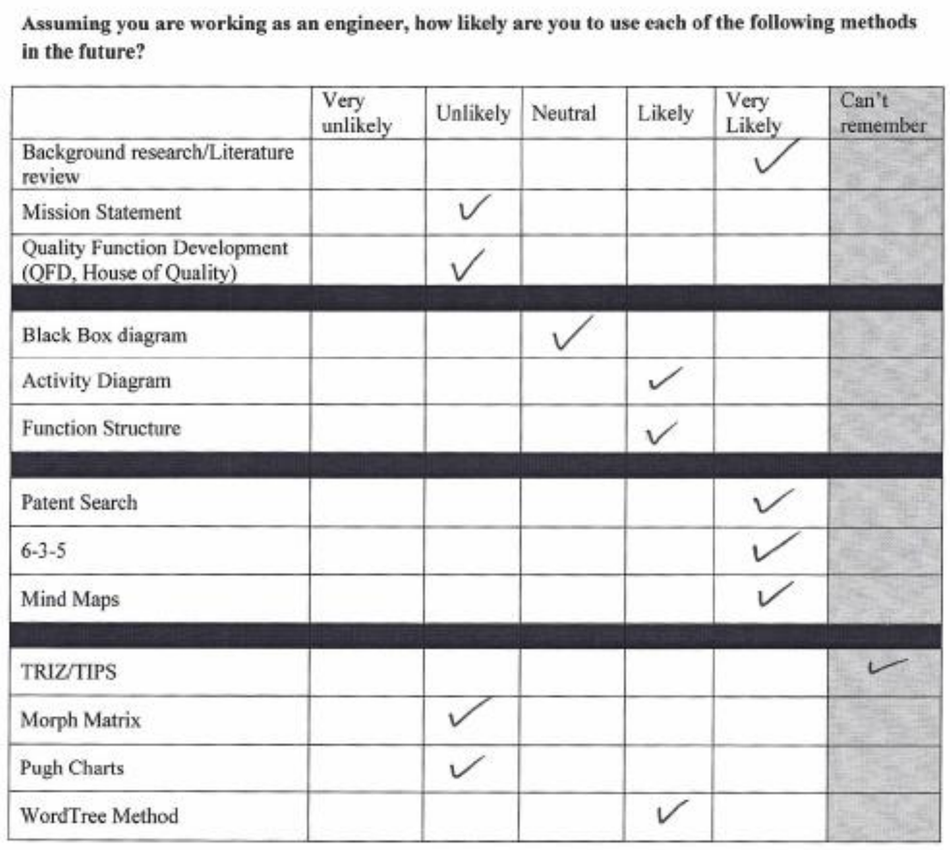
What is the value of each of the following for a **TYPICAL ENGINEERING DESIGN PROBLEM?**

	Zero value	A little value	Medium value	High value	Extremely valuable	Can't remember
Background research / Literature review					✓	
Mission Statement			✓			
Quality Function Development (QFD, House of Quality)		✓				
Black Box diagram		✓				
Activity Diagram			✓			
Function Structure			✓			
Patent Search				✓		
6-3-5					✓	
Mind Maps					✓	
TRIZ/TIPS						✓
Morph Matrix			✓			
Pugh Charts	✓					
WordTree Method				✓		



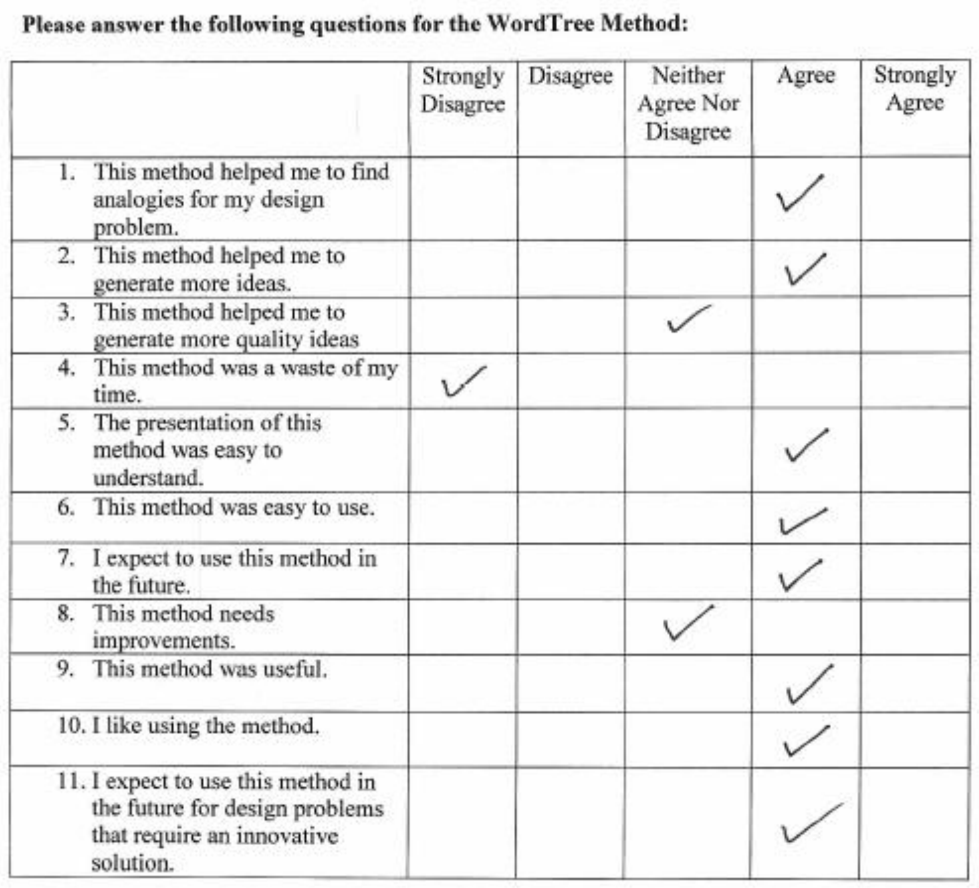
What is the value of each of the following for a **<u>DEIGN PROBLEM THAT REQUIRES AN INNOVATIVE SOLUTION?</u>**

	Zero value	A little value	Medium value	High value	Extremely valuable	Can't remember
Background research/Literature review					✓	
Mission Statement		✓				
Quality Function Development (QFD, House of Quality)		✓				
Black Box diagram			✓			
Activity Diagram			✓			
Function Structure				✓		
Patent Search					✓	
6-3-5					✓	
Mind Maps					✓	
TRIZ/TIPS						✓
Morph Matrix			✓			
Pugh Charts	✓					
WordTree Method				✓		



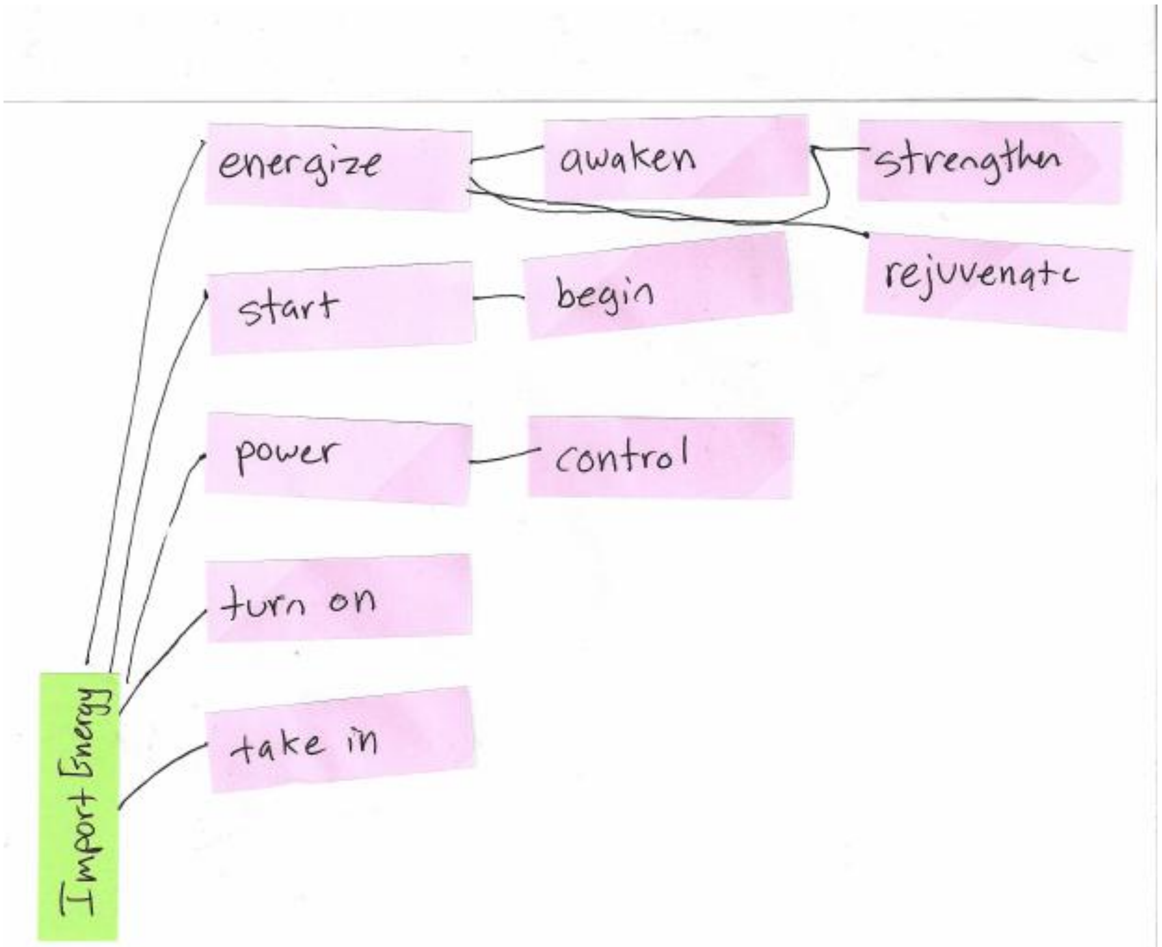
Assuming you are working as an engineer, how likely are you to use each of the following methods in the future?

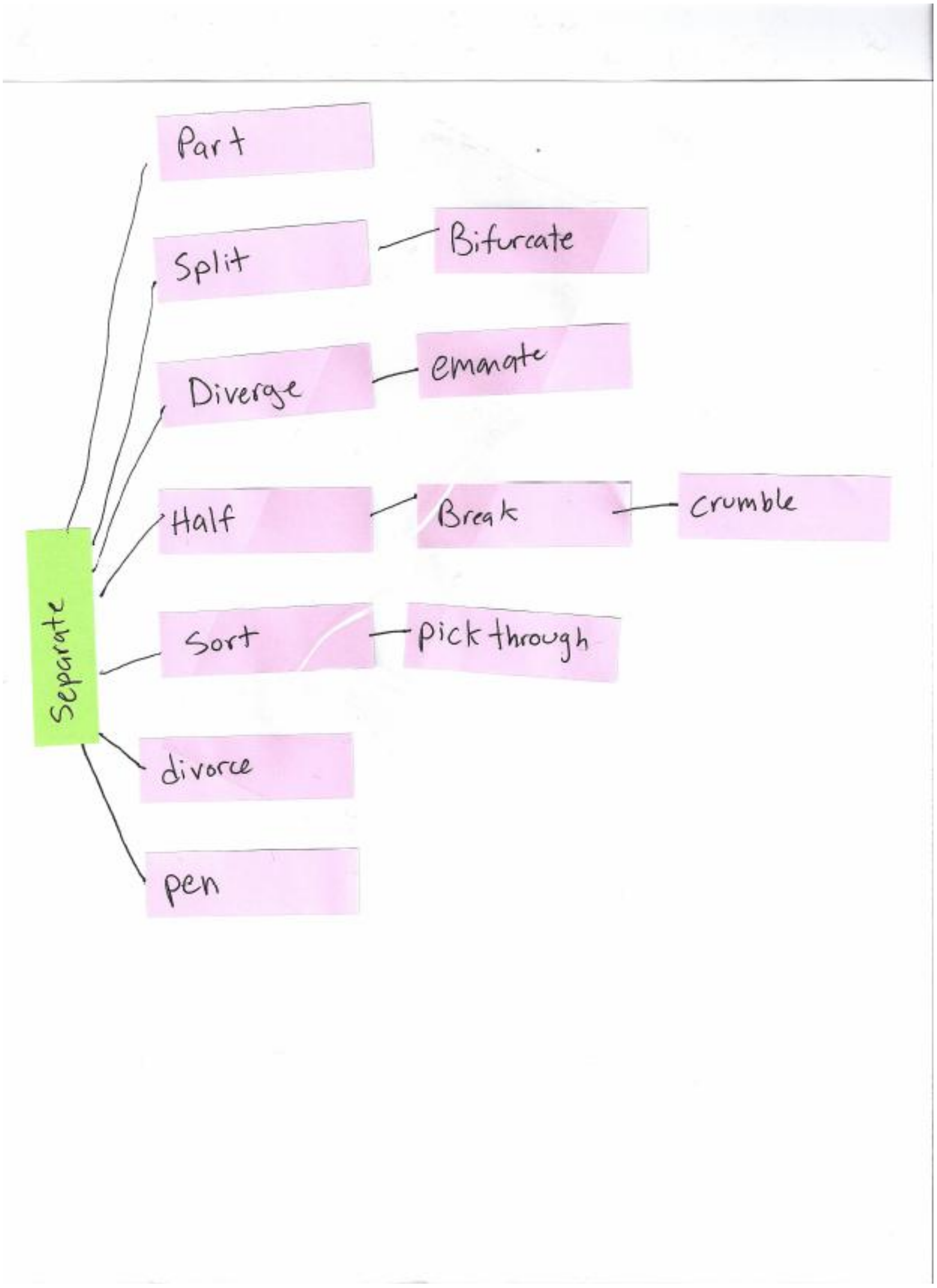
	Very unlikely	Unlikely	Neutral	Likely	Very Likely	Can't remember
Background research/Literature review					✓	
Mission Statement		✓				
Quality Function Development (QFD, House of Quality)		✓				
Black Box diagram			✓			
Activity Diagram				✓		
Function Structure				✓		
Patent Search					✓	
6-3-5					✓	
Mind Maps					✓	
TRIZ/TIPS						✓
Morph Matrix		✓				
Pugh Charts		✓				
WordTree Method				✓		

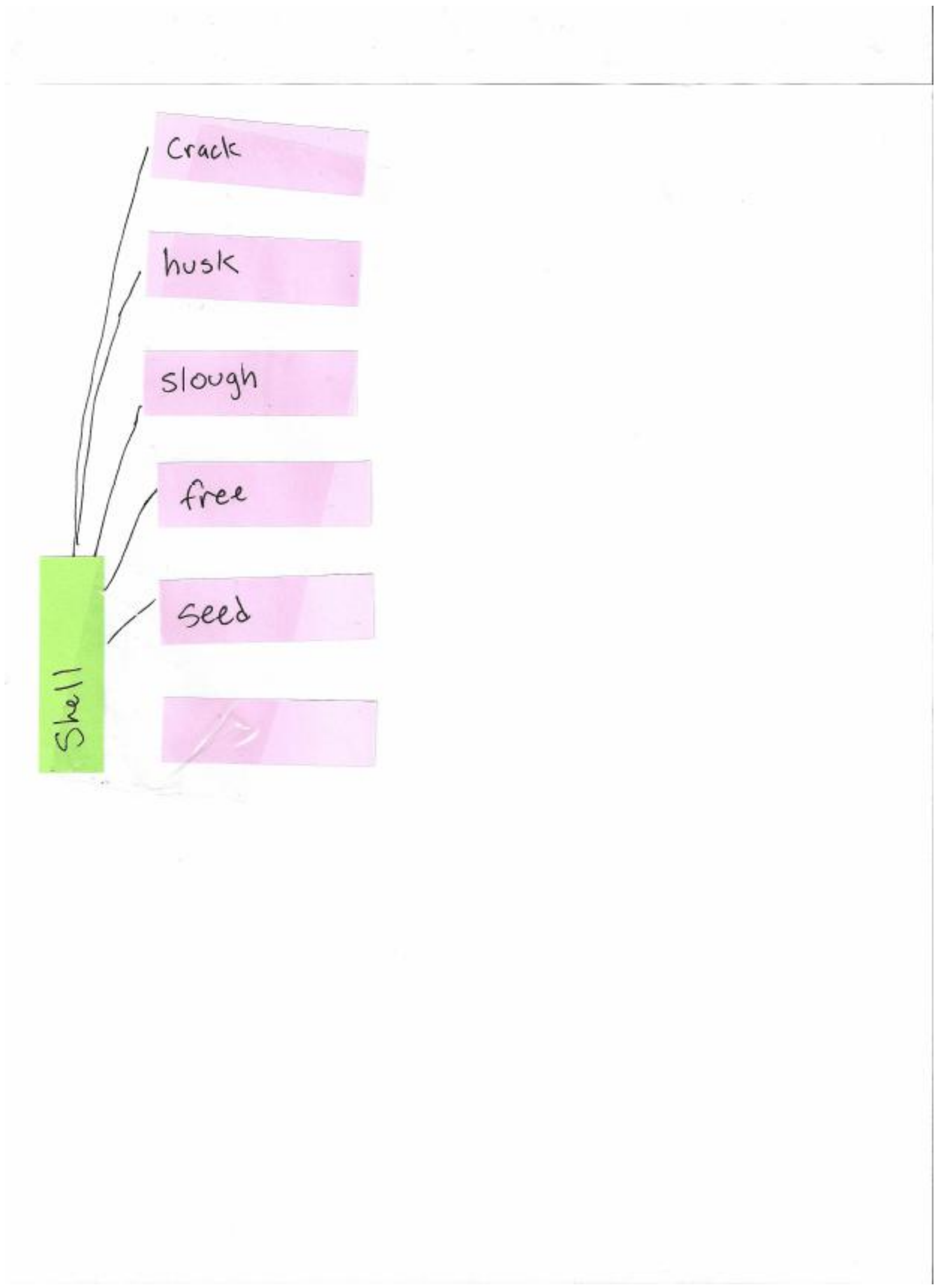


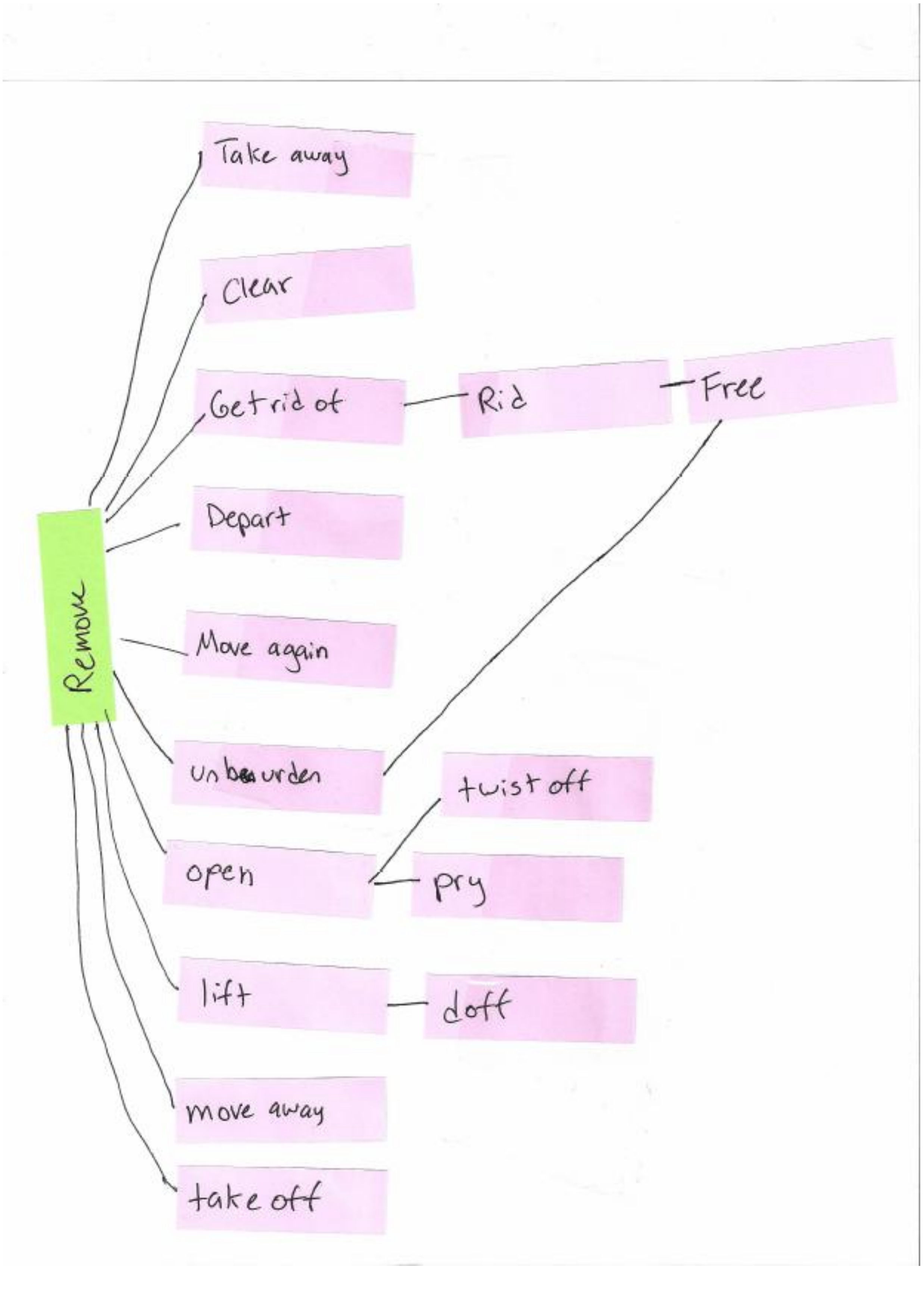
Please answer the following questions for the WordTree Method:

	Strongly Disagree	Disagree	Neither Agree Nor Disagree	Agree	Strongly Agree
1. This method helped me to find analogies for my design problem.				✓	
2. This method helped me to generate more ideas.				✓	
3. This method helped me to generate more quality ideas			✓		
4. This method was a waste of my time.	✓				
5. The presentation of this method was easy to understand.				✓	
6. This method was easy to use.				✓	
7. I expect to use this method in the future.				✓	
8. This method needs improvements.			✓		
9. This method was useful.				✓	
10. I like using the method.				✓	
11. I expect to use this method in the future for design problems that require an innovative solution.				✓	


Import Energy
energize
awaken
strengthen
rejuvenate
start
begin
power
control
turn on
take in


Separate
Part
Split
Bifurcate
Diverge
emanate
Half
Break
crumble
Sort
pick through
divorce
pen


Crack
husk
slough
free
seed
Shell


Remove
Take away
Clear
Get rid of
Rid
Free
Depart
Move again
unburden
twist off
open
pry
lift
doff
move away
take off

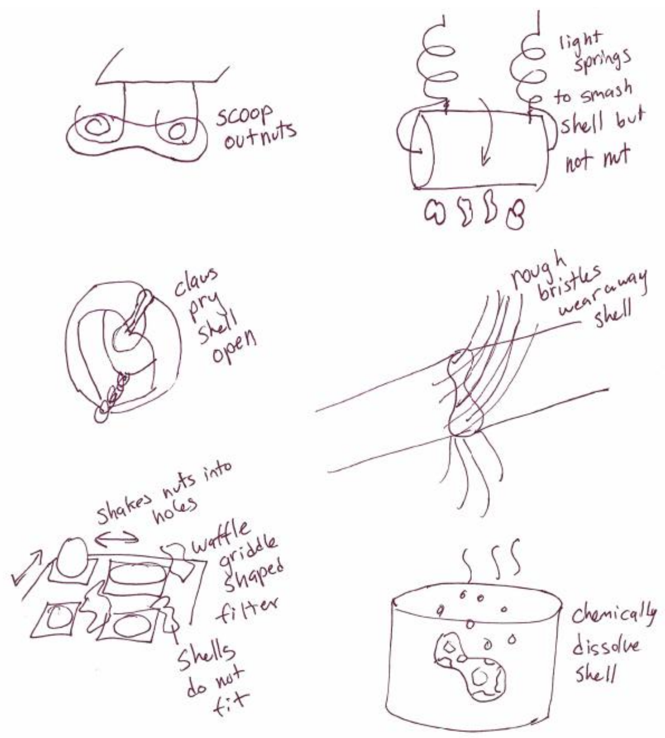
Identify and list potential analogies and analogies domains from your WordTrees. You will have ten minutes to do this.

Potential Analogies

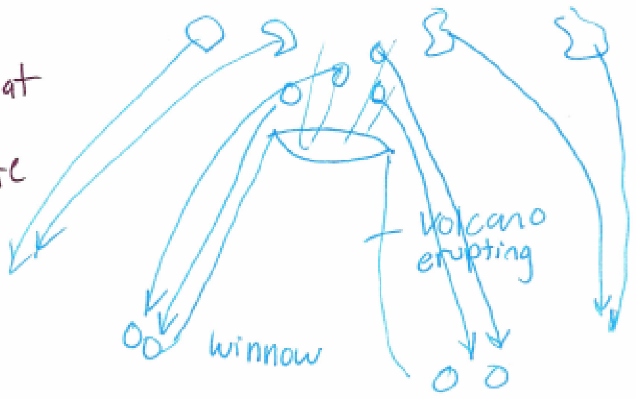
1. waffle griddle filter shape
2. corn husking/shucking
3. other nut types - shelling
4. divorce
5. filter water
6. cracking eggs to cook
7. unravelling a sweater
8. dispersing fertilizer
9. volcano erupting
10. panning for gold
11. pitting olives
12. Opening a suitcase
13. peeling a banana
14. tearing away present wrapping paper
15. Chemically taking off the support plastic on a 3D printed model
16. scalping grass
17. carving out pumpkin guts
18.
19.
20.

Potential Analogous Domains

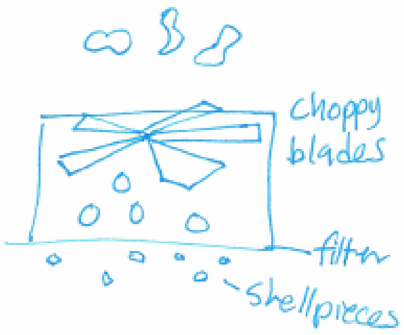
1. cooking
2. produce preparation
3. nuts
4. dividing people
5. food preparation
6. Cooking
7.
8. gardening
9. nature
10. historically separating
11. cooking
12. travel
13. cooking
14. holiday activities
15. [illegible]
16. gardening
17. holiday activities
18.
19.
20.

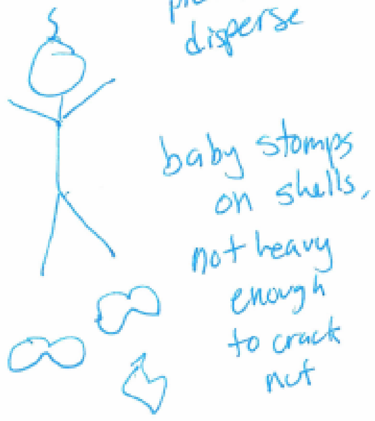

scoop out nuts
light springs to smash shell but not nut
claws pry shell open
rough bristles wear away shell
shakes nuts into holes
waffle griddle shaped filter
shells do not fit
chemically dissolve shell

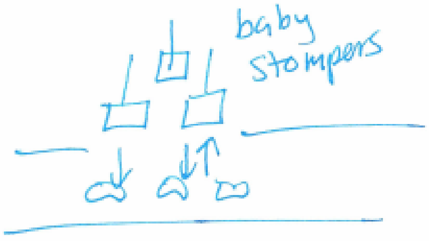
shells float to separate

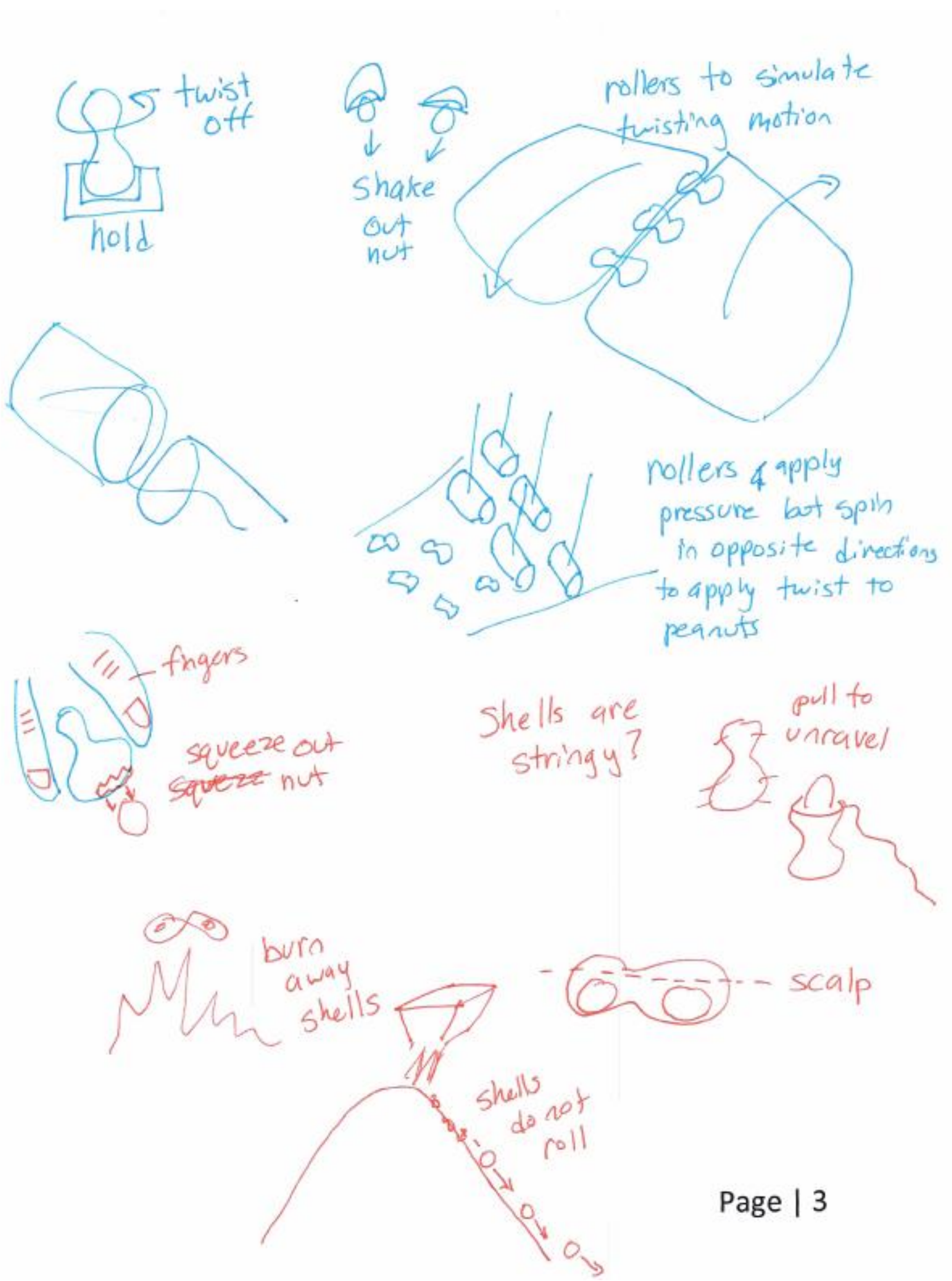

volcano erupting
winnow

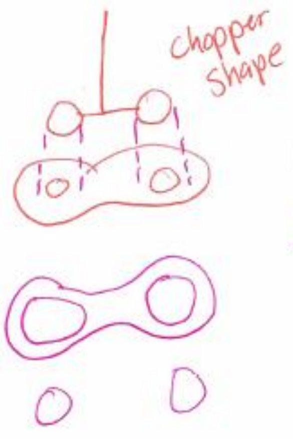
shell pieces disperse

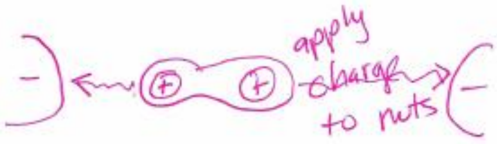

choppy blades
filter
shellpieces

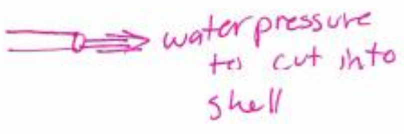

baby stomps on shells, not heavy enough to crack nut

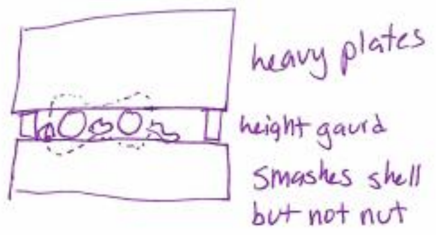

baby stompers

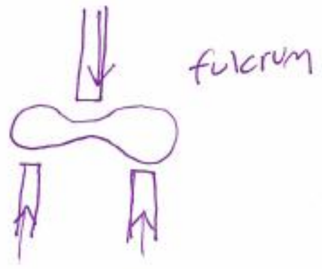

twist off
hold
shake out nut
rollers to simulate twisting motion
rollers & apply pressure but spin in opposite directions to apply twist to peanuts
fingers
squeeze out ~~squeeze~~ nut
Shells are stringy?
pull to unravel
burn away shells
scalp
shells do not roll
Page | 3

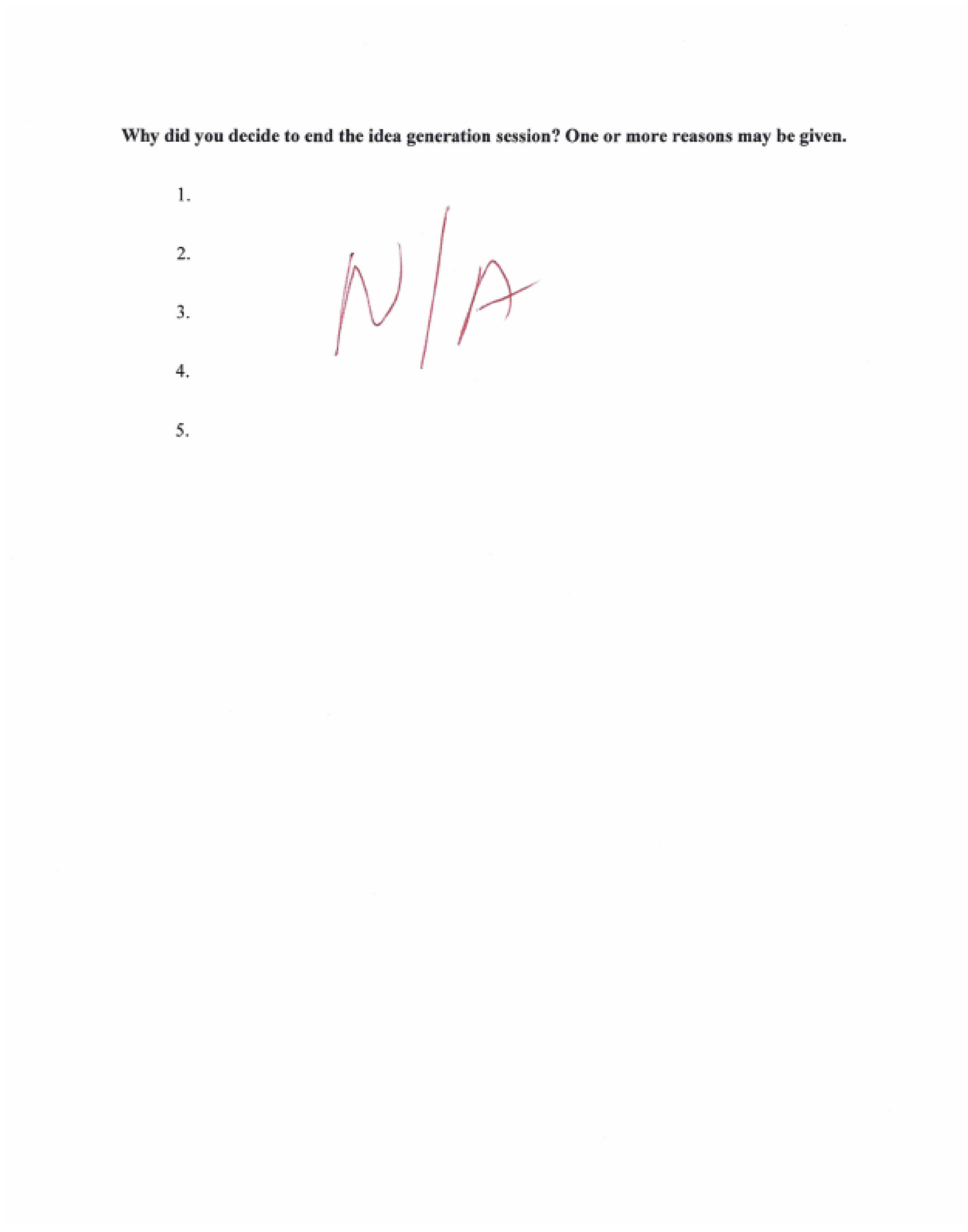

chopper shape


apply charge to nuts


water pressure to cut into shell


heavy plates
height gaurd
smashes shell but not nut


fulcrum



Why did you decide to end the idea generation session? One or more reasons may be given.

1.

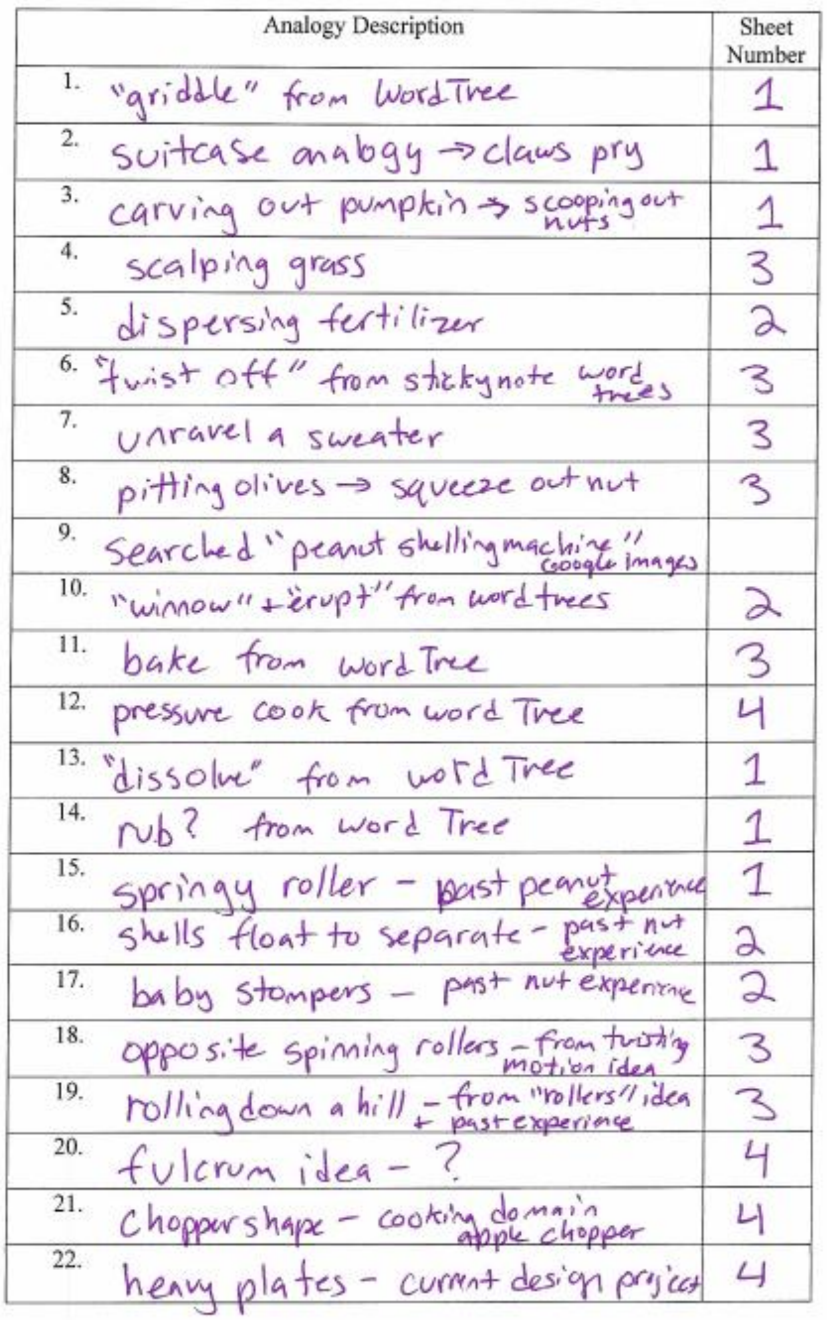
2.

3.

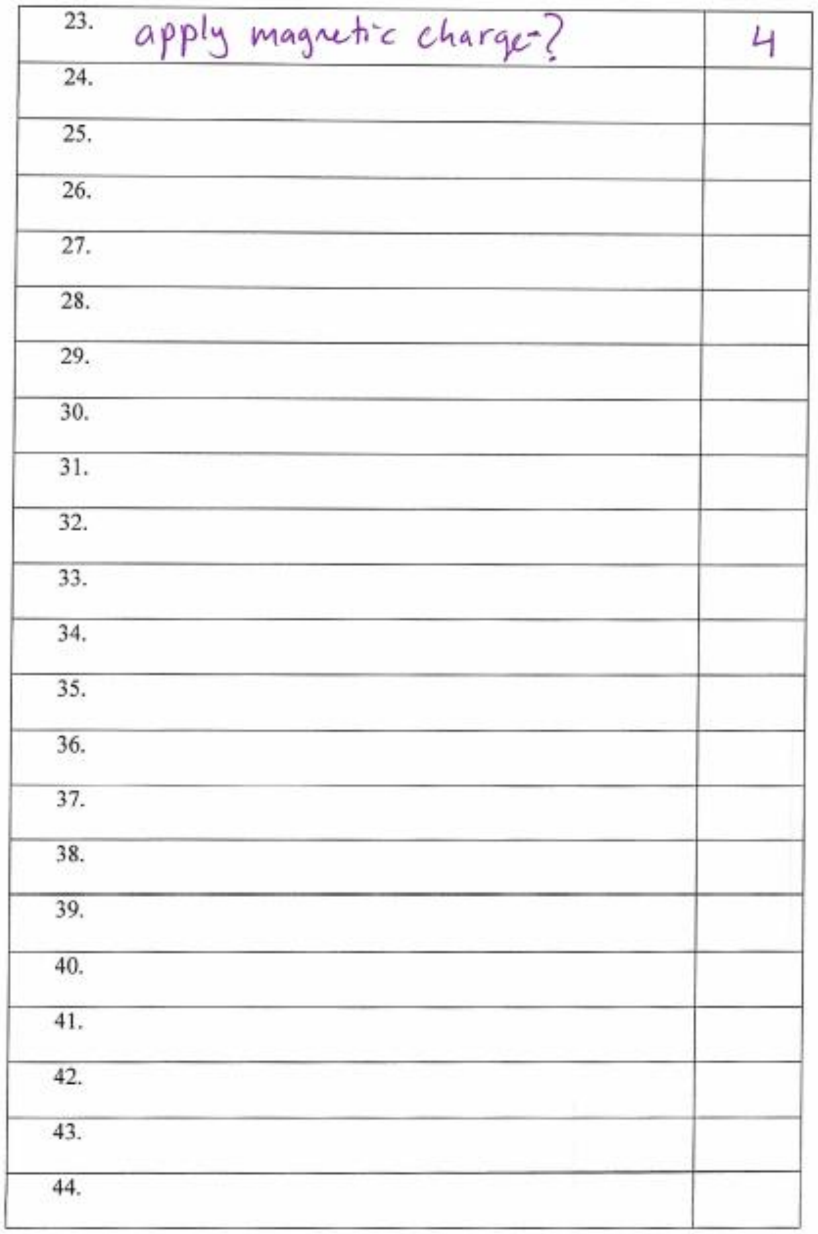
4.

5.

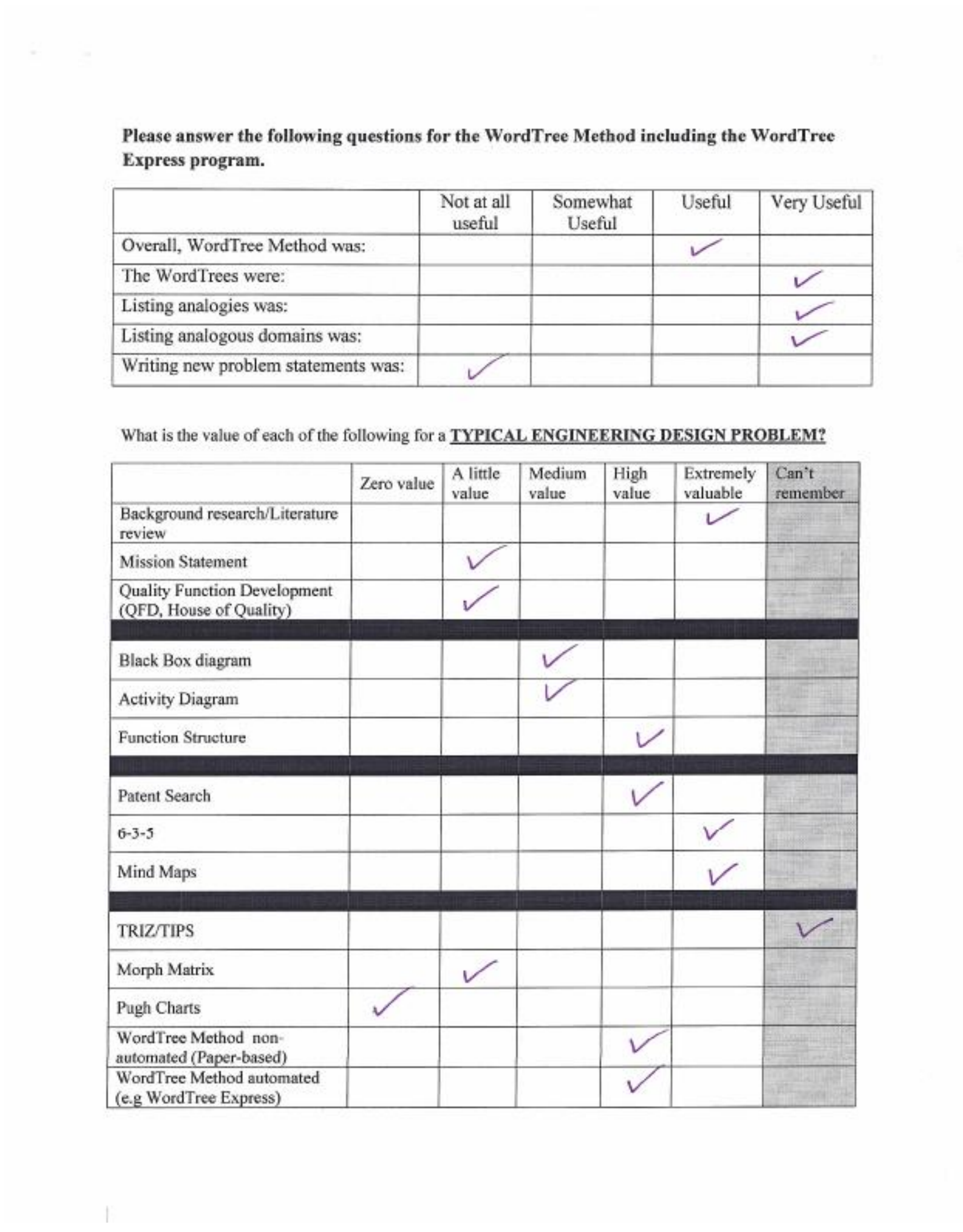
N/A



Analogy Description	Sheet Number
1. "griddle" from WordTree	1
2. suitcase analogy → claws pry	1
3. carving out pumpkin → scooping out nuts	1
4. scalping grass	3
5. dispersing fertilizer	2
6. "twist off" from stickynote word trees	3
7. unravel a sweater	3
8. pitting olives → squeeze out nut	3
9. Searched "peanut shelling machine" Google images	
10. "winnow" + "erupt" from word trees	2
11. bake from Word Tree	3
12. pressure cook from word Tree	4
13. "dissolve" from word Tree	1
14. rub? from word Tree	1
15. springy roller - past peanut experience	1
16. shells float to separate - past nut experience	2
17. baby stompers - past nut experience	2
18. opposite spinning rollers - from twisting motion idea	3
19. rolling down a hill - from "rollers" idea + past experience	3
20. fulcrum idea - ?	4
21. Chopper shape - cooking domain apple chopper	4
22. heavy plates - current design project	4



23. apply magnetic charge?	4
24.	
25.	
26.	
27.	
28.	
29.	
30.	
31.	
32.	
33.	
34.	
35.	
36.	
37.	
38.	
39.	
40.	
41.	
42.	
43.	
44.	

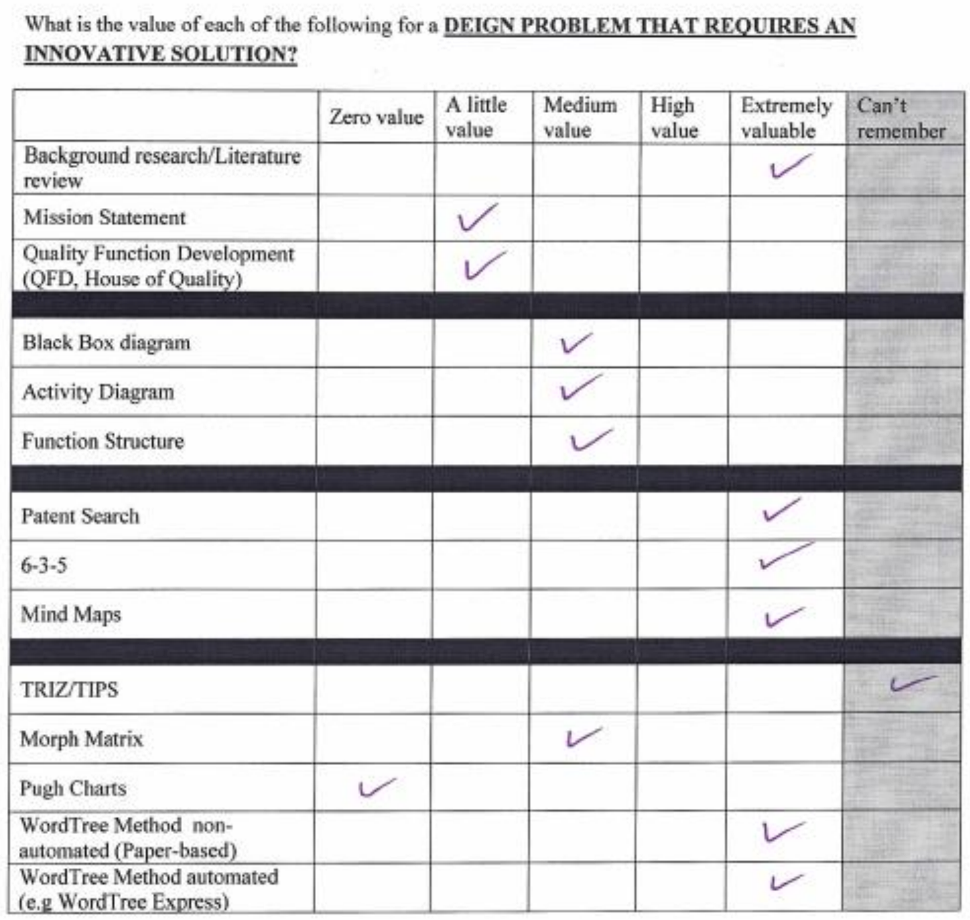


Please answer the following questions for the WordTree Method including the WordTree Express program.

	Not at all useful	Somewhat Useful	Useful	Very Useful
Overall, WordTree Method was:			✓	
The WordTrees were:				✓
Listing analogies was:				✓
Listing analogous domains was:				✓
Writing new problem statements was:	✓			

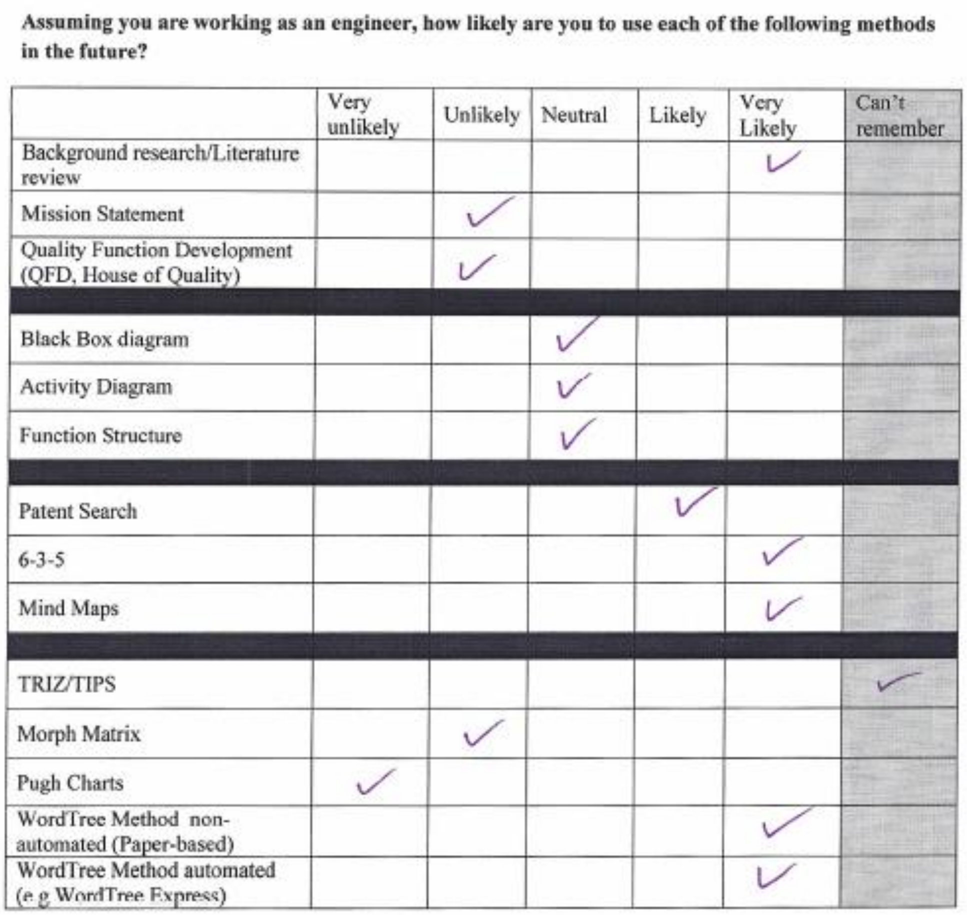
What is the value of each of the following for a **<u>TYPICAL ENGINEERING DESIGN PROBLEM?</u>**

	Zero value	A little value	Medium value	High value	Extremely valuable	Can't remember
Background research/Literature review					✓	
Mission Statement		✓				
Quality Function Development (QFD, House of Quality)		✓				
Black Box diagram			✓			
Activity Diagram			✓			
Function Structure				✓		
Patent Search				✓		
6-3-5					✓	
Mind Maps					✓	
TRIZ/TIPS						✓
Morph Matrix		✓				
Pugh Charts	✓					
WordTree Method non-automated (Paper-based)				✓		
WordTree Method automated (e.g WordTree Express)				✓		



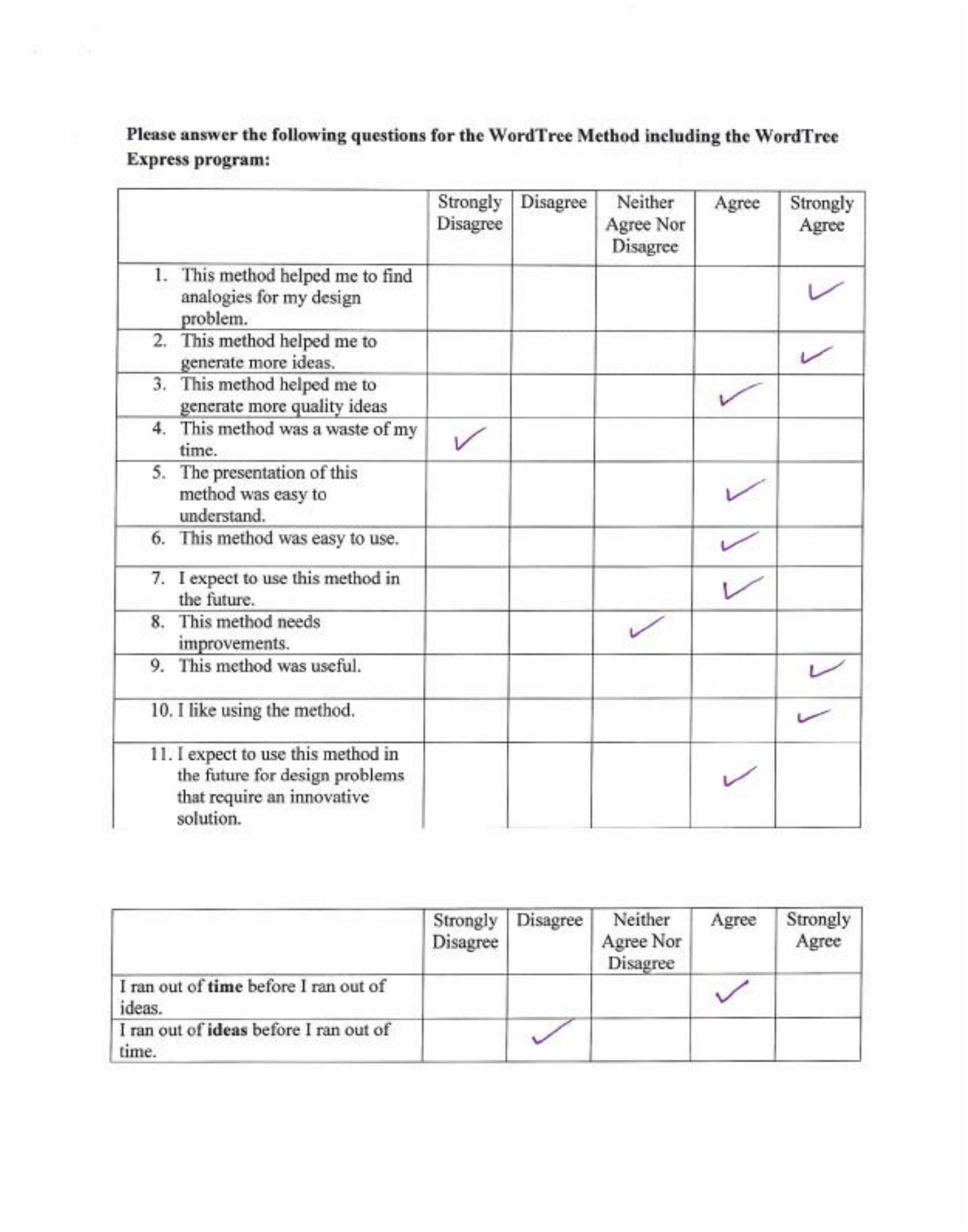
What is the value of each of the following for a **DEIGN PROBLEM THAT REQUIRES AN INNOVATIVE SOLUTION?**

	Zero value	A little value	Medium value	High value	Extremely valuable	Can't remember
Background research/Literature review					✓	
Mission Statement		✓				
Quality Function Development (QFD, House of Quality)		✓				
Black Box diagram			✓			
Activity Diagram			✓			
Function Structure			✓			
Patent Search					✓	
6-3-5					✓	
Mind Maps					✓	
TRIZ/TIPS						✓
Morph Matrix			✓			
Pugh Charts	✓					
WordTree Method non-automated (Paper-based)					✓	
WordTree Method automated (e.g WordTree Express)					✓	



Assuming you are working as an engineer, how likely are you to use each of the following methods in the future?

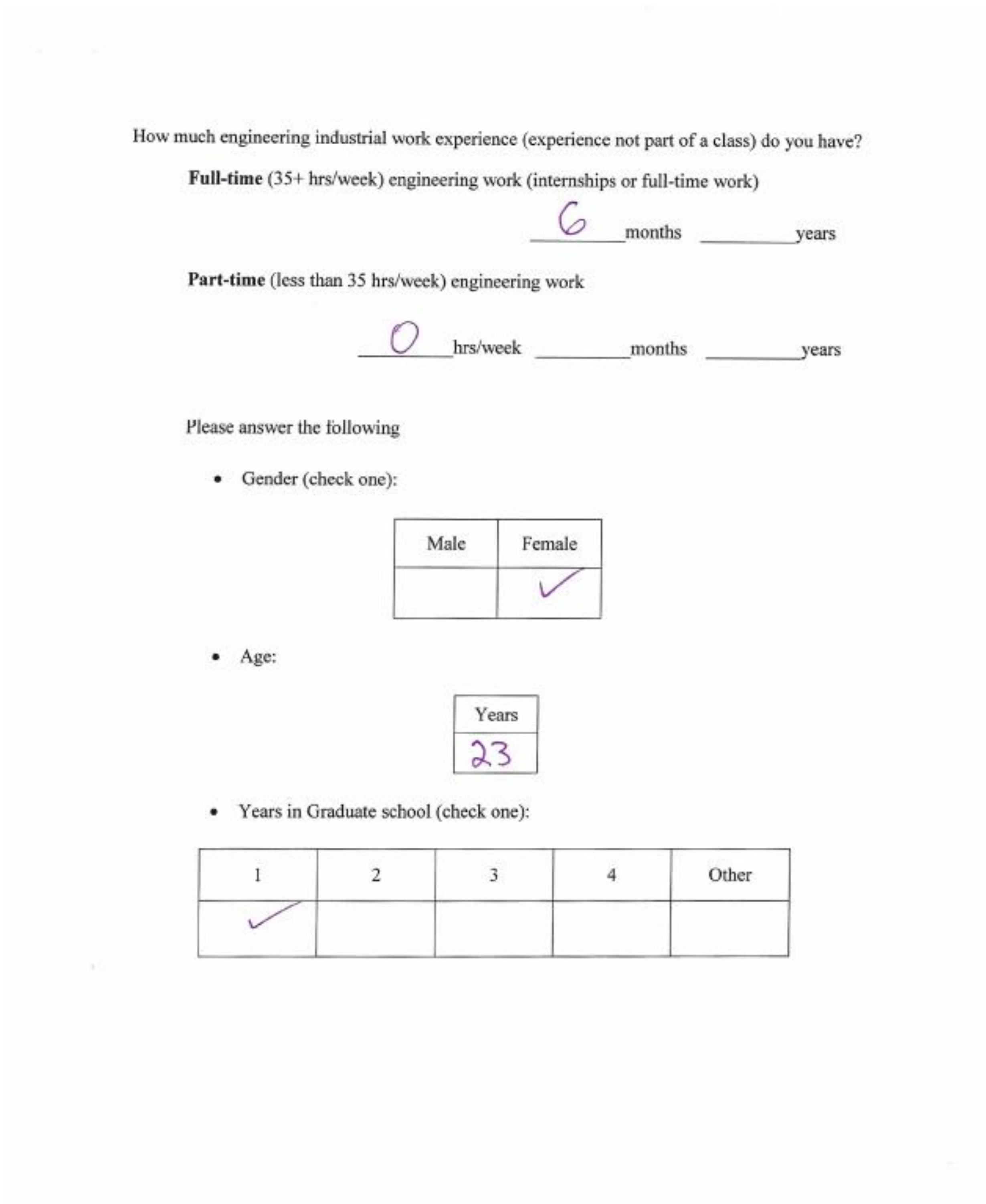
	Very unlikely	Unlikely	Neutral	Likely	Very Likely	Can't remember
Background research/Literature review					✓	
Mission Statement		✓				
Quality Function Development (QFD, House of Quality)		✓				
Black Box diagram			✓			
Activity Diagram			✓			
Function Structure			✓			
Patent Search				✓		
6-3-5					✓	
Mind Maps					✓	
TRIZ/TIPS						✓
Morph Matrix		✓				
Pugh Charts	✓					
WordTree Method non-automated (Paper-based)					✓	
WordTree Method automated (e.g WordTree Express)					✓	



Please answer the following questions for the WordTree Method including the WordTree Express program:

	Strongly Disagree	Disagree	Neither Agree Nor Disagree	Agree	Strongly Agree
1. This method helped me to find analogies for my design problem.					✓
2. This method helped me to generate more ideas.					✓
3. This method helped me to generate more quality ideas				✓	
4. This method was a waste of my time.	✓				
5. The presentation of this method was easy to understand.				✓	
6. This method was easy to use.				✓	
7. I expect to use this method in the future.				✓	
8. This method needs improvements.			✓		
9. This method was useful.					✓
10. I like using the method.					✓
11. I expect to use this method in the future for design problems that require an innovative solution.				✓	

	Strongly Disagree	Disagree	Neither Agree Nor Disagree	Agree	Strongly Agree
I ran out of **time** before I ran out of ideas.				✓	
I ran out of **ideas** before I ran out of time.		✓			



How much engineering industrial work experience (experience not part of a class) do you have?

Full-time (35+ hrs/week) engineering work (internships or full-time work)

6 months ______ years

Part-time (less than 35 hrs/week) engineering work

0 hrs/week ______ months ______ years

Please answer the following

- Gender (check one):

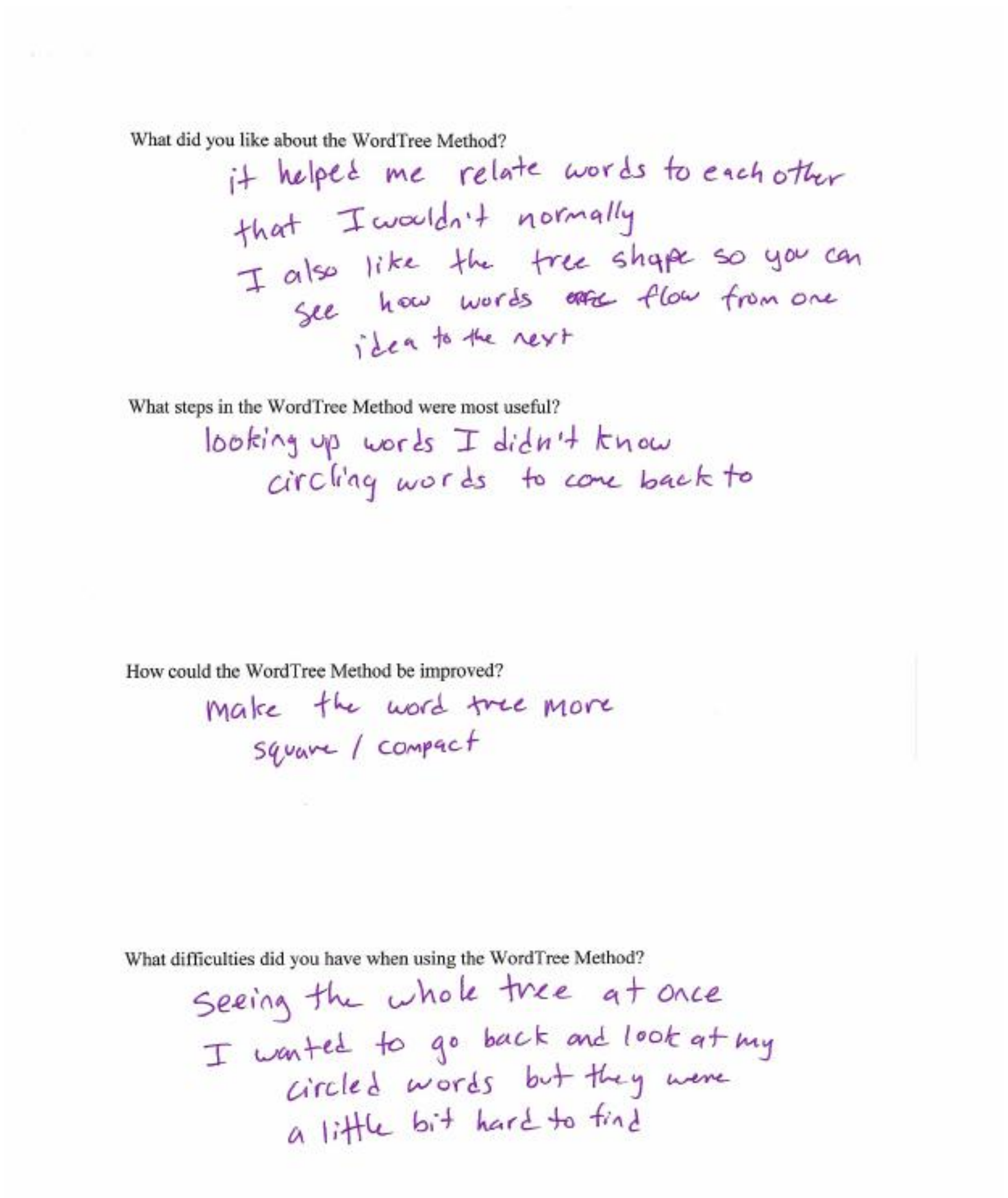
Male	Female
	✓

- Age:

Years
23

- Years in Graduate school (check one):

1	2	3	4	Other
✓				



What did you like about the WordTree Method?

it helped me relate words to each other that I wouldn't normally
I also like the tree shape so you can see how words ~~[illegible]~~ flow from one idea to the next

What steps in the WordTree Method were most useful?

looking up words I didn't know
circling words to come back to

How could the WordTree Method be improved?

make the word tree more square / compact

What difficulties did you have when using the WordTree Method?

Seeing the whole tree at once
I wanted to go back and look at my circled words but they were a little bit hard to find

Where did you need more guidance from the WordTree Method?

no where?

What do you think of the WordTree Express program? How would you compare your experience when you made WordTrees manually to using this automated method?

I like how easily it made the word tree for me, but maybe it could also generate a long list of the words and the structure could be shown in some other way in addition to the tree

It's much easier than doing the wordnet tree by hand

Please state any additional comments you have about the experiment. Use the back of the paper if needed.

fun

very useful

learned new words

Thank you very much for your time.

WORDTREE EXPRESS TUTORIAL/MANUAL

WordTree Express User's Guide/Tutorial ver.1.0

INSTALLATION

1. Copy the **WTE package** folder to your desktop.
2. Copy the "**WTE**" and "**WTE database**" folders to **C:\**
3. Install the Graphviz program by running the "**graphviz-2.26.3.msi**" file in the **WTE package** folder.
4. Install **Inkscape** by running the **Inkscape-0.47-3.exe** file in the **WTE package** folder.
5. Restart your computer (recommended)
6. Install WordTree Express by double clicking the **setup.exe** file in the **Desktop\WTE Package\WTE deploy\Debug** folder.

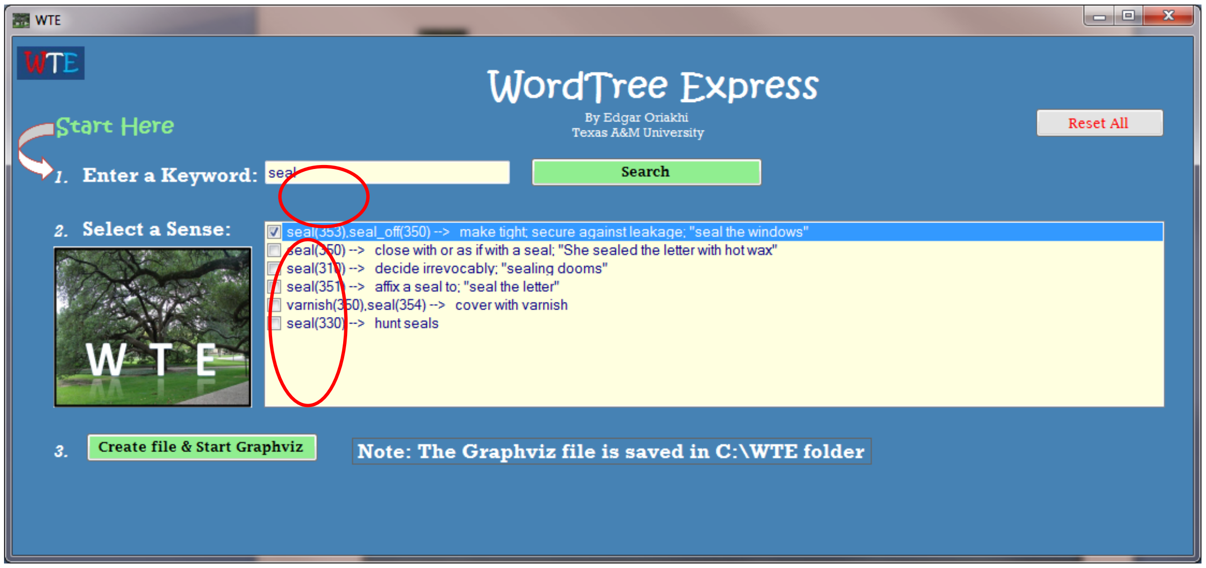
USING WTE

11. Double click the WordTree Express shortcut icon on your desktop.

(WordTree Express program icon as seen on the desktop)

12. Type a keyword in the textbox, click search and select a sense.

    The program should look like this:



WTE

WordTree Express

By Edgar Oriakhi
Texas A&M University

Start Here

1. Enter a Keyword: seal

Search

Reset All

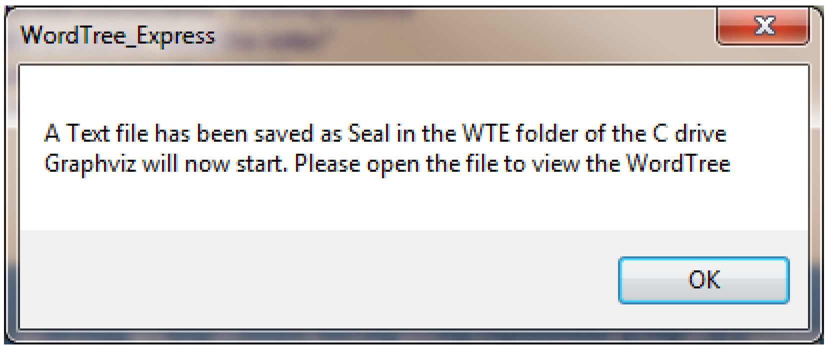
2. Select a Sense:

- [✓] seal(353),seal_off(350) --> make tight; secure against leakage; "seal the windows"
- [ ] seal(350) --> close with or as if with a seal; "She sealed the letter with hot wax"
- [ ] seal(310) --> decide irrevocably; "sealing dooms"
- [ ] seal(351) --> affix a seal to; "seal the letter"
- [ ] varnish(350),seal(354) --> cover with varnish
- [ ] seal(330) --> hunt seals

3. Create file & Start Graphviz

Note: The Graphviz file is saved in C:\WTE folder

13. Click on the "**Create file & Start Graphviz**" button to create a Graphviz file and start the Graphviz program. The following popup message will be displayed to let you know that a file has been saved as the keyword you typed in C:\WTE: Click on OK

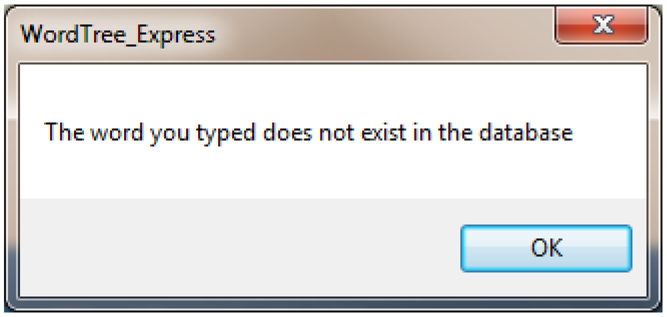


WordTree_Express

A Text file has been saved as Seal in the WTE folder of the C drive
Graphviz will now start. Please open the file to view the WordTree

OK

If the "Keyword" you typed doesn't exist you will see the following message:

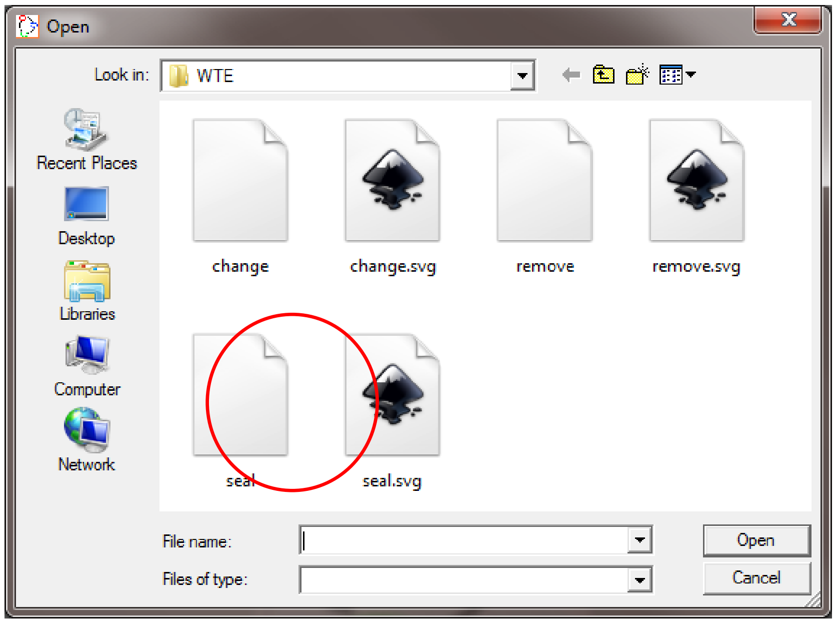


WordTree_Express

The word you typed does not exist in the database

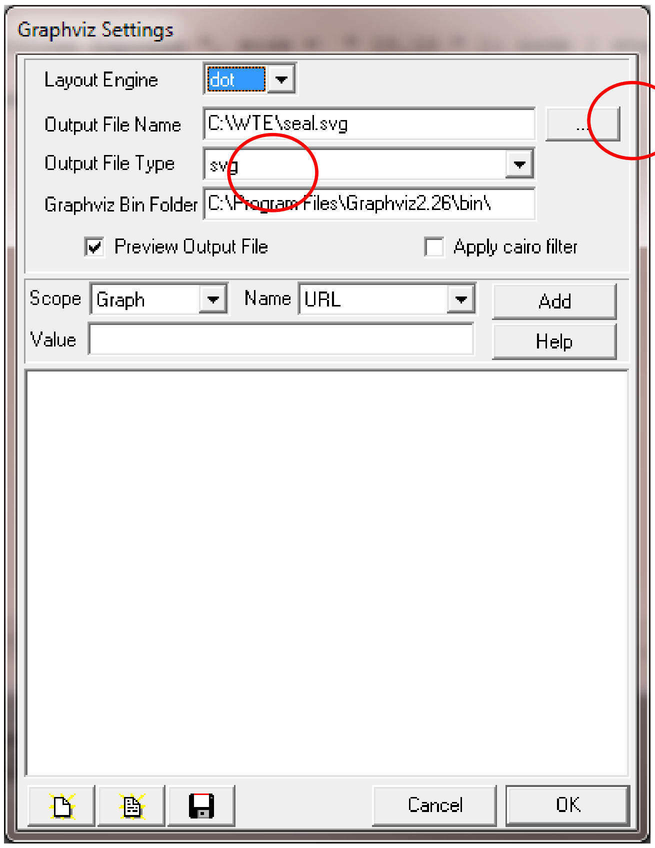
If you receive this error message you should check the spelling or type a different Keyword.

14. In the Graphviz program point to **file-->open** and select the created **Keyword** text file in C:\WTE.



15. Click on **run** and choose "**.svg**"(Scalable vector graphics) as the "**output file type**" in the popup window.

16. Click on the box next to "Output file name" to give a name to the file. The default file name is the keyword you typed in the search field (in this example "Seal").


Graphviz Settings

Layout Engine: dot

Output File Name: C:\WTE\seal.svg

Output File Type: svg

Graphviz Bin Folder: C:\Program Files\Graphviz2.26\bin\

[✓] Preview Output File

[ ] Apply cairo filter

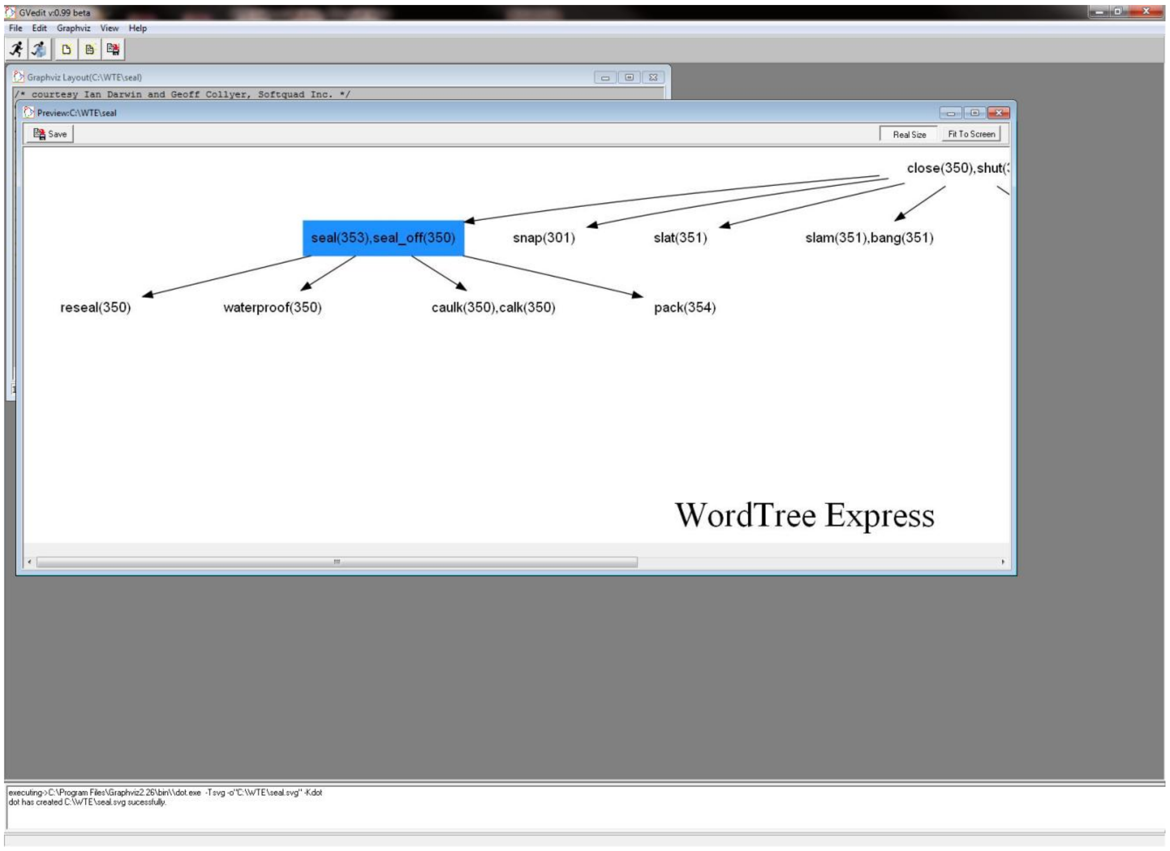
Scope: Graph

Name: URL

Value:

17. Next, **click OK** to save a scalable vector graphics version of the WordTree and to display the graphviz output as shown below.

Note: Every word on the WordTree has a numeric identifier used by the program for classification purposes and should be ignored when reading the output.



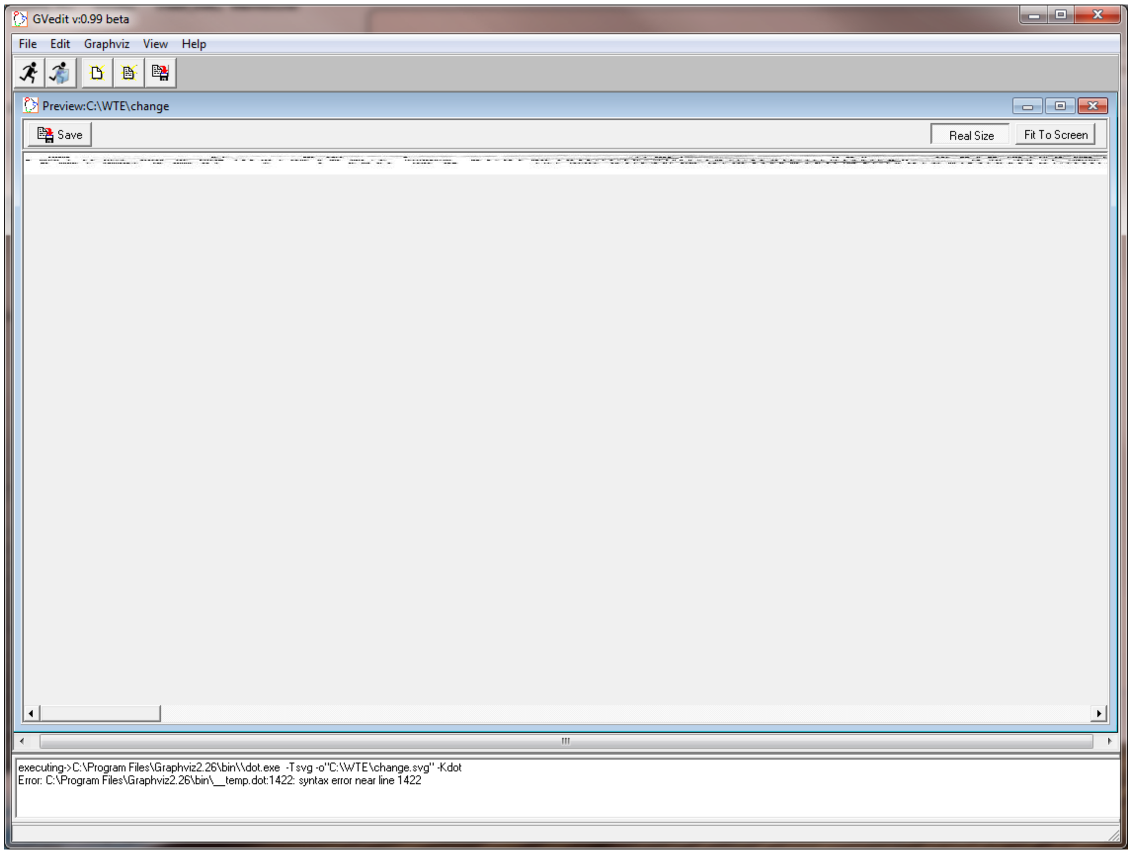
```
/* courtesy Ian Darwin and Geoff Collyer, Softquad Inc. */
```

```
executing->C:\Program Files\Graphviz2.26\bin\\dot.exe  -Tsvg -o"C:\WTE\seal.svg" -Kdot
dot has created C:\WTE\seal.svg sucessfully.
```

Graphviz WordTree output for the word "Seal"

18. In some cases (as shown below) the WordTree created is too large to be displayed using Graphviz, therefore a solution is to open the created ".svg" file

with Inkscape or Microsoft Visio. You should make Inkscape or Microsoft Visio the default ".svg" file handler. It is highly recommended to use Inkscape as it has a better zoom feature than Microsoft Visio.

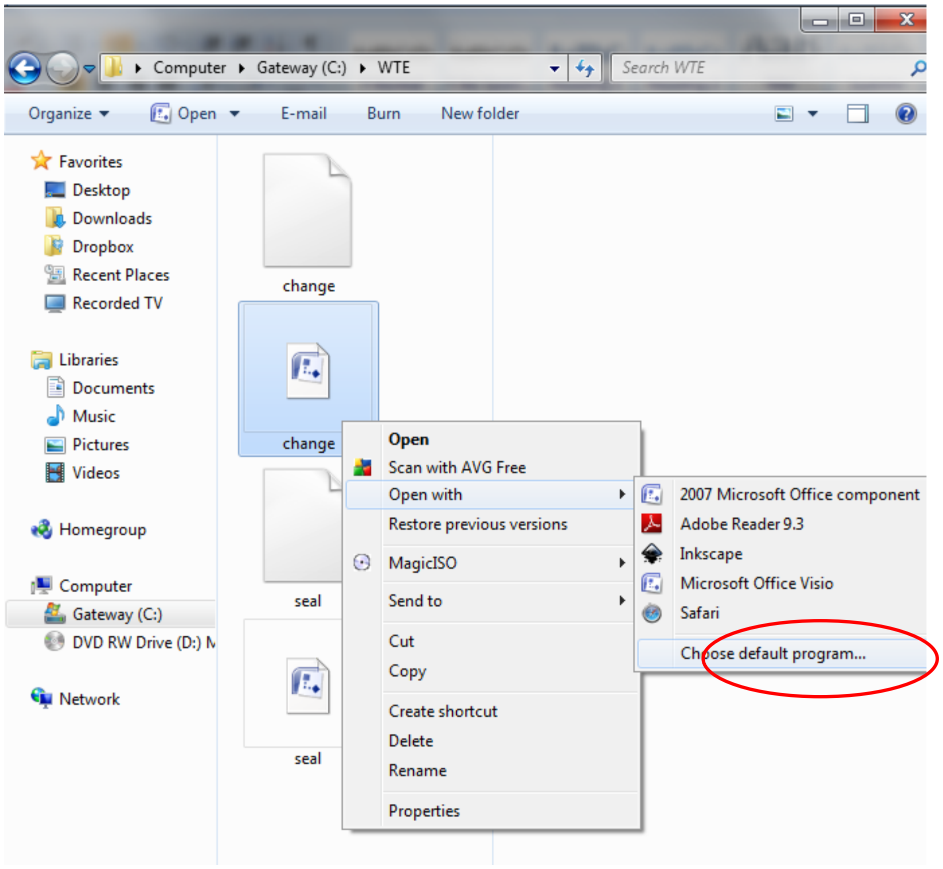


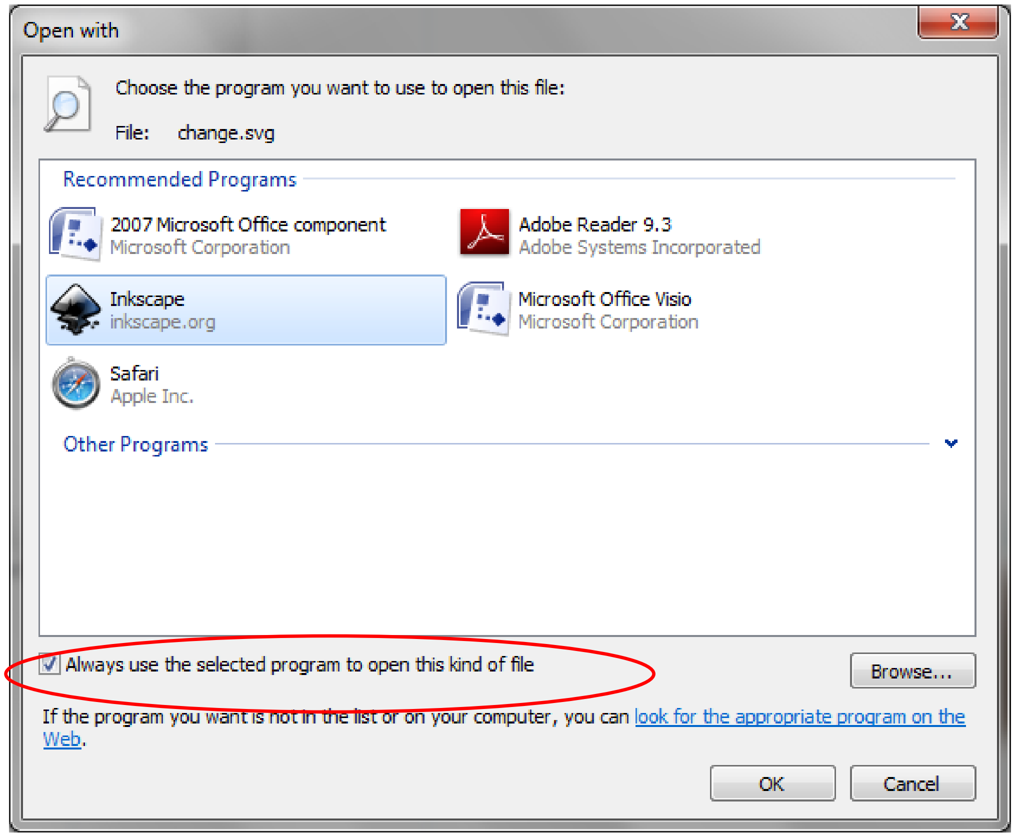
```
executing->C:\Program Files\Graphviz2.26\bin\\dot.exe  -Tsvg -o"C:\WTE\change.svg" -Kdot
Error: C:\Program Files\Graphviz2.26\bin\__temp.dot:1422: syntax error near line 1422
```

Graphviz WordTree output for the word "Change"

19. To make Inkscape or Microsoft Visio the default program for ".svg" files, right click on any the ".svg" files created as shown in the figure below and select "open with". Click "set default program" and brows program files to select

Inkscape or Microsoft Visio. Make sure the checkbox: "**Always use the selected program to open this kind of file**" is **checked**.





Open with

Choose the program you want to use to open this file:

File: change.svg

Recommended Programs

2007 Microsoft Office component
Microsoft Corporation

Adobe Reader 9.3
Adobe Systems Incorporated

Inkscape
inkscape.org

Microsoft Office Visio
Microsoft Corporation

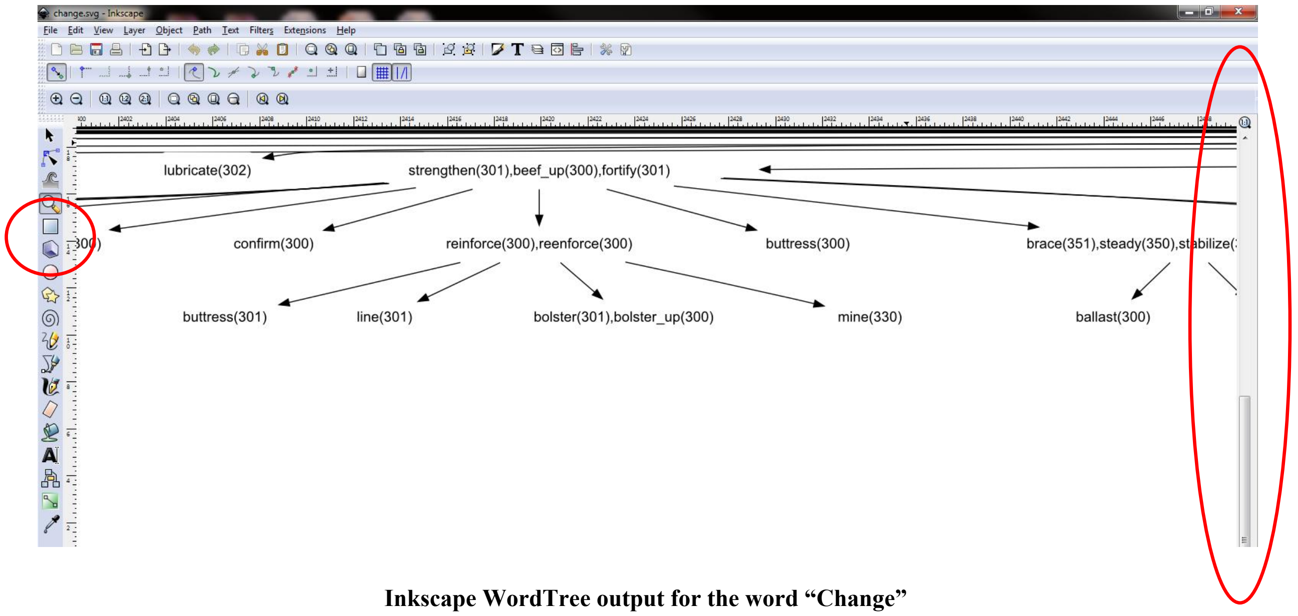
Safari
Apple Inc.

Other Programs

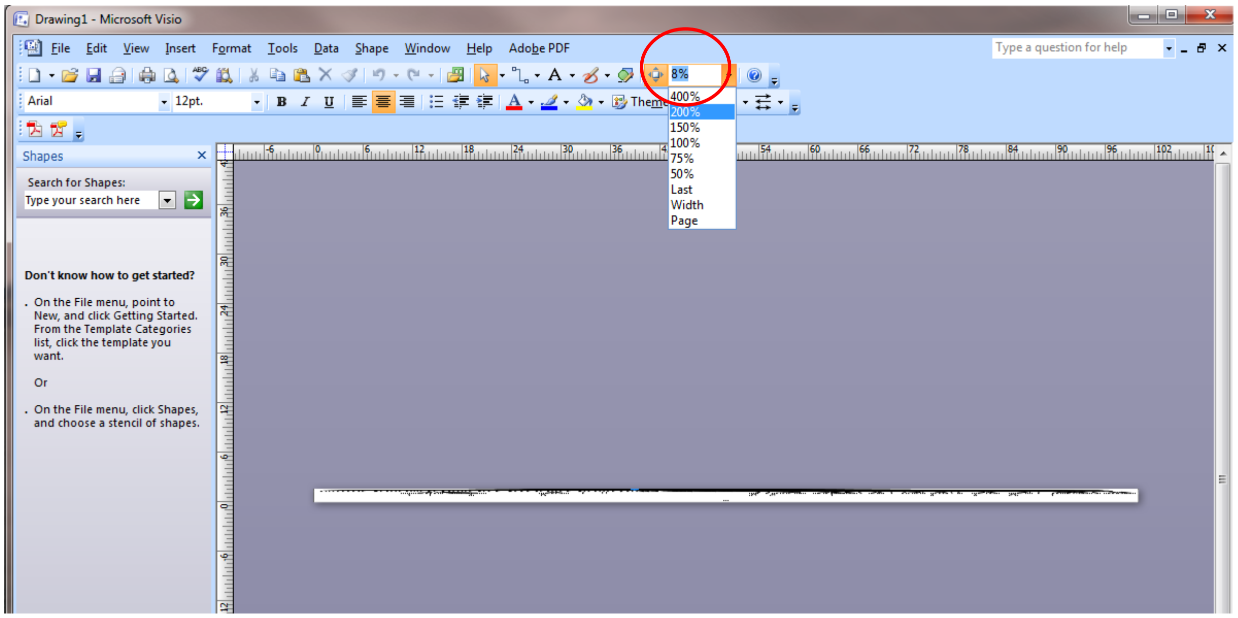
[✓] Always use the selected program to open this kind of file

If the program you want is not in the list or on your computer, you can look for the appropriate program on the Web.

20. Once a ".svg" file is open, use the **zoom** feature in Inkscape or Visio to adjust the graphical display. In **Inkscape** this is done by left clicking anywhere on the graph as shown below. To zoom out of the screen hold the shift key while clicking the graph area. In **Visio** adjust the zoom level in the drop down option as shown. Use the scroll bars to navigate the entire graph.


change.svg - Inkscape
File Edit View Layer Object Path Text Filters Extensions Help
lubricate(302)
strengthen(301),beef_up(300),fortify(301)
300)
confirm(300)
reinforce(300),reenforce(300)
buttress(300)
brace(351),steady(350),stabilize(
buttress(301)
line(301)
bolster(301),bolster_up(300)
mine(330)
ballast(300)

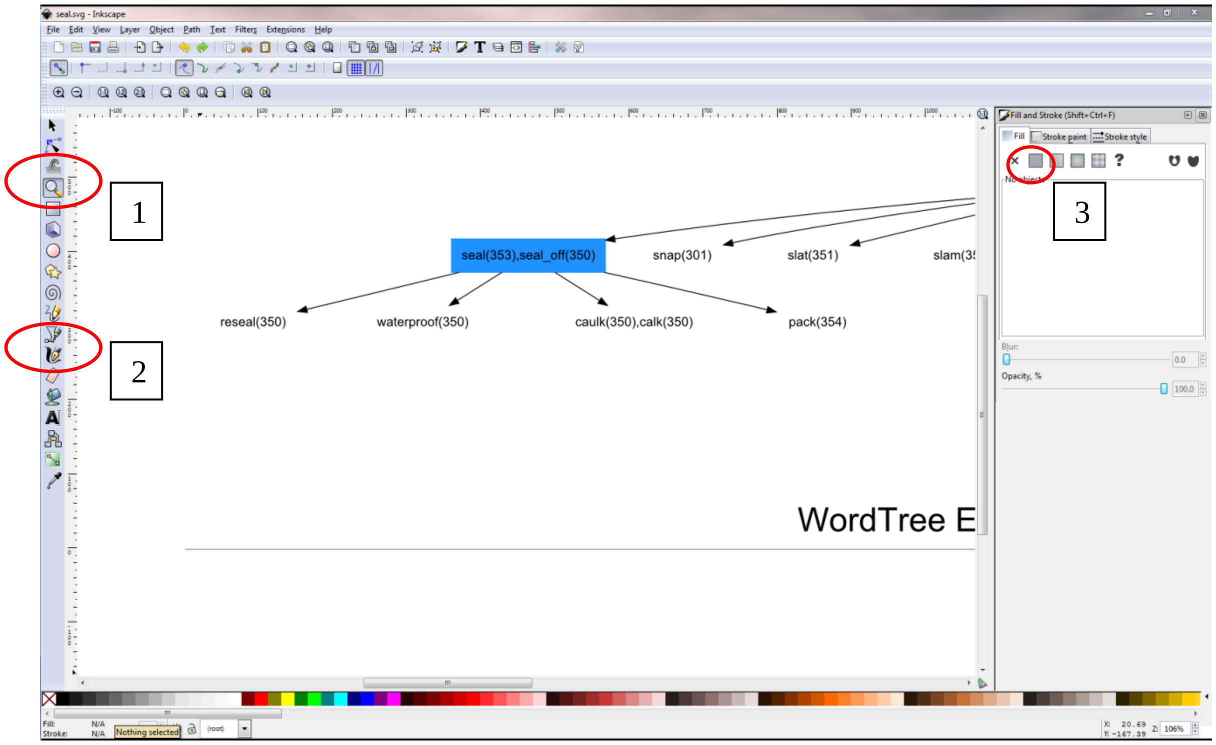
Inkscape WordTree output for the word "Change"

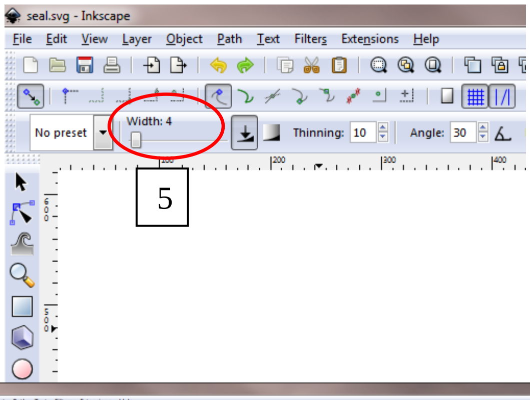

Drawing1 - Microsoft Visio
File Edit View Insert Format Tools Data Shape Window Help Adobe PDF
Type a question for help
8%
400%
200%
150%
100%
75%
50%
Last
Width
Page
Arial
12pt.
Shapes
Search for Shapes:
Type your search here
Don't know how to get started?
On the File menu, point to New, and click Getting Started. From the Template Categories list, click the template you want.
Or
On the File menu, click Shapes, and choose a stencil of shapes.

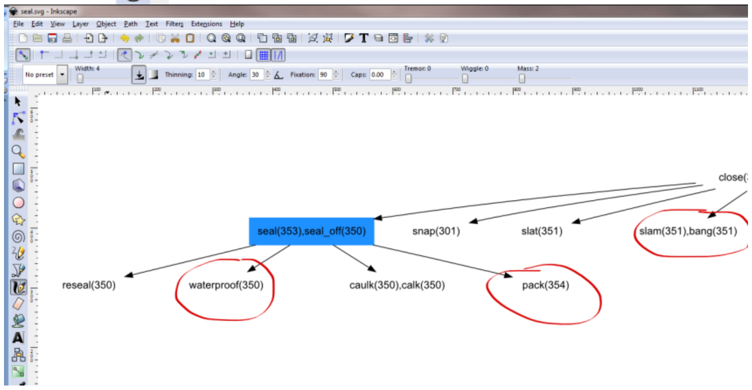
Visio WordTree output for the word "Remove"

21. To write on graph in **Inkscape**, you can use and edit the pen feature (shown on the next figure) by selecting the pen (1), double click on the fill option at the bottom left of the screen (2) and select the desired pen color (3). Adjust the

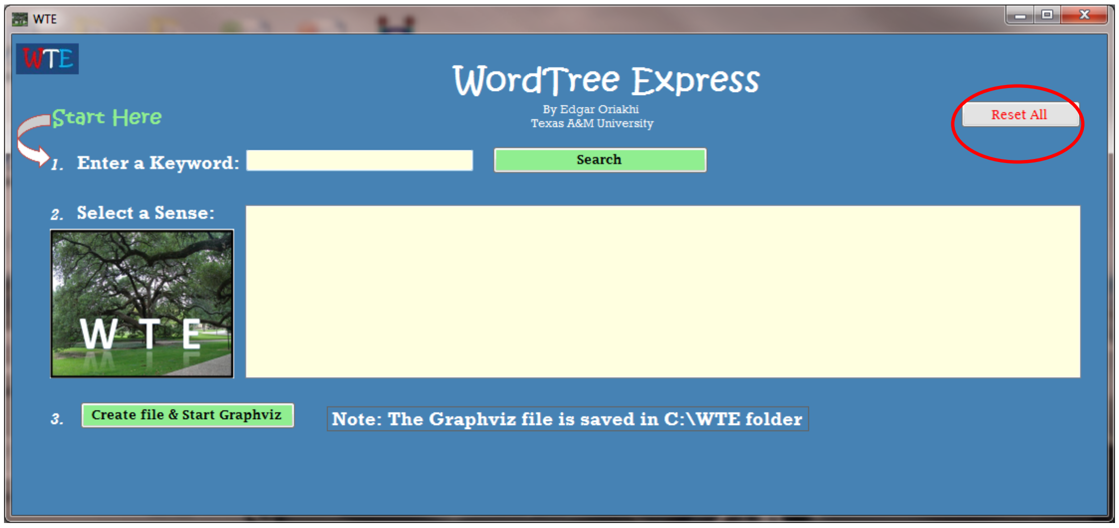
thickness of the pen stroke by adjusting the level as shown in the figure below (5).




seal.svg - Inkscape
File Edit View Layer Object Path Text Filters Extensions Help
No preset
Width: 4
Thinning: 10
Angle: 30
5



22. To perform a new search using WordTree Express, click on the "**Reset All**" button and type a new keyword.



WORDTREE EXPRESS PROGRAM CODE

```
Imports System.IO


Public Class Form1

    Private _arrWord As New ArrayList

    Private _arrMeaning As New ArrayList

    Private _arrSenses As New ArrayList

    Private _arrDefinition As New ArrayList

    Private _arrDefNum As New ArrayList

    Private _arrSensesW As New ArrayList


    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load

        Try

            funcLoadFiles()


        Catch ex As Exception


        End Try

    End Sub


    Private Sub funcSearch()

        Try

            File.Create("C:\WTE\" & txtSearch.Text & "")

            Dim arrDefined As New ArrayList

            Dim strSearch As String = txtSearch.Text

            Dim strMeaning, strWord As String
            Dim intRow, intNumRows, intPtrCnt, intNewCnt As Integer

            Do While (strSearch.IndexOf(Space(1)) >= 1)
                strSearch = strSearch.Replace(Space(1), "_") 'Replaces
spaces with "_" in the input.
            Loop

            intNumRows = _arrWord.Count - 1
            For intRow = 0 To intNumRows
                If strSearch.ToLower =
_arrWord(intRow).ToString.ToLower Then
                    Exit For
                End If
                If intRow = intNumRows Then
                    MsgBox("The word you typed does not exist in the
database")
                End If
            Next

            strMeaning = _arrMeaning(intRow)

            Dim strMean() As String
            strMean = strMeaning.Split(" ")
            intPtrCnt = CInt(strMean(3))
            _arrSenses.Clear()
            For intNewCnt = (intPtrCnt + 6) To strMean.Length - 1
                strWord = strMean(intNewCnt).Trim
                If String.IsNullOrEmpty(strWord) = False Then
                    _arrSenses.Add(strMean(intNewCnt))
                    Console.WriteLine(strMean(intNewCnt))
                End If
                Dim strNdef, strWordW As String
                Dim intWordCnt, intRowN, intNumRowsN, intNewCntN As
Integer
                Dim strDefinition As String
                intNumRowsN = _arrDefNum.Count - 1
                For intRowN = 0 To intNumRowsN
                    If strMean(intNewCnt) = _arrDefNum(intRowN) Then
                        Exit For
                    End If
                Next
                strNdef = _arrDefinition(intRowN)
                Dim strNWord() As String
                strNWord = strNdef.Split(" ")



                If strNWord(3) = "0a" Then
                    strNWord(3) = 10
                ElseIf strNWord(3) = "0b" Then
                    strNWord(3) = 11
                ElseIf strNWord(3) = "0c" Then
                    strNWord(3) = 12
                ElseIf strNWord(3) = "0d" Then
```

```
                    strNWord(3) = 13
                ElseIf strNWord(3) = "0e" Then
                    strNWord(3) = 14
                ElseIf strNWord(3) = "0f" Then
                    strNWord(3) = 15
                End If
                intWordCnt = CInt(strNWord(3))
                _arrSensesW.Clear()
                strWordW = Nothing
'<<<<<<<<<<<<<<<<<<<<<
                For i = 4 To (strNWord(3) * 2) + 2
                    strWordW += strNWord(i) & "(" & strNWord(1) &
strNWord(i + 1) & ")" & ","
                    i = i + 1
                Next
                strWordW = strWordW.Substring(0, strWordW.Length - 1)
                For intNewCntN = 0 To strNdef.Length - 1
                    If strNdef(intNewCntN) = "|" Then
                        strDefinition = strNdef.Substring(intNewCntN +
1)    '   <-----------------
                        CheckedListBox1.Items.Add(strWordW & " " & "--
>" & "   " & strDefinition)
                    End If
                Next

            Next
```

```
funcShowTree(CInt(_arrSenses(CheckedListBox1.SelectedIndex)))
        Catch ex As Exception

        End Try
    End Sub

    Private Sub funcShowTree(ByVal intSenseNum As Integer)

        Try

            Dim strRowValue(), strNWord(), strNWordb() As String
            Dim arrChildren, arrChildrenb, arrChildrend, arrChildrene,
arrChildrenf, arrChildrenbx, arrChildrencx, arrChildrendx,
arrChildrenex, arrChildrenfx, arrChildrenc, strChildren, strChildrenb,
strChildrend, strChildrene, strChildrenf, strChildrenbx, strChildrencx,
strChildrendx, strChildrenex, strChildrenc, arrParentb, strParentb As
New ArrayList
            Dim strMeaning, strWord As String
            Dim intSenseRow, intParent, intNumRows, intRow, intNumRowsN
As Integer
            Dim strNdef, strNdefb, strParent As String
            Dim intRowN As Integer

            For counter = 0 To 10
'Assuming no more than 10 levels up is a possibility
                intNumRowsN = _arrDefNum.Count - 1
                intNumRows = _arrDefNum.Count - 1
                For intSenseRow = 0 To intNumRows
                    If intSenseNum =
CInt(_arrDefNum(intSenseRow).ToString) Then
                        Exit For
                    End If
                Next
                strMeaning = _arrDefinition(intSenseRow)
                strRowValue = strMeaning.Split(" ")
                intNumRows = strRowValue.Length - 1

                For intRow = 0 To intNumRows
                    If strRowValue(intRow) = "@" Then
                        strWord = strRowValue(intRow + 1).Trim
                        If String.IsNullOrEmpty(strWord) = False Then
                            intParent = CInt(strWord)       '<----------
------
                            intSenseNum = intParent
                        End If
                    End If
                Next
                counter = counter + 1
            Next
            strParent = Nothing
            If intParent > Nothing Then
                For intRowN = 0 To intNumRowsN
                    If intParent = _arrDefNum(intRowN) Then
                        Exit For
                    End If
                Next
                strNdef = _arrDefinition(intRowN)
                strNWord = strNdef.Split(" ")

                If strNWord(3) = "0a" Then
                    strNWord(3) = 10
                ElseIf strNWord(3) = "0b" Then
                    strNWord(3) = 11
                ElseIf strNWord(3) = "0c" Then
                    strNWord(3) = 12
                ElseIf strNWord(3) = "0D" Then
                    strNWord(3) = 13
                ElseIf strNWord(3) = "0e" Then
                    strNWord(3) = 14
                ElseIf strNWord(3) = "0f" Then
                    strNWord(3) = 15
                End If
                strParent = Nothing
'<<<<<<<<<<<<<<<<<<<<<<
                For i = 4 To (strNWord(3) * 2) + 2
                    strParent += strNWord(i) & "(" & strNWord(1) &
strNWord(i + 1) & ")" & "," 'tttttttttttttttttttt
                    i = i + 1
                Next
```

```
                strParent = strParent.Substring(0, strParent.Length -
1)


            ElseIf intParent = Nothing Then
                strParent =
CheckedListBox1.SelectedItem.ToString.Split(" ")(0) 'txtSearch.Text
This tells it to use the selected checkbox keyword as the input keyword
            End If

            '///////////////////////////////////////2ND
LEVEL/////////////////////////////////////////////////////////////////
///////////////////////////

            intNumRowsN = _arrDefNum.Count - 1
            intNumRows = _arrDefNum.Count - 1
            For intSenseRow = 0 To intNumRows
                If intSenseNum = CInt(_arrDefNum(intSenseRow).ToString)
Then
                    Exit For
                End If
            Next

            strMeaning = _arrDefinition(intSenseRow)
            strRowValue = strMeaning.Split(" ")
            intNumRows = strRowValue.Length - 1
            For intRow = 0 To intNumRows
```

```
                If strRowValue(intRow) = "~" Then
                    strWord = strRowValue(intRow + 1).Trim
                    If String.IsNullOrEmpty(strWord) = False Then
                        arrChildren.Add(CInt(strWord))
                    End If

                End If
            Next

            For intRowN = 0 To arrChildren.Count - 1

                For intRow = 0 To intNumRowsN
                    If arrChildren(intRowN) =
CStr(_arrDefNum(intRow).ToString) Then
                        Exit For
                    End If
                Next

                strNdefb = _arrDefinition(intRow)
                strNWordb = strNdefb.Split(" ")

                If strNWordb(3) = "0a" Then
                    strNWordb(3) = 10
                ElseIf strNWordb(3) = "0b" Then
                    strNWordb(3) = 11
                ElseIf strNWordb(3) = "0c" Then
                    strNWordb(3) = 12
                ElseIf strNWordb(3) = "0D" Then
                    strNWordb(3) = 13
                ElseIf strNWordb(3) = "0e" Then
                    strNWordb(3) = 14
                ElseIf strNWordb(3) = "0f" Then
                    strNWordb(3) = 15
                End If
                Dim strWordX As String
'<<<<<<<<<<<<<<
                strWordX = Nothing
                For j = 4 To (strNWordb(3) * 2) + 2
                    strWordX += strNWordb(j) & "(" & strNWordb(1) &
strNWordb(j + 1) & ")" & "," 'ttttttttttttttttttttttt
                    j = j + 1
                Next
                strWordX = strWordX.Substring(0, strWordX.Length - 1)
                strChildren.Add(CStr(strWordX))


            Next

            TextBox3.Text = "/* courtesy Ian Darwin and Geoff Collyer,
Softquad Inc. */" & vbCrLf & "digraph unix {" & vbCrLf & "graph
[fontname = ""Sans"", fontsize = 36, label =  "" \n\n\n\nWordTree
Express "", size =  "" 10,10 "" ]; node [ color=white, fontname =
""Sans"" ]; " & vbCrLf & "size= "" 100,150 "";" & vbCrLf & """" &
CheckedListBox1.SelectedItem.ToString.Split(" ")(0) & """" & " " &
```

```
"[sides=4, color = dodgerblue, style = filled, fontname = ""Sans""];" &
vbCrLf

            For intRowN = 0 To arrChildren.Count - 1 'q
                TextBox3.Text += """" & strParent & """" & " " & "->" &
" " & """" & strChildren(intRowN) & """" & vbCrLf
            Next
            '///////////////////////////////////////////3RD
LEVEL///////////////////////////////////////////////////////////////////
/////////////////////////

            For intRowX = 0 To arrChildren.Count - 1
                intNumRowsN = _arrDefNum.Count - 1
                intNumRows = _arrDefNum.Count - 1

                For intSenseRow = 0 To intNumRows
                    If arrChildren(intRowX) =
CInt(_arrDefNum(intSenseRow).ToString) Then
                        Exit For
                    End If
                Next

                strMeaning = _arrDefinition(intSenseRow)
                strRowValue = strMeaning.Split(" ")
                intNumRows = strRowValue.Length - 1

                For intRow = 0 To intNumRows
                    If strRowValue(intRow) = "~" Then
                        strWord = strRowValue(intRow + 1).Trim
                        If String.IsNullOrEmpty(strWord) = False Then
                            arrChildrenb.Add(CInt(strWord))
                            arrChildrenbx.Add(CInt(strWord))
                        End If

                    End If
                Next
                If arrChildrenb Is Nothing = False Then
                    For intRowN = 0 To arrChildrenb.Count - 1

                        For intRow = 0 To intNumRowsN
                            If arrChildrenb(intRowN) =
CStr(_arrDefNum(intRow).ToString) Then

                                Exit For
                            End If
                        Next
                        strNdefb = _arrDefinition(intRow)
                        strNWordb = strNdefb.Split(" ")

                        If strNWordb(3) = "0a" Then
                            strNWordb(3) = 10
                        ElseIf strNWordb(3) = "0b" Then
                            strNWordb(3) = 11
                        ElseIf strNWordb(3) = "0c" Then
```

```
                            strNWordb(3) = 12
                        ElseIf strNWordb(3) = "0D" Then
                            strNWordb(3) = 13
                        ElseIf strNWordb(3) = "0e" Then
                            strNWordb(3) = 14
                        ElseIf strNWordb(3) = "0f" Then
                            strNWordb(3) = 15
                        End If
                        Dim strWordX2 As String
'<<<<<<<<<<<<<<
                        strWordX2 = Nothing
                        For j = 4 To (strNWordb(3) * 2) + 2
                            strWordX2 += strNWordb(j) & "(" &
strNWordb(1) & strNWordb(j + 1) & ")" & ","     'ttttttttttttttttt
                            j = j + 1
                        Next
                        strWordX2 = strWordX2.Substring(0,
strWordX2.Length - 1)
                        strChildrenb.Add(CStr(strWordX2))
                        strChildrenbx.Add(CStr(strWordX2))
                    Next


                End If
                For intRowN = 0 To arrChildrenb.Count - 1 'q
```

```
                    TextBox3.Text += """" & strChildren(intRowX) & """"
& " " & "->" & " " & """" & strChildrenb(intRowN) & """" & vbCrLf
Next
                arrChildrenb.Clear()                   '************
                strChildrenb.Clear()                   '************
            Next

            '/////////////////////////////////////////////4TH
LEVEL///////////////////////////////////////////////////////////////////
///////////////////////////

            For intRowZ = 0 To arrChildrenbx.Count - 1
                intNumRowsN = _arrDefNum.Count - 1
                intNumRows = _arrDefNum.Count - 1

                arrChildrenc.Clear()                   '************

                For intSenseRow = 0 To intNumRows
                    If arrChildrenbx(intRowZ) =
CInt(_arrDefNum(intSenseRow).ToString) Then
                        Exit For
                    End If
                Next

                strMeaning = _arrDefinition(intSenseRow)
                strRowValue = strMeaning.Split(" ")
                intNumRows = strRowValue.Length - 1
                For intRow = 0 To intNumRows
                    If strRowValue(intRow) = "~" Then
                        strWord = strRowValue(intRow + 1).Trim
                        If String.IsNullOrEmpty(strWord) = False Then
                            arrChildrenc.Add(CInt(strWord))
                            arrChildrencx.Add(CInt(strWord))
                        End If

                    End If
                Next

                If arrChildrenc Is Nothing = False Then

                    For intRowN = 0 To arrChildrenc.Count - 1

                        For intRow = 0 To intNumRowsN
                            If arrChildrenc(intRowN) =
CStr(_arrDefNum(intRow).ToString) Then

                                Exit For
                            End If
                        Next
                        strNdefb = _arrDefinition(intRow)
                        strNWordb = strNdefb.Split(" ")
                        If strNWordb(3) = "0a" Then
                            strNWordb(3) = 10
                        ElseIf strNWordb(3) = "0b" Then
                            strNWordb(3) = 11
                        ElseIf strNWordb(3) = "0c" Then
                            strNWordb(3) = 12
                        ElseIf strNWordb(3) = "0D" Then
                            strNWordb(3) = 13
                        ElseIf strNWordb(3) = "0e" Then
                            strNWordb(3) = 14
                        ElseIf strNWordb(3) = "0f" Then
                            strNWordb(3) = 15
                        End If

                        Dim strWordX3 As String
'<<<<<<<<<<<<<
                        strWordX3 = Nothing
                        For j = 4 To (strNWordb(3) * 2) + 2
                            strWordX3 += strNWordb(j) & "(" &
strNWordb(1) & strNWordb(j + 1) & ")" & ","     'tttttttttttttt
                            j = j + 1
                        Next
                        strWordX3 = strWordX3.Substring(0,
strWordX3.Length - 1)
                        strChildrenc.Add(CStr(strWordX3))
                        strChildrencx.Add(CStr(strWordX3))
                    Next


                End If
                For intRowJ = 0 To arrChildrenc.Count - 1 'q
                    TextBox3.Text += """" & strChildrenbx(intRowZ) &
"""" & " " & "->" & " " & """" & strChildrenc(intRowJ) & """" & vbCrLf
Next
                arrChildrenc.Clear()               '************
                strChildrenc.Clear()               '************
            Next


            '\\\\\\\\\\\\\\\\\\\\\\\\\\5th
LEVEL\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
\\\\\\\\\\\\\
            For intRowZ = 0 To arrChildrencx.Count - 1
                intNumRowsN = _arrDefNum.Count - 1
                intNumRows = _arrDefNum.Count - 1

                arrChildrend.Clear()               '************

                For intSenseRow = 0 To intNumRows
                    If arrChildrencx(intRowZ) =
CInt(_arrDefNum(intSenseRow).ToString) Then
                        Exit For
                    End If
                Next

                strMeaning = _arrDefinition(intSenseRow)
                strRowValue = strMeaning.Split(" ")
                intNumRows = strRowValue.Length - 1

                For intRow = 0 To intNumRows
                    If strRowValue(intRow) = "~" Then
                        strWord = strRowValue(intRow + 1).Trim
                        If String.IsNullOrEmpty(strWord) = False Then
                            arrChildrend.Add(CInt(strWord))
                            arrChildrendx.Add(CInt(strWord))
                        End If

                    End If
                Next

                If arrChildrend Is Nothing = False Then

                    For intRowN = 0 To arrChildrend.Count - 1

                        For intRow = 0 To intNumRowsN
                            If arrChildrend(intRowN) =
CStr(_arrDefNum(intRow).ToString) Then

                                Exit For
```

```
                                                End If
                                            Next
                                            strNdefb = _arrDefinition(intRow)
                                            strNWordb = strNdefb.Split(" ")

                                            If strNWordb(3) = "0a" Then
                                                strNWordb(3) = 10
                                            ElseIf strNWordb(3) = "0b" Then
                                                strNWordb(3) = 11
                                            ElseIf strNWordb(3) = "0c" Then
                                                strNWordb(3) = 12
                                            ElseIf strNWordb(3) = "0D" Then
                                                strNWordb(3) = 13
                                            ElseIf strNWordb(3) = "0e" Then
                                                strNWordb(3) = 14
                                            ElseIf strNWordb(3) = "0f" Then
                                                strNWordb(3) = 15
                                            End If

                                            Dim strWordX4 As String
'<<<<<<<<<<<<<<
                                            strWordX4 = Nothing
                                            For j = 4 To (strNWordb(3) * 2) + 2
                                                strWordX4 += strNWordb(j) & "(" &
strNWordb(1) & strNWordb(j + 1) & ")" & ","     'tttttttttttttt
                                                j = j + 1
                                            Next
```

```
                        strWordX4 = strWordX4.Substring(0,
strWordX4.Length - 1)
                        strChildrend.Add(CStr(strWordX4))
                        strChildrendx.Add(CStr(strWordX4))




                    Next


                End If
                For intRowJ = 0 To arrChildrend.Count - 1
                    TextBox3.Text += """" & strChildrencx(intRowZ) &
"""" & " " & "->" & " " & """" & strChildrend(intRowJ) & """" & vbCrLf
Next
                arrChildrend.Clear()               '************
                strChildrend.Clear()               '************
            Next
            '///////////////////////////6th
level///////////////////////////////////////////////////////////////
//////////////
            For intRowZ = 0 To arrChildrendx.Count - 1
                intNumRowsN = _arrDefNum.Count - 1
                intNumRows = _arrDefNum.Count - 1

                arrChildrene.Clear()               '************
                For intSenseRow = 0 To intNumRows
                    If arrChildrendx(intRowZ) =
CInt(_arrDefNum(intSenseRow).ToString) Then
                        Exit For
                    End If
                Next

                strMeaning = _arrDefinition(intSenseRow)
                strRowValue = strMeaning.Split(" ")
                intNumRows = strRowValue.Length - 1

                For intRow = 0 To intNumRows
                    If strRowValue(intRow) = "~" Then
                        strWord = strRowValue(intRow + 1).Trim
                        If String.IsNullOrEmpty(strWord) = False Then
                            arrChildrene.Add(CInt(strWord))
                            arrChildrenex.Add(CInt(strWord))
                        End If

                    End If
                Next

                If arrChildrene Is Nothing = False Then

                    For intRowN = 0 To arrChildrene.Count - 1
```

```
                        For intRow = 0 To intNumRowsN
                            If arrChildrene(intRowN) =
CStr(_arrDefNum(intRow).ToString) Then

                                Exit For
                            End If
                        Next
                        strNdefb = _arrDefinition(intRow)
                        strNWordb = strNdefb.Split(" ")

                        If strNWordb(3) = "0a" Then
                            strNWordb(3) = 10
                        ElseIf strNWordb(3) = "0b" Then
                            strNWordb(3) = 11
                        ElseIf strNWordb(3) = "0c" Then
                            strNWordb(3) = 12
                        ElseIf strNWordb(3) = "0D" Then
                            strNWordb(3) = 13
                        ElseIf strNWordb(3) = "0e" Then
                            strNWordb(3) = 14
                        ElseIf strNWordb(3) = "0f" Then
                            strNWordb(3) = 15
                        End If

                        Dim strWordX5 As String
'<<<<<<<<<<<<<<
                        strWordX5 = Nothing
                        For j = 4 To (strNWordb(3) * 2) + 2
                            strWordX5 += strNWordb(j) & "(" &
strNWordb(1) & strNWordb(j + 1) & ")" & ","     'tttttttttttttt
                            j = j + 1
                        Next
                        strWordX5 = strWordX5.Substring(0,
strWordX5.Length - 1)
                        strChildrene.Add(CStr(strWordX5))
                        strChildrenex.Add(CStr(strWordX5))




                    Next


                End If
                For intRowJ = 0 To arrChildrene.Count - 1
                    TextBox3.Text += """" & strChildrendx(intRowZ) &
"""" & " " & "->" & " " & """" & strChildrene(intRowJ) & """" & vbCrLf
Next
                arrChildrene.Clear()            '*************
                strChildrene.Clear()            '*************
            Next
            '///////////////////////////7th
level/////////////////////////////////////////////////////////////////////
//////////////
```

```
            For intRowZ = 0 To arrChildrenex.Count - 1
                intNumRowsN = _arrDefNum.Count - 1
                intNumRows = _arrDefNum.Count - 1

                arrChildrenf.Clear()                 '************

                For intSenseRow = 0 To intNumRows
                    If arrChildrenex(intRowZ) =
CInt(_arrDefNum(intSenseRow).ToString) Then
                        Exit For
                    End If
                Next

                strMeaning = _arrDefinition(intSenseRow)
                strRowValue = strMeaning.Split(" ")
                intNumRows = strRowValue.Length - 1

                For intRow = 0 To intNumRows
                    If strRowValue(intRow) = "~" Then
                        strWord = strRowValue(intRow + 1).Trim
                        If String.IsNullOrEmpty(strWord) = False Then
                            arrChildrenf.Add(CInt(strWord))
                            arrChildrenfx.Add(CInt(strWord))
                        End If

                    End If
                Next
                If arrChildrenf Is Nothing = False Then

                    For intRowN = 0 To arrChildrenf.Count - 1

                        For intRow = 0 To intNumRowsN
                            If arrChildrenf(intRowN) =
CStr(_arrDefNum(intRow).ToString) Then

                                Exit For
                            End If
                        Next
                        strNdefb = _arrDefinition(intRow)
                        strNWordb = strNdefb.Split(" ")

                        If strNWordb(3) = "0a" Then
                            strNWordb(3) = 10
                        ElseIf strNWordb(3) = "0b" Then
                            strNWordb(3) = 11
                        ElseIf strNWordb(3) = "0c" Then
                            strNWordb(3) = 12
                        ElseIf strNWordb(3) = "0D" Then
                            strNWordb(3) = 13
                        ElseIf strNWordb(3) = "0e" Then
                            strNWordb(3) = 14
                        ElseIf strNWordb(3) = "0f" Then
                            strNWordb(3) = 15
                        End If

                        Dim strWordX6 As String
'<<<<<<<<<<<<<<
                        strWordX6 = Nothing
                        For j = 4 To (strNWordb(3) * 2) + 2
                            strWordX6 += strNWordb(j) & "(" &
strNWordb(1) & strNWordb(j + 1) & ")" & ","     'tttttttttttttt
                            j = j + 1
                        Next
                        strWordX6 = strWordX6.Substring(0,
strWordX6.Length - 1)
                        strChildrenf.Add(CStr(strWordX6))




                    Next


                End If
                For intRowJ = 0 To arrChildrend.Count - 1
TextBox3.Text += """" & strChildrenex(intRowZ) & """" & " " & "->" & "
" & """" & strChildrenf(intRowJ) & """" & vbCrLf
Next
                arrChildrenf.Clear()               '************
                strChildrenf.Clear()               '************
```

```
            Next

'///////////////////////////6th/////////////////////////////////////
////////////////////////////////////////



'///////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////
////////
            TextBox3.Text += "}"

        Catch ex As Exception
        End Try
    End Sub

    Private Sub funcGraphViz()
        Dim x As Integer = CheckedListBox1.SelectedIndex
'**
        Try
            Dim FILE_NAME As String = "C:\WTE\" & txtSearch.Text & ""
            If System.IO.File.Exists(FILE_NAME) = True Then

                Dim objWriter As New System.IO.StreamWriter(FILE_NAME)
                objWriter.Write(TextBox3.Text)
                objWriter.Close()
                My.Computer.FileSystem.RenameFile("C:\WTE\" &
txtSearch.Text, txtSearch.Text & x + 1) ' **

                MsgBox("A new Text file named " & """" & txtSearch.Text
& x + 1 & """" & " has been created and saved in C:\WTE folder." &
ControlChars.NewLine & "Graphviz will now start. Please open the new
file to view the WordTree")
            Else
                MsgBox("File Does Not Exist")
            End If
        Catch ex As Exception
        End Try

    End Sub

    Private Sub ClearForm()
        For Each ctrl As Control In Me.Controls
            If TypeOf ctrl Is TextBox Then
                DirectCast(ctrl, TextBox).Text = String.Empty
            End If
        Next
        CheckedListBox1.Items.Clear()
    End Sub

    Private Sub funcLoadFiles()
        Try
```

```
            Dim sr As System.IO.StreamReader =
System.IO.File.OpenText("C:\WTE database\index.txt")
            Dim StrArray(), strLine() As String
            Dim intRow, intTotal As Integer
            Dim sData As String

            sData = sr.ReadToEnd
            StrArray = Split(sData, ControlChars.NewLine)
            'close stream
            sr.Close()

            intTotal = StrArray.Length - 1
            For intRow = 0 To intTotal
                strLine = StrArray(intRow).Split(" ")
                _arrWord.Add(strLine(0))
                _arrMeaning.Add(StrArray(intRow))
            Next

            sr = System.IO.File.OpenText("C:\WTE
database\database.txt")

            sData = sr.ReadToEnd
            'fill array with data
            StrArray = Split(sData, ControlChars.NewLine)
            'close stream
            sr.Close()
```

```
                intTotal = StrArray.Length - 1
                For intRow = 0 To intTotal
                    strLine = StrArray(intRow).Split(" ")
                    _arrDefNum.Add(strLine(0))
                    _arrDefinition.Add(StrArray(intRow))
                Next

            Catch ex As Exception

            End Try
        End Sub

        Private Sub btnGenerate_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles btnGenerate.Click
            funcSearch()
        End Sub

        Private Sub txtSearch_TextChanged(ByVal sender As System.Object,
ByVal e As System.EventArgs)

        End Sub

        Private Sub TextBox3_TextChanged(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles TextBox3.TextChanged

        End Sub
```

```
    Private Sub Label5_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Label5.Click

    End Sub

    Private Sub Label7_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs)

    End Sub


    Private Sub PictureBox1_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs)

    End Sub

    Private Sub CheckedListBox1_SelectedIndexChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
CheckedListBox1.SelectedIndexChanged

    End Sub

    Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button2.Click

        Cursor.Current = Cursors.WaitCursor
        funcShowTree(CInt(_arrSenses(CheckedListBox1.SelectedIndex)))

        Threading.Thread.Sleep(2000)

        funcGraphViz()

        Dim p As New System.Diagnostics.Process

        p.StartInfo.FileName = "Gvedit.exe"

        p.Start()


    End Sub


    Private Sub TextBox1_TextChanged(ByVal sender As System.Object,
ByVal e As System.EventArgs)


    End Sub


    Private Sub BtcClear_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles BtcClear.Click

        ClearForm()

    End Sub


    Private Sub Label11_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Label11.Click


    End Sub


    Private Sub PictureBox2_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles PictureBox2.Click
```

```
    End Sub

    Private Sub txtSearch_KeyDown(ByVal sender As System.Object, ByVal
e As System.Windows.Forms.KeyEventArgs) Handles txtSearch.KeyDown
        If e.KeyCode = Keys.Enter Then
            funcSearch()
        End If
    End Sub
End Class
```

VITA

Edgar Velazquez Oriakhi obtained his Bachelor of Science degree in mechanical engineering from Prairie View A&M University, Texas, in December 2008. He pursued his graduate studies at Texas A&M University in mechanical engineering from August 2009 and received his Master of Science degree in May 2011. His research interest includes design-by-analogy and innovation in design.

Permanent Address: Department of Mechanical Engineering, ENPH, 3123 TAMU, College Station, Texas 77843, USA.

Email: edgar780@yahoo.com