

Brigham Young University
BYU ScholarsArchive

Theses and Dissertations

2015-07-01

Terminological Mediation in Information Technology and Related Fields

Jessica Smith Richards
Brigham Young University - Provo

Follow this and additional works at: https://scholarsarchive.byu.edu/etd

 Part of the Industrial Technology Commons

BYU ScholarsArchive Citation

Richards, Jessica Smith, "Terminological Mediation in Information Technology and Related Fields" (2015). *Theses and Dissertations*. 5572.
https://scholarsarchive.byu.edu/etd/5572

This Thesis is brought to you for free and open access by BYU ScholarsArchive. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of BYU ScholarsArchive. For more information, please contact ellen_amatangelo@byu.edu.

Terminological Mediation in Information Technology and Related Fields

Jessica Smith Richards

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

Joseph J. Ekstrom, Chair
Kevin B. Tew
Derek L. Hansen

School of Technology

Brigham Young University

July 2015

Copyright © 2015 Jessica Smith Richards

All Rights Reserved

ABSTRACT

Terminological Mediation in
Information Technology
and Related Fields

Jessica Smith Richards
School of Technology, BYU
Master of Science

Terminological dissonance is created by the inherent ambiguity of natural language and compounded by ontological specialization efforts within fields. Terminological dissonance creates high-risk miscommunications in two key areas: within Information Technology as a singular domain, and also between IT and other fields in interdisciplinary projects. A comprehensive literature review revealed a lack of previous effort to acknowledge or solve problems of terminological dissonance within Information Technology.

This research provides a comprehensive overview and definition of the terminology mediation space as it relates to Information Technology and adjacent fields. An analysis and verification of the contents and implementation of the terminology mediation tool Termediator has also been created as part of this research. The Termediator tool's conceptual model is further validated through the analysis of its synonymous and polysemous clustering methods and results.

Keywords: terminology, ontology, communication, definition, glossary, language

TABLE OF CONTENTS

1 Introduction 1
1.1 Nature of the Problem 2
1.1.1 Terminological Dissonance 4
1.1.2 Need for Research 4
1.1.3 Focus of the Research 5
1.1.4 Summary of Introductory Material 5
1.2 Research Objectives 6
1.3 Definitions 6
2 Literature Review 8
2.1 Introduction 8
2.2 Terminology 8
2.3 Knowledge Bases 9
2.4 Domains 9
2.5 Conceptualizations of Knowledge Bases 10
2.5.1 Termbases 10
2.5.2 Glossaries 10
2.5.3 Taxonomies 11
2.5.4 Ontologies 11
2.6 Ambiguity in Terminology 13
2.7 Polysemy 14
2.8 Synonymy 16
2.9 Terminological Dissonance Example 17
2.10 Factors and Inevitability of Ambiguity 18
2.10.1 Ambiguity is Enjoyable 18

2.10.2 Ambiguity is Advantageous 19
2.10.3 Ambiguity Due to Natural Language Evolution 20
2.10.4 Ambiguity Due to Polysemy Preference 20
2.10.5 Ambiguity Due to Resistance to Terminological Change 22
2.11 Dissonance in Context 22
2.11.1 Dissonance in Single Domain Scope 22
2.11.2 Dissonance in Interdisciplinary Work 23
2.11.3 Dissonance in Text-Based Communication 24
2.12 Costs of Terminological Dissonance 25
2.12.1 Business Costs 25
2.12.2 Life and Death Costs 25
2.13 Ambiguity Removal 27
2.14 Denial of Ambiguity 27
2.14.1 Disambiguated Language Schemes 28
2.14.2 Bodies of Standardized Terms 29
2.14.3 Ontology Creation 29
2.14.4 Automated Ontology Processing 29
2.14.5 Top-Down Standardization 30
2.15 The Precision Problem 30
2.16 Terminology Mediation Tools 33
2.16.1 SDL MultiTerm 33
2.16.2 CRCTOL 33
2.16.3 Termediator 34
3 Methodology 35
3.1 Introduction 35
3.2 Define Terminology Mediation 35

3.3 Verify the Contents and Implementation of Termediator 36
3.4 Validate the Conceptual Polysemy Model Prototyped in Termediator 37
4 Termediator Documentation 40
4.1 Initial Prototype 40
4.2 Data Acquisition and Normalization 40
4.3 Polysemy Identification 46
4.3.1 Discovering the Candidate Threshold 47
4.4 Convergences 49
4.5 Examples of Synonymy Identification 52
4.6 Examples of Polysemy Identification 61
4.7 Multidisciplinary Dissonance Identification 67
4.7.1 IT + CS + GD 68
4.7.2 BPM + WF + ISYS + IT 69
4.7.3 Telecommuting 70
4.7.4 Education 71
5 Discussion 72
5.1 Future Developments for Termediator 72
5.1.1 Crowdsourced Data for Polysemy Candidate Threshold 72
5.1.2 Warning List Generation for Synonymy 74
5.1.3 Browsing for Polysemous Tools 75
5.1.4 Integration of Synonymy and Polysemy Interfaces 75
5.1.5 Integration of Synonymy and Polysemy Interfaces 76
5.1.6 Code Efficiency 78
5.1.7 Mobile Application 78
5.1.8 User Visibility of Polysemy Clustering Methods 78
5.1.9 Sophistication of Current Clustering Methods 79

5.2 Future Efforts in Terminological Mediation 81
5.2.1 Awareness of Terminological Dissonance 81
5.2.2 Awareness of Terminological Precision and Information Silos 81
5.2.3 Redirection of Effort Toward Mediation Tools 82
5.2.4 Inline Mediation Tools 82
6 Survey 83
6.1 Survey Format 83
6.2 Results 88
6.3 Survey Conclusions 90
7 Summary 91
7.1 Conclusion 96

LIST OF TABLES

Table 1. Survey Question Format 39

Table 2. Glossary XML Structure 41

Table 3. Basic Vector Creation 45

Table 4. Definitions of "Interface" in Multiple Domains 69

Table 5. Definitions of "Data" in Multiple Domains. 70

Table 6. Survey Results 89

LIST OF FIGURES

Figure 1. Polysemy Illustrated in Conversation 3

Figure 2. Spectrum of Ontology Structure 12

Figure 3. Diagram of Polysemy 14

Figure 4. Diagram of Synonymy 16

Figure 5. First Name Polysemy 21

Figure 6. One Potential Reason Why Lojban Did Not Gain Influence 28

Figure 7. Tunnel Diagram of Terminological Precision 31

Figure 8. Term, Concept, and Synonyms 43

Figure 9. Term, Concept, and Similarity Table 44

Figure 10. A Sample Dendrogram 48

Figure 11. Clusters in the Web Application 48

Figure 12. Cluster Convergences 49

Figure 13. Synonym Matches in "Abstraction" 53

Figure 14. Synonym Matches in "Acceptance Criteria" 54

Figure 15. Synonym Matches in "Agile Development" 55

Figure 16. Synonym Matches in "Help Desk Management" 55

Figure 17. Synonym Matches in "Information Processing" 56

Figure 18. Synonym Matches in "Source Code" 56

Figure 19. Synonym Matches in "Terminal" 57

Figure 20. Synonym Matches in "Trojan" 58

Figure 21. Synonym Matches in "Twisted Pair Cable" 59

Figure 22. Synonym Matches in "URL" 59

Figure 23. Synonym Matches in "User Interface" 60

Figure 24. Synonym Matches in "Help Desk Management" 60

Figure 25. Cluster Definitions 73

Figure 26. Manual Search 75

Figure 27. Diagram of Proposed UI 77

Figure 28. Overlapping Cluster Model 80

1 Introduction

The continual specialization of vocabulary has led to confusion and miscommunication within and between disciplines. This thesis refers to this problem overall as "terminological dissonance."

Miscommunication can easily break a team project, or even snowball into detrimental effects on an entire organization. One ambiguously worded message can be unwittingly passed on to a chain of colleagues, and soon enough an entire department takes away an unintended meaning. Products could be changed or manufactured the wrong way, entire orders cancelled, or contractors laid off—all because of a slightly ambiguous message.

Semantic miscommunication is usually more often discussed in linguistics than in Information Technology, however this topic has particular relevance to IT practitioners. As the tool supplier to many other disciplines, being in IT requires a high level of communicative adaptability in order to effectively work with people in adjacent fields. An IT professional could easily coordinate with a graphic designer, workflow manager, computer scientist, and systems engineer—all in the same day. This constant flow of interdisciplinary communication necessitates the understanding and management of communicative language.

1.1 Nature of the Problem

Consider a humorous example of potential miscommunication:

"When told to 'secure' a building it has been related that (Kasser, 2007):

- The Navy issues a purchase order for the building.
- The Air Force locks the doors and turns on the alarm system.
- The Army evacuates the personnel, then locks the doors and turns on the alarm system.
- The Marines assault the building using ground troops and air support, and then deploy squads in and around the building checking the credentials of all who aspire to enter the building."

In this example, the word "secure" was attached to different meanings. Technically speaking, "secure" is a "polysemous" term, and polysemy is one of the main causes of terminological dissonance. Although the example is clearly a joke, it illustrates how a familiar and common word holds drastically different meanings for different people.

A more sobering example of actual terminological dissonance can be seen in the tragedy of an airline crash:

> When Flight 52 arrived at Kennedy Airport, due to the fog and wind, only one runway was open for the 33 planes that were attempting to land every hour. Flight 52's fuel situation soon became desperate. Although they reported being low on fuel, the aircraft's crew did not explicitly declare that there was a "fuel emergency" to the local controllers, which would have indicated that the plane was actually in danger of crashing. The airplane was given a landing pattern that it had too little fuel to execute. The Boeing 707 slammed into the village of Cove Neck, Long Island, killing 65 of its 149 passengers and eight out of nine of its crew (Cushman, 1990).

In the previous example the traffic controllers had been trained that a "fuel emergency" meant that the plane was in danger of crashing. The pilots, on the other hand, thought that

repeatedly saying that they were very low on fuel indicated their dire situation. The pilots thought that “low on fuel” and “fuel emergency” were synonymous, and that one of the meanings of “low on fuel” was that the plane was in immediate danger. The traffic controllers, hearing the terms from the pilot, interpreted a different meaning. The end result was a death rate of nearly half of the passengers and all but one of the crew.

To solidify the understanding of terminological dissonance, it is helpful to see how polysemy plays out in an abstract conversation:

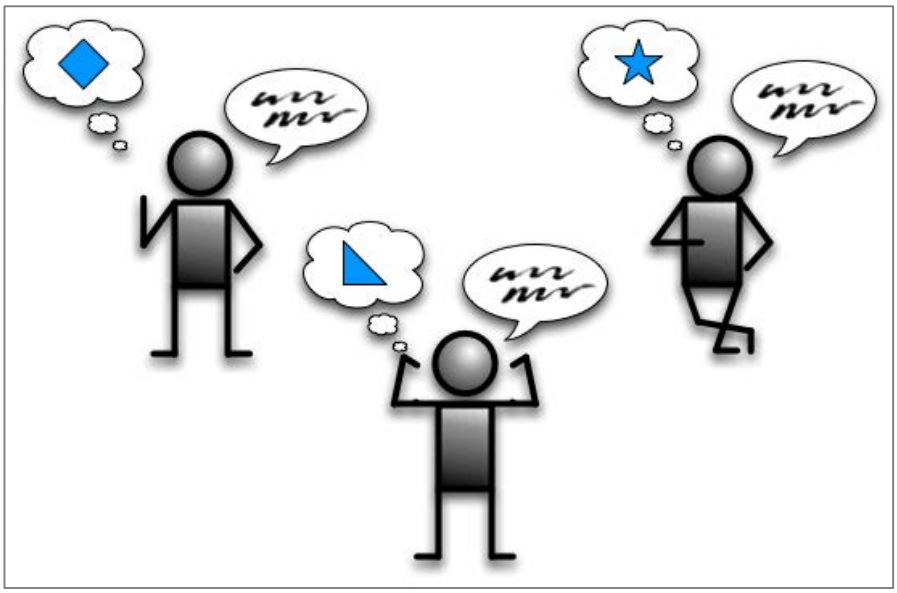


Figure 1. Polysemy Illustrated in Conversation

Every person speaks the same term, or words (indicated by the scribble marks in the figure), yet each person associates that same term with a different concept or meaning. Since the term is the sole representation of the concept in the conversation, the message syntax is received correctly but interpreted in three different ways dependent on the recipient’s stored definition of the term.

1.1.1 Terminological Dissonance

Terminological dissonance is miscommunication that occurs not because someone said the wrong thing, but because there are multiple "right" meanings for what is being said. This type of dissonance often occurs between two or more sympathetic, educated, and invested parties. As an analyst of behavioral communication so aptly put it, "Trouble develops when there really is no difference of opinion, when everyone is sincerely trying to get along…this is the type of miscommunication that drives people crazy. It is usually caused by a difference in conversational styles." (Smiley, 1986)

1.1.2 Need for Research

On the surface, the problem of terminological dissonance seems uncomplicated. Upon introduction of the problem, a typical response is that if we are clearer in our speech and writing that the problem is resolved.

Unfortunately that simple solution has proven ineffective due to *context discrepancies* caused by language ambiguity. When we speak, we automatically attach a term to a definition or image in our head. However, the image I see for "interface" may not be the same image you have for "interface." Term ambiguity is an inevitable feature of natural language that is especially magnified in interdisciplinary projects. "Interface" could mean any number of things to the graphic designer and to the systems engineer. In many brief workplace conversations, two people may easily discuss prospective changes to the "interface" without realizing that they are referring to two different things. Clarity is not the issue, context is.

1.1.3 Focus of the Research

The aspect of miscommunication this research focuses on is how key terms are attached to different concepts depending on the context. Context for term usage is made of up factors such as: field of study, type of project, applications used, personal experience, corporate training, place of education, team manager, and others. No one person has exactly the same context as another person when using a particular term. This context discrepancy causes "terminological dissonance" which in turn causes miscommunications that are not obviously identified.

1.1.4 Summary of Introductory Material

So far this thesis has illustrated a subtle communications problem. When one hears unknown words, such as in a foreign language, the failure to communicate is obvious. However, when one hears words that sound correct in a certain context, the failure to communicate is not realized and sometimes produces serious consequences. There is humor in miscommunication; however, it is a serious matter when project failure occurs because of a misunderstanding—especially when that misunderstanding could have been prevented.

As subtle miscommunications are not often recognized at the time they occur, it would be prudent to research solutions that prevent such miscommunications before they start, or at least streamline the recovery process. How can we prevent and troubleshoot terminological dissonance, and furthermore, how can we do this specifically in IT and through IT tools? This area of "terminology mediation" is relatively unexplored in IT and can benefit from serious research efforts. Development and validation of terminology meditation tools may pioneer a shift in how IT and related fields effectively handles terminological dissonance.

1.2 Research Objectives

This research attempted to accomplish the following objectives.

• O1. Describe and define "terminology mediation" as an area of research and "terminology mediation tools" in an Information Technology context.

• O2. Verify the contents and implementation of the Termediator tool.

• O3. Show that the Termediator tool generates a list of twenty polysemous terms that are more polysemous than a randomly generated list. Terms in both lists will be generated from the same set of Information Technology glossary data. The polysemy comparison is determined by user survey data.

1.3 Definitions

Understanding the following key terms will aid in the comprehension of this thesis. More detailed background information is provided in the literature review chapter.

Domain: This thesis uses domain to refer to a field of study in either a professional or academic context. For example Information Technology is one domain while Business Process Management is another.

Knowledge Base: A set of data committed to a conceptualization. For example, a glossary is a knowledge base that has a set of terms and concepts committed to it.

Ontology: A formal specification of a shared conceptualization. Typically the specification is of concepts and their relationships to each other. Although ontologies are one of the pillars of the Semantic Web, they do not have a universally accepted definition.

Polysemy: The coexistence of many possible meanings for a word or phrase.

Synonymy: Two or more terms having the same meaning.

Taxonomy: A hierarchical classification system.

Termediator: A software tool created to identify synonymy and polysemy from a compendium of terms and concepts.

Terminology: A group of specialized terms and concepts we use to communicate. Most fields of interests have their own set of terminology.

Terminological dissonance: Miscommunication that occurs between educated and sympathetic parties due to semantic discrepancies such as synonymy and polysemy. Social factors such as hostility and ignorance do not factor into terminological dissonance.

2 Literature Review

2.1 Introduction

A review of literature across multiple domains is conducted to accurately define the multidisciplinary problem of terminological dissonance. The various factors that create terminological dissonance are discussed. Tools that attempt to perform terminology meditation are introduced and their basic implementation is documented.

2.2 Terminology

Terminology is the group of specialized terms and concepts we use to communicate in intra- and inter- disciplinary projects. The following dictionary definitions illustrate the term in context:

> The system of terms belonging to or peculiar to a science, art, or specialized subject; nomenclature; *the terminology of botany* (Random House Dictionary 2014).

> The body of specialized words relating to a particular subject (Collins English Dictionary, Complete and Unabridged 10th Edition, 2009).

The philosopher Etienne Bonnot de Condillac observed that, "every science requires a special language because every science has its own ideas." He also noted that the natural evolution of terminology can be problematic for standardization efforts: "it seems that one ought

to begin by composing this language, but people begin by speaking and writing, and the language remains to be composed" (Condillac, 1776).

2.3 Knowledge Bases

A knowledge base is a set of data that is committed to a conceptualization. In the realm of terminology, these conceptualizations are typically glossaries, taxonomies, and ontologies. This is a surprisingly difficult idea to grasp at first. The idea of cupcakes and cupcake molds has been previously used to delineate the difference between a knowledge base and its conceptualization (Buitelaar, Cimmiano, & Magnini, 2005). The knowledge base is the collection of cupcakes of different sizes, colors, and flavors. The conceptualizations, such as an ontology or taxonomy, are the cupcake molds.

2.4 Domains

The word "domain" means different things to different areas of expertise. In this thesis, domain is used to simply refer to an area of study in either a professional or academic context. For example: Information Technology is one domain while Business Process Management is another.

The data used in this research comes from knowledge bases of terms and concepts such as glossaries and ontologies. Typically the authors of these knowledge bases self-declare the domain of their work. In fact, most glossaries are titled something very generic such as "Computer Science Terms" or "Systems Engineering Glossary". Even glossaries that are not titled in this manner typically state its author as an organization or group centered in a specific field of study.

With that in mind, the word “domain” is used throughout this thesis to loosely define the origin of glossary data as defined by that particular glossary’s author.

2.5 Conceptualizations of Knowledge Bases

2.5.1 Termbases

A termbase is a central repository of terms that allows the management of those terms (Wright & Budin, 2001). Termbases are primarily used in multilingual settings where terms and their concepts must be translated between languages. The termbase would allow management of those terms in both source and target languages. Besides the term itself, entries may contain any of the following information:

- An ID.
- Author.
- Concept or definition of the term.
- Creation and modification dates.
- Domain or subject area.
- Grammatical information.
- Source or context of the term.
- Notes.

2.5.2 Glossaries

A glossary, also known as a controlled vocabulary, is a list of terms in a field of study paired with corresponding definitions. All terms in a glossary ideally have clear and non-redundant definitions, although that ideal is often not upheld in practice. Some terminological

experts go on to state that a glossary must resolve ambiguities of polysemy through explicit name-qualifiers and synonymy by preferred term hierarchies (Pidcock, 2003). There are typically few rules that define glossary relationships beyond an associative relationship to each other; the most common categorization is the field of study where the term resides (e.g. terms are in a Graphic Design glossary because they are all commonly used by Graphic Designers).

2.5.3 Taxonomies

A taxonomy is a set of hierarchical relationships that conceptualizes a set of terms and concepts in a particular field of study. Each term is in one more parent-child relationships within the taxonomy. There may be different types of parent-child relationship in a single taxonomy, such as type-instance, genus-species, whole-part.

2.5.4 Ontologies

And lastly, ontologies—the most varied and potentially confusing terminological conceptualizations. Ontologies are specifications of concepts and their relationships to each other. These specifications are both formal and explicit, and traditionally ontologies veered toward a strict methodology that used axioms to validate and enforce constraints (Gruber, 1993). Typically the most formal ontologies were also natural language independent and did not contain lexical knowledge (Hjelm, 2009). Over time, however, an ontology spectrum has developed that ranges from the traditional "heavyweight" ontologies to more "lightweight" ontologies that do not use axioms at all (Uschold & Gruninger, 2004). It should be noted that the two other conceptualizations introduced in this paper, glossaries and taxonomies, are often considered a form of ontology. As illustrated in *Fig. 1*, basic glossaries lie at the most lightweight end of the

spectrum, strict taxonomies near the middle, and ontologies based on general logics lie at the most formal end (Wong, Liu, & Bennamoun, 2012).

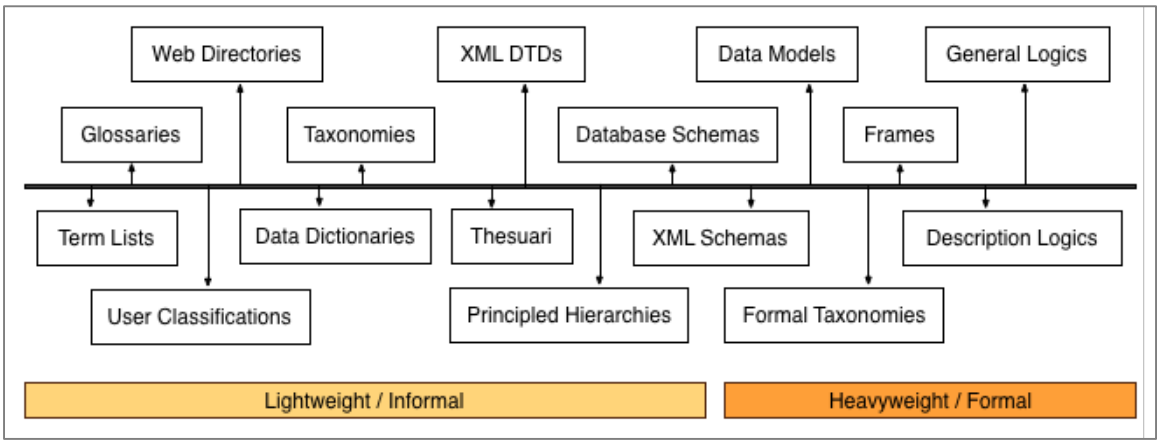


Figure 2. Spectrum of Ontology Structure

It is therefore important to remember that all of the conceptualizations are ontologies in the abstract sense. The variances between conceptualizations become clear when we determine where they lie on the spectrum of ontology structure.

The number of ontologies has grown exponentially over the years, and the reach of ontological research has spread from its roots in artificial intelligence labs to domain experts in a wide array of disciplines. We can see ontologies not only within esoteric academic circles, but in publicly available content on familiar websites. Ontologies are widely used in web applications for web directories (e.g. Google and Yahoo) or product classification (e.g. Amazon). Ontologies are also present in in Semantic Web standards such as RDF (Resource Description Framework) and Topic Maps (Fluit, Horst, Meer, Sabou, & Mika, 2003).

While we may interact with ontologies on a daily basis, the underlying point and purpose may not be entirely clear. It is important to define not only what ontologies are and what issues they face, but why they are created at all.

Some common reasons for ontology creation are (Noy & McGuiness, 2000):

- To analyze existing domain knowledge.
- To enable reuse of domain knowledge.
- To separate domain knowledge from operational knowledge.
- To explicitly define domain assumptions.
- To share the structure of domain knowledge among people or software agents.

Because ontologies center on documenting and defining domain knowledge, they are used especially in fields that possess a large quantity of specialized terms. This makes ontologies a central part of terminology in Information Technology and adjacent fields.

2.6 Ambiguity in Terminology

Within every ontological structure lies some level of terminological ambiguity. Recall that the purpose of ontologies revolves around the definition and use of knowledge within a domain. With the exception of ontologies on the most extremely formal end of the spectrum, most ontologies contain knowledge that is at least moderately natural language dependent. If we think of a typical glossary, the knowledge within comes in the form of natural language term-concept pairs, and therefore the meaning of the concepts can vary wildly depending on language, wording, context, and audience.

As the concepts themselves are subject to the nuances of natural language, they are subject to natural language ambiguity as well. Polysemy and synonymy are two extremely common forms of natural language ambiguity.

2.7 Polysemy

Many factors contribute to terminological miscommunication and the two main players are "polysemy" and "synonymy."

Polysemy is the potential for a term to have multiple meanings.

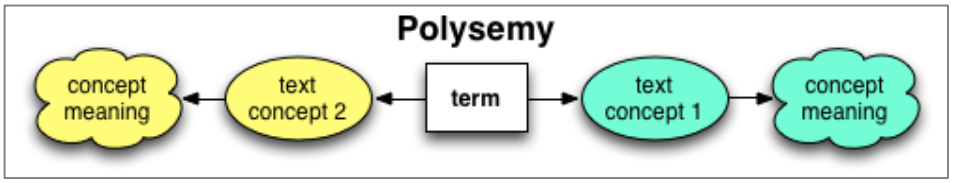


Figure 3. Diagram of Polysemy

The word "process" is a polysemous term. This term is very commonly used in a broad range of disciplines. Definitions of "process" are listed below in a selection of different domains. The key differences in each definition are highlighted.

Workflow Management

An *activity* that is part of a data flow diagram.

Information Security

In computer terms, a process refers to one of dozens of *programs* which are running to keep the computer running.

Information Systems

A process, in business terms, refers to *a series of linked tasks*, which together, result in a specified objective.

Information Technology

A *collection of resources* that enable the execution of program instructions. These resources can include virtual memory, I/O descriptors, a runtime stack, signal handlers, user and group IDs, and access control tokens. A more high-level view is that a process is a ``heavyweight" unit of execution with its own address space.

Software Engineering

An *executable unit* managed by an operating system scheduler.

Business Process Management

A set of *business tasks* designed to deliver value to an internal or external client. A process may be comprised of any combination of sub-processes and activities.

System Engineering

A set of interrelated or interacting activities which *transforms inputs into outputs*.

Computer Science

Any *operation* or combination of operations affecting data.

As you can see, the definition of “process” can vary quite a bit even in highly interrelated fields. Sometimes even the same discipline has different definitions of the same term dependent on project, company, or regional context.

2.8 Synonymy

Another type of miscommunication stems from synonymy. Terms that are synonymous share one or more similar concepts, or in other words, the same concept is linked to multiple terms.

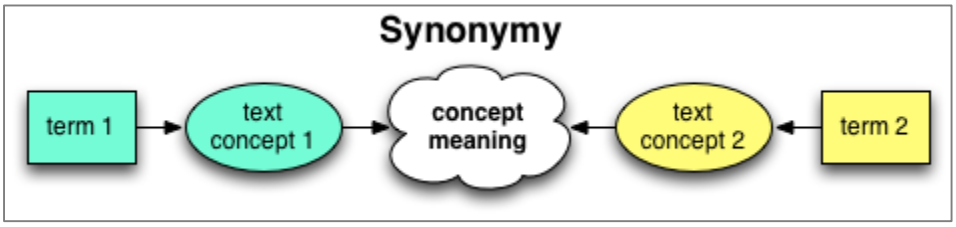


Figure 4. Diagram of Synonymy

One example of a synonymous pairing is “process” with the term “task”. Two definitions illustrating both concepts are listed below.

Task

A procedure that includes goals, steps, skills, start state, inputs, end state, and outputs to accomplish an activity.

Process

A process, in business terms, refers to a series of linked steps, inputs, and outputs, which together, result in a specified objective.

This example illustrates how synonymy can also be confusing when two terms refer to the same concept. It is especially problematic with a term such as “process” that is both polysemous and synonymous.

2.9 Terminological Dissonance Example

How dissonance occurs in an actual situation may not seem obvious on the surface. The following example may provide some insight.

The term "ATM" occurs in four communities that frequently interact: finance, technology, biology, and medicine.

ATM in finance: A computerized electronic machine that performs basic banking functions (as handling check deposits or issuing cash withdrawals). Also called automated teller machine (Nationwide Payment Solutions, 2010).

ATM in technology: The ITU standard for a cell-relay based communications system encompassing voice, data and video traffic. ATM provides standards for 25Mbps and 155Mbps transmission speeds. Because of the expense of the architecture, most networks do not handle this all the way to the workstation but larger networks will use it as a backbone. The unique function of this over other backbones other than speed is the self handled ability to prioritize traffic and requests (Computer Support Group, Inc., 2014).

ATM in biology: Ataxia telangiectasia mutated. A checkpoint kinase which transduces genomic stress signals to stop cell cycle progression and promote DNA repair, acting via p53, a tumour suppressor protein. Its cognate gene, ATM (see below), is mutated in ataxia telangiectasia, a rare neurodegenerative disease characterised by ataxia telangiectasias, increased chromosome fragility when exposed to ionising radiation and predisposition to lymphomas (Segen, 2005).

ATM in biology: A gene on chromosome 11q22-q23, which encodes a PI3/PI4 cell-cycle checkpoint kinase that phosphorylates, thereby regulating a broad range of downstream

proteins—e.g., tumor suppressor proteins p53 and BRCA1, checkpoint kinase CHK2, checkpoint proteins RAD17 and RAD9, and DNA repair protein NBS1 (Segen, 2005).

ATM in medicine: Atmosphere, atmospheric (Dictionaries, 2002).

Now, perhaps the financier will never have a conversation with the biologist that brings the conflicting definitions of ATM to light. However, an IT professional could easily encounter facets of medicine, biology, and finance just by contracting with one company. It is not entirely implausible that an IT professional may be required to set up an automated teller machine in a building that uses an asynchronous transfer mode network to communicate with others about their work on ataxia telangiectasia mutated.

2.10 Factors and Inevitability of Ambiguity

Now that we have noted that ambiguity is an inherent feature of natural language the full context of ambiguity must be reviewed in relation to IT terminological mediation.

Ambiguity matters because it prevents effective and efficient communication. Synonymy and polysemy are two forms of ambiguity. Ambiguity in all of its forms promotes terminological dissonance.

Current linguistic research supports the notion that ambiguity is largely unavoidable. The reasons for this conclusion are documented.

2.10.1 Ambiguity is Enjoyable

One of the major reasons that ambiguity will never be eradicated from natural language is that many people find the ambiguity enjoyable in of itself. Early traditional research into polysemy and other forms of linguistic ambiguity stated that such ambiguity is a “phenomenon

of the dictionary" or a purely "cognitive phenomenon." It was assumed that people strive for singular meanings in their use of terms, and that polysemy was an unintended result of misinterpretation or of uncontrolled dissemination of faulty definitions (Lehrer, 1990). However, more recent linguistic researchers have declared that polysemy is more than an unwanted artifact, and that ambiguity in discourse is often intentional as it possesses enjoyable and valuable communicational functions (Nerlich & Clarke, 2001). One example is how often polysemy is used as a form of humor, either as deliberate jokes or when one spontaneously "falls into a semantic trap." Some researchers have stated that it is the polysemy itself that both makes conversation interesting and keeps the development of language alive (Grice, 1975). If we truly disambiguate all of our terms then we may kill the enjoyment of language: consider that totally disambiguated language is how we talk to computers but not necessarily to people.

2.10.2 Ambiguity is Advantageous

In highly technological fields, language clarity is often prized as people in these fields must often "talk" to man-made machines without a truly sentient brain. Talking effectively to a computer requires speaking its language perfectly with singular meaning, as the machine has little to no capability to process polysemy on its own. As technicians, then, we often forget how ambiguity can be highly advantageous in people-centric fields. Advertising is just one area where the value of polysemy is not only being recognized but also specifically studied in context. Concepts such as "synchronic polysemy" and "diachronic polysemy" can dramatically affect an advertisement's efficacy as its message varies between recipients and also varies over repeated exposure (Puntoni, Schroeder, & Riston, 2010). Synchronic polysemy means that polysemy occurs simultaneously across two or more audiences. The same advertisement can be used to

target different cultures and demographics. Diachronic polysemy occurs when a layered advertisement delivers multiple meanings over time to the same audience. Repeated exposure enables the advertiser to advantageously retain the commercial's relevance through the deliverance of polysemy over time. Through these methods and others, advertisers often employ polysemy specifically in "strategic ambiguity." By strategically using polysemy, advertisers can: target multiple groups with one ad, increase an ad's influence over time with the same audience, or engage the audience via the enjoyment of language that polysemy provides. Strategic ambiguity is also a valuable skill for other professions such as political speech writers, literary authors, and slogan creators (Nerlich & Clarke, 2001).

2.10.3 Ambiguity Due to Natural Language Evolution

The development of language also has its place as a contributor to ambiguity. Although there are organizations that attempt to control language, their efforts at language purism have failed to make a significant dent in the sheer number of ways language is used ambiguously. Words are created, used, and repurposed constantly with different meanings attached. This naturally results in synonymy, polysemy, and varying patterns of use across the globe (Christiansen & Kirby, 2003).

2.10.4 Ambiguity Due to Polysemy Preference

A common phenomenon is the human preference for a small total vocabulary with polysemous terms over a large vocabulary with disambiguated terms. For example, we all know people who share the same common name, such as Mary or John, and can probably keep them

fairly straight. However, if a person with a unique name is introduced, we may be less likely to remember the new name.

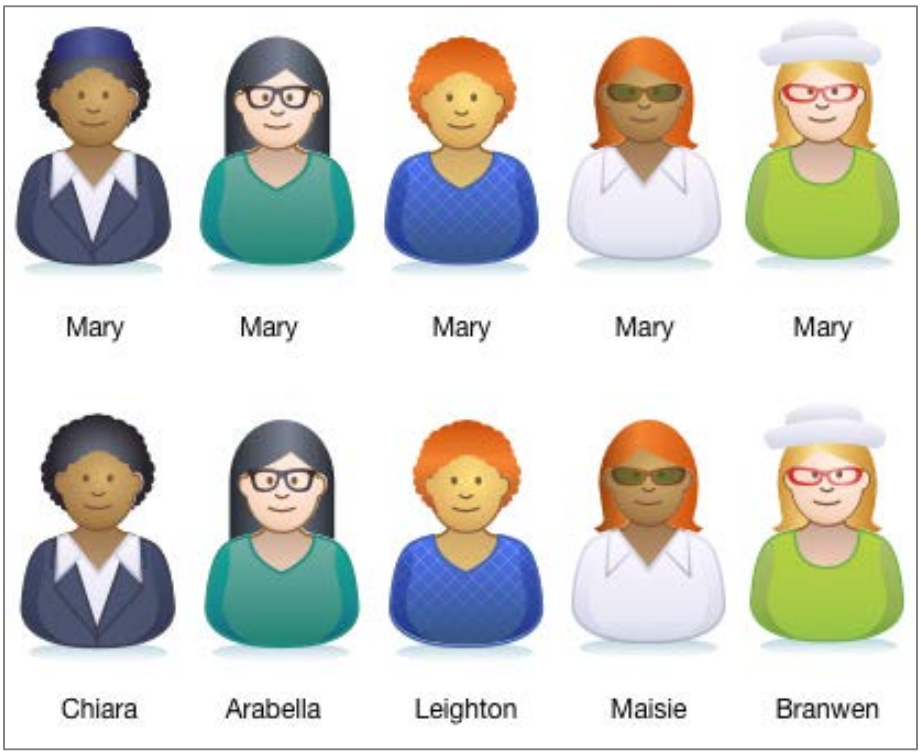


Figure 5. First Name Polysemy

Consider the two scenarios depicted in *Figure 5*. Is it easier to remember the names of the first group of people or the second? The polysemy of knowing five "Marys" and "Johns" may be preferable to the unambiguous task of remembering a high number of unique names (Klepousniotou & Baum, 2007). This preference for polysemy may bleed over into technical fields where the definitions of "design," "system," and "constraint" completely depend on context.

2.10.5 Ambiguity Due to Resistance to Terminological Change

Retroactive efforts to change, divide, or otherwise standardize synonymous or polysemous terms are often unsuccessful. Although the introduction of a new term or a new meaning can be manipulated by those presenting it, once the ambiguity comes into play it is very hard to remove. The “accepted” use of concepts tends to develop bottom-up, with the practitioners setting the standard over time, and top-down standardization efforts are rarely implemented effectively by the relevant community at large (Gong, Shuai, & Comrie, 2014).

2.11 Dissonance in Context

The awareness of ambiguity and its negative product, terminological dissonance, is sorely lacking. Although it is not a solution in of itself, ambiguity awareness training is a noble effort. It is important that professionals in all spheres understand the effects of terminological ambiguity—especially in the hard sciences where "soft" studies on language and communication may not be emphasized. The examples below illustrate the current state of ambiguity as it relates to terminological dissonance in several contexts.

2.11.1 Dissonance in Single Domain Scope

A very common idea is that each field can simply specialize terminology to suit their own needs with little consequence. However, research on the findings of Termediator, a terminology mediation tool, has found that even within one domain there exists significant terminological dissonance (Richards, Riley, Ekstrom, & Tew, 2013). The Termediator data set reveals that even within one area of study there exist multiple glossaries and other knowledge bases that attempt to

re-define the same set of terms. These terminological conflicts potentially create miscommunication within and between same-discipline organizations, projects, and teams.

2.11.2 Dissonance in Interdisciplinary Work

Interdisciplinarity is on the rise both professionally and academically. Examples can be seen in organizations across the globe. Harvard Law Today reports that the boundaries have blurred between Harvard Law School and Harvard Business School, with more co-hosted classes, symposiums, conferences, and curricula (Subramanian & Minow, 2013). Researchers commissioned by the Swedish Council for Research and Planning on knowledge production in universities have made similar observations: they state that we have moved beyond disciplined-based studies to a transdisciplinary state (Hessels & Lente, 2008). Advisors to the Department of Health and Human Services have determined that the increasingly interprofessional state of healthcare is not only inevitable but also necessary (Department of Health and Human Services, 2012).

In a previous era, it may not have mattered if two disciplines had differing definitions of the same term. Their contact with each other would have been brief and inconsequential. In today's interdisciplinary era, however, the effects of terminological dissonance are substantial. If a web developer and a graphic designer work together on a website, differences in terminology such as "interface" can significantly hinder or negatively alter a project. The consequences become more severe in collaborations that pair technologists and the caretakers of life and death; medical systems, traffic control, emergency response, and disaster recovery are just few of the fields that use technology heavily. Medical practitioners and facilities have more than doubled their use of IT between years 2012 and 2013 (Department of Health and Human Services, 2013).

Failures to communicate between the designers and the users of critical systems can result in catastrophic failures. These communication failures can be as simple as the previously discussed issues of synonymy and polysemy, where different terms refer to the same concept, or one term is overloaded with multiple meanings.

Even before the advent of global interdisciplinarity, there have always been professions that thrive on interdisciplinary work. By definition, service-providing fields such as Information Technology sit at the borders of several other disciplines. How do you properly deliver a service to multiple audiences when errors in transmission are summarily ignored? Ignoring ambiguity overall is not an option when certain disciplines have to transverse disciplinary boundaries just to complete basic job tasks. The general increase in interdisciplinary and multidisciplinary work worldwide attests to the fact that terminological ambiguity will become more of a problem in the future, not less. Ambiguity awareness must be a top priority for interdisciplinary fields. As one analyst eloquently summarized:

> In business, miscommunication can be fatally damaging. Technology makes communication fast and loud. A company's culture and its customers' goodwill can shift quickly. Is it any wonder that some of the world's leading businesses invest considerable resources developing better communication skills in their people? (Lundrigan, 2013)

2.11.3 Dissonance in Text-Based Communication

It is not only interdisciplinary work that increases the number and intensity of miscommunication in the workplace. Email use has risen, along with other forms of text-based communication such as social media messages and phone texts (Legatt, 2011). Higher rates of communication via text—whether that is email, phone text, or online messaging—dramatically increases the total amount of miscommunications and conflict escalations (Byron, 2006).

2.12 Costs of Terminological Dissonance

2.12.1 Business Costs

Terminological dissonance not only wastes time and risks project failure, but the miscommunications resulting from it are quite the rising business expense. The estimated total cost of employee misunderstanding? $37 billion annually across the US and UK. The research indicates that only one in three organizations *claims* to have done *anything* to close the gap between understanding and misunderstanding within their own company. This in spite of the fact that 99% of companies surveyed reported that employee misunderstanding had placed the company at risk of injuries to the public, hurt sales and profits, and reduced customer satisfaction (Cognisco, 2008).

2.12.2 Life and Death Costs

The scope of terminological ambiguity can be both overwhelming and intimidating. It is a topic not typically addressed between technologists; a brilliant computer scientist may quickly blanch at using their logical expertise to solve problems in linguistics. Although the problem is definitely large, the potential harm caused by miscommunication is so great that it simply cannot be ignored. The following situations illustrate how the proper management of terminology became a matter of life and death:

A military investigation has determined that miscommunication between U.S. air and ground forces led to the death of five U.S. soldiers. The report cites a collective failure by soldiers, commanders and air crew members that resulted in the death of five Americans and one Afghan who were mistaken for the enemy and attacked with two laser-guided bombs from a B1 bomber (Time Warner Communications, 2014).

The ambulance arrived 46 minutes after Myrna first called 911. Burt went into cardiac arrest on the way to the hospital and died that day in the emergency room. The delay in response appears to have been caused by a chain of misunderstandings, miscommunication and technical glitches. (Kaufmann, 2014).

When Flight 52 arrived at Kennedy Airport, due to the fog and wind, only one runway was open for the 33 planes that were attempting to land every hour. Flight 52's fuel situation soon became desperate. Although they reported being low on fuel, the aircraft's crew did not explicitly declare that there was a "fuel emergency" to the local controllers, which would have indicated that the plane was actually in danger of crashing. The airplane was given a landing pattern that it had too little fuel to execute. The Boeing 707 slammed into the village of Cove Neck, Long Island, killing 65 of its 149 passengers and eight out of nine of its crew (Cushman, 1990).

Medical errors may be the third leading cause of death in the United States. A multi-organizational study led by the University of San Francisco found that the transfer of medical data (and subsequent miscommunications) between departments and practitioners was a key factor in preventable deaths and other adverse events due to medical error (Starmer, et al., 2014).

A study by the American College of Surgeons found that roughly 20 percent of surgical malpractice claims were filed due to miscommunication errors. Of the 460 claims analyzed, 36 were due to miscommunications between patients and their family members, 35 were between patients and doctors, and 19 were between patients and nurses (Griffen, et al., 2007).

2.13 Ambiguity Removal

The problems of terminological ambiguity are not new, and several solutions have been previously proposed and implemented. Once the ambiguity problem has been sufficiently acknowledged by a particular community, simply ignoring it becomes impossible. The next natural response to the problem is to "solve" it by removing ambiguity as much as possible. The processes by which ambiguity is removed from terminology, as well as how those processes do not solve the problem of terminological dissonance, are documented. Ultimately these solutions have failed in one way or another, which led us to conduct further research on the problem from new perspectives.

2.14 Denial of Ambiguity

Polysemy, synonymy, and other forms of ambiguity can and do create serious miscommunication problems in the workplace and elsewhere. However, the standard project manager has no idea how to effectively tackle the problem—assuming he has really thought about the problem at all. "Ambiguity training" is not a part of most management curricula nor is it a commonly discussed except among linguistic experts and aficionados.

We would suggest that most managers take the default option when it comes to ambiguity: they press "ignore." Even those who have experienced significant roadblocks because of miscommunication may apply a "live and let live" philosophy, assuming that conflicts are the exception are than the rule. Yet this approach actually increases the chance that future crises will occur: the most frequent miscommunications occur in situations where no linguistic risk is perceived (Condamines, 2010).

2.14.1 Disambiguated Language Schemes

The reasons previously detailed give us a sense for how ambiguity is unavoidable in many natural language situations. This inevitability even prompted the Logical Language Group to construct a new language, Lojban, based entirely on mathematical logic based on James Cook Brown's research some twenty years earlier (Goertzel, 2013). Although the "Loglan / Lojban" project is beyond the scope of this article, suffice to say that one of the goals was to completely avoid synonymy and polysemy in terms.

While the project has achieved moderate success within a small sphere of influence, few of us can just start speaking Lojban in our professional sphere. An unambiguous constructed language in the minority of use is simply not a practical solution for those of us who must work with the linguistic majorities in the world. *Figure 6* (Munroe, 2015) illustrates the humor in the situation. We are then left to find another method by which we can work *with* natural language to deal with its inherent terminological ambiguity.



Figure 6. One Potential Reason Why Lojban Did Not Gain Influence

2.14.2 Bodies of Standardized Terms

One of the most common ways to disambiguate terminology is to create standardized works such as glossaries, ontologies, and taxonomies. The generalized structure of each form was detailed in the introduction. While the data in these bodies of knowledge can be utilized, the glossaries themselves do not resolve or remove ambiguity in the "big picture." The most a glossary can do is moderately remove ambiguity within small groups that are committed to adhering to that particular glossary's definitions at all times (Cargill, 2011).

2.14.3 Ontology Creation

One obstacle these knowledge bases face is the sheer amount of work it requires to create one in the first place. The most formal and standardized knowledge bases, such as ontologies, often standardize a great deal of terminology in a methodical and consistent manner. This gives them the potential to disambiguate terminology in a given field. However ontologies are extremely complicated, and difficult to write, compare, and maintain (Miller L. , 2000). Some have even characterized ontologies as tediously handcrafted sources of information (Wong, Liu, & Bennamoun, 2012). This not only makes good ontologies relatively rare, but it also makes it difficult for ontologies to stay relevant. If an ontology cannot keep up with new terminology in a changing world, it will quickly become obsolete.

2.14.4 Automated Ontology Processing

In an attempt to increase the influence of ontological works, many researchers have focused on automated ontological processing and learning. There are many knowledge-based applications that rely on the automated input of domain ontologies. Disambiguation in this

processing is a key step—it is extremely difficult for a knowledge-based application to make decisions based on ambiguous natural language. Ontology processing methods range from simple text mining to complicated word sense disambiguation algorithms. Word sense disambiguation (WSD), is a computation linguistic approach that attempts to precisely select the appropriate meaning of a term for a context. WSD is seen by some as a holy grail to solve the terminological ambiguity problem. However, the fact remains that in over forty years of research, WSD has failed to prove itself as the final solution. At this time there is no algorithm capable of disambiguating what human knowledge has accumulated (Quiroga-Clare, 2003).

2.14.5 Top-Down Standardization

Another obstacle faced is top-down standardization versus bottom-up implementation. When a glossary, taxonomy, or ontology is created, the hope is that practitioners will refer to these knowledge bases and change their use of terminology accordingly. This ideal usage scenario is rarely realized (Cargill, 2011). This may not necessarily be the ontology's fault; the simple fact is that the implementation of top-down standardization does not streamline the execution of terminological changes in the workplace. In order to even use the current ontologies we have, changes need to be made that allow the everyday employee to implement more precise definitions in his own field.

2.15 The Precision Problem

When detailing the failures of ontologies in the terminological ambiguity field, it must be realized that it is not simply a failure to execute. Many of the previous issues discussed give the impression that if we only were better in our implementation of standardized bodies of

knowledge then we would have a solution. However, let us consider what any ontology is actually doing: it is "tightening" the definitions, or in other words, it is making definitions more "precise" in their usage.

When multiple knowledge bases in multiple fields—sharing common key terms between them—decide to utilize precision on their ambiguous terms, an interesting effect occurs. As precision is executed simultaneously in multiple ontologies each execution is also isolated from the others. As definitions become "tighter" in each ontology, more conflicts arise when we need to combine and share results (Ekstrom, 2012).

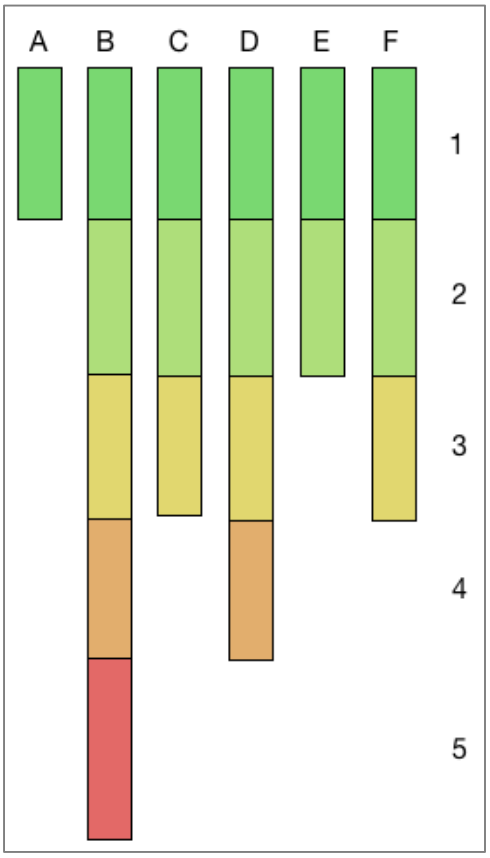


Figure 7. Tunnel Diagram of Terminological Precision

The letters in *Fig. 8* represent different groups of people all using the same term. These groups could be different disciplines, or even different departments in a university or corporation. For the sake of argument let's say the term is "process". The colors and numbers represent the similarities between each group's definition of "process." Each group started out with a high-level concept that was almost the same. At this point in time, "process" was a term that was easy to use in intergroup conversations. However, as time went on, some of the groups further specialized their meaning of the term according to their group's needs.

We can see that group B has dug the deepest tunnel of precision, while group A has the shallowest. In other words, group B's definition of "process" is extremely specific while group A did not change their usage of the term from its original meaning. Group B's specific definition shares the most commonality with group D's fairly specific definition—those two groups would not have many miscommunications over this particular term. However, if group B and group A needed to collaborate, they would quickly run into problems as their working definitions have little in common.

For group A and group B, it would be similar to saying that you and I must have a lot in common because we share the surname "Smith." When the surname originally came into being, perhaps people named "Smith" shared the same profession or close blood ties. Now it's just a term used to refer to any number of people. Such informational silos around terms create significant problems in interdisciplinary information sharing (Miller, Jones, Graves, & Sievert, 2010).

2.16 Terminology Mediation Tools

Because attempts to remove ambiguity from language have been less than adequate, some have shifted their focus to tools that help manage and mediate terminological dissonance. The most relevant tools are documented.

2.16.1 SDL MultiTerm

The simplest implementation of terminology mediation is actually called "terminology management" and involves storing and managing a corpus of terminology for a specific organization. SDL MultiTerm is a commercial application that can import existing glossary data from Excel files and other documents to compile a termbase. The focus of tools like MultiTerm are to import, store, and export data. This type of application is typically used in conjunction with translation tools for multi-language corporations.

2.16.2 CRCTOL

Further towards this idea of "mediation" is a tool called CRCTOL, or Concept-Relation-Concept Tuple-based Ontology Learning (Jian & Ah-Hwee, 2009). CRCTOL is a domain ontology learning system that uses a full text parsing technique combined with statistical and lexico-syntactic methods. Traditional ontology learning systems use shallow Natural Language Processing (NLP) techniques to extract concepts and IS-A relationships from documents. While this approach is certainly better than nothing, it leaves a large burden of tedious extraction work on the human agent. Applications such as CRCTOL create solid foundations for terminology mediation tools because they can detect more complex concepts and relationships while leaving the nuances of ambiguity up to us.

2.16.3 Termediator

The Termediator tool was developed specifically for the purpose of interdisciplinary terminological mediation (Richards, Riley, Ekstrom, & Tew, 2013). Termediator uses a cosine vector model to identify synonymous terms. Polysemous terms are identified by the combination of clustering methods and candidate thresholds using cosine, latent semantic indexing, and latent dirichlet allocation models. There are other tools that attempt to identify semantic dissonance, but Termediator is currently the only tool we know of that attempts to mediate conflicting terms and concepts within an interdisciplinary context.

3 Methodology

3.1 Introduction

This research defines the space of terminology mediation within the scope of Information Technology and related fields. Once the groundwork for that concept is laid, the research analyzes the implementation and contents of the terminology management tool Termediator. The conceptual prototype proposed by Termediator is validated by manual and statistical analysis.

3.2 Define Terminology Mediation

Objective 1: Describe and define "terminology mediation" as an area of research and "terminology mediation tools" in an Information Technology context.

The terminology mediation space is defined via a comprehensive overview of the following: the background of terminologies and ontologies; the sister space of terminology management; terminology mediation concepts listed under other areas of study; an overview of relevant tools; linguistic factors that contribute to the need for terminology mediation; benefits of terminology mediation in singular and multidisciplinary work; relevance to Information Technology.

This objective is also augmented by the author's participation in the development of Termediator and subsequent publications and presentations on said tool.

This definition of the space proposes that a solution to terminological dissonance is "terminological mediation" as opposed to previous attempts at bypass, removal, and base management of terminology.

3.3 Verify the Contents and Implementation of Termediator

Objective 2: Verify the contents and implementation of the Termediator tool.

The contents and implementation of the Termediator tool are discussed and illustrated in detail. This discussion documents how Termediator accomplishes identification tasks in synonymy and polysemy in the following domains:

- Information Technology
- Information Security
- Computer Science
- Information Systems
- Graphic Design
- Systems Engineering
- Software Engineering
- Business Process Management
- Workflow
- User Experience Design
- Enterprise Architecture
- Robotics
- Telecommunications

Termediator's evolution and record of development is recorded and presented. The tool's history provides a roadmap for future research in terminology mediation. Usage technique and features of the user interface are also discussed.

Samplings of Termediator's synonymy and polysemy results are provided to illustrate the tool's term dissonance identification ability. These results are also presented in multidisciplinary team project contexts to illustrate the potential for Termediator to prevent miscommunication via preliminary identification of dissonance in a multipl domain set.

3.4 Validate the Conceptual Polysemy Model Prototyped in Termediator

Objective 3: Show that the Termediator tool generates a list of twenty polysemous terms that are more polysemous than a randomly generated list. Terms in both lists will be generated from the same set of Information Technology glossary data. The polysemy comparison is determined by user survey data.

This final objective is accomplished through a user survey. This survey validates Termediator's ability to identify groups of highly polysemous terms. Because the tool's implementation of polysemy is more sophisticated, the analysis focuses on polysemy results as opposed to synonymy results. The scope of the data is limited to concepts and terms located within the Information Technology knowledge domain.

Validation of the polysemy algorithm uses data collected from a user survey. Participants consist of university students enrolled in the Winter 2015 Capstone class (IT 466) in the Brigham Young University Information Technology program. This user base coincides with the chosen scope of Information Technology terms and dissonance within IT. This set of participants

provides a preliminary insight into terminology comprehension within Information Technology academic programs.

Extra credit was provided as an incentive to complete the survey. Thirty-four students participated in and completed the survey.

The survey began with a short presentation on the basic definition of polysemy. Such a presentation was necessary as the concept of "polysemy" is not common knowledge among IT students. This survey presented the following information:

- What is polysemy?
- Three examples of how polysemy can lead to miscommunication.
- A sample list of twenty polysemous words.
- The words will not be IT specific as not to confound the survey results.
- 2-5 minute question and answer period should any of the participants not grasp the concept of polysemy.

Once the presentation concluded, the participants were asked to compare terms from a list generated by Termediator and a list randomly generated from the same dataset.

Each participant compared 20 pairs of terms, one at a time, and chose which member of the pair is more polysemous. The survey was in multiple choice format and the users were given the choice of the first term, second term, or "not sure."

A response to each question was required, so there were no incomplete surveys submitted. The survey was set up online on the Qualtrics survey system. The list of questions in the survey was presented as follows in *Table 1*.

Table 1. Survey Question Format

Which term is more polysemous?
1. Term A1 2. Term A2 3. Not sure
1. Term B1 2. Term B2 3. Not sure
….

The position of terms in a pair are randomized. This randomization ensured that there was not a deliberate pattern in the survey where all of the randomly generated terms were in one column and all of Termediator's terms were in the other. However, this randomization was done before the survey was created; this ensured that all participants viewed exactly the same survey and had a similar survey experience.

4 Termediator Documentation

4.1 Initial Prototype

In 2010 the first prototype of the Termediator tool was created. This was the first attempt to build software to investigate and attempt a partial solution for synonymy and polysemy. This prototype parsed and normalized the ISO/IEC 24765 (sevocab) data into Python 'dict' data structures. To grant web access to the data, we used a Django (Django Project, 2013) interface paired with the dictionary persisted in SQLite as the database. Through this interface we sought to create a way to explore the terminology in ways that the sevocab did not allow.

The main function implemented was a web "term browser" that allowed the user to browse terms by how many concepts they had. Sorting high to low on the number of associated concepts is useful when searching for potentially dissonant terms.

Although the research at this time (Ekstrom, 2012) was very preliminary, the work performed on this initial prototype gave us the framework for more sophisticated tools in the years to come.

4.2 Data Acquisition and Normalization

We started out with the hundred-plus glossaries in the ISO/IEC 24765 (also known as the *sevocab*) as our main dataset. Over time we have added hundreds more glossaries written by

standards organizations, universities, corporations, and grassroots coalitions. Currently our dataset encompasses 18 disciplines, 48755 terms, and 87635 concepts.

Following the inclusion of *sevocab*, we located many of the additional glossaries through web searches and by pursuing citations in previously acquired academic papers and journal articles. Some examples of search terms used: "computer science glossary," "workflow terms," "concepts in systems engineering," "telecommunications knowledgebase," "graphic design termbase," and so on. Tweaking search terms and using different search engines enabled a continuous influx of web glossaries for quite some time. we also perused known universities, corporations, and other organizations for their publicly available glossaries that may not have been indexed by a search engine.

To parse and normalize the data, we built Python parsers and web scrapers utilizing ElementTree (ElementTree API, 2014) and BeautifulSoup (BeautifulSoup API, 2014) libraries. Through these libraries we were able to interpret and transform glossary data from PDF and HTML files into standardized XML. These XML files conform to each other via an XSD, which is a schema that defines the structure of an XML document. Our XSD uses XML Schema standard version 1.0. This is the most widely accepted version of the language at the time of this writing. A short example of a glossary's tag structure is as follows:

Table 2. Glossary XML Structure

```
<Glossary>
<Entry>
   <Term>Term Text</Term>
   <Concept>Concept 1 Text</Concept>
   <Concept>Concept 2 Text</Concept>
</Entry>
</Glossary>
```

As you can see from the example, each glossary contains a root **<Glossary>** that contains <Entry> tags. Each **<Entry>** has one term and one or more **<Concept>** tags.

Additionally, **<Glossary>** tags have three attributes: OriginName, OriginAuthor, and OriginURL. Terms can optionally contain child **<TermAnnotation>** elements of four types: Note, SeeAlso, Synonym, or Reference.

After we have a group of XML files ready, a merger program combines and sorts all of the terms and concepts in these XML files into one compendium. The compendium is then dumped into a SQLite database for our synonymy and polysemy analysis.

Because HTML and PDF files are coded for layout, and not for content, there was no standard format shared between each glossary. This meant that building one parser that could handle every glossary was impossible. Parsers or scrapers were hand-coded for each acquired glossary. Many of the glossaries had incorrect syntax or inconsistencies in the placement of content. This required additional conditional statements to ferret out these errors in the parsing process. An example of a parser for PDF file using ElementTree is located in *Appendix A*, and a parser for a HTML file using BeautifulSoup is in *Appendix B*.

What followed the initial glossary aggregation prototype was the "Termediator" tool: this tool's end goal was to automatically identify synonymous dissonant terms between two or more fields (Richards, Riley, Ekstrom, & Tew, 2013). Recall that of the two types of dissonance, there is synonymy and polysemy, and at this point the tool only focused on detecting synonymy. There was a lot of work to be done to reach that point, and the first step was to create a standardized method for data input and normalization. Once our data acquisition chain was in place, we proceeded to quadruple the size of our data set and broaden its reach by bringing in glossaries from over fifteen overlapping domains of interest.

In layman's terms, detecting synonymy is detecting when term A and term B share Concept C. To identify synonymous terms, a vector model "similarities matrix" was created to compare every concept with every other concept; each relationship was then assigned a similarity ranking. A perfect similarity ranking of 1 meant the concepts were identical, and anything close to 1 meant the concepts were very similar. Termediator then linked each concepts to its 3 most similar concepts in the web interface. At this point, there was not yet an automated way to list synonymous terms, they could only be identified by manually browsing through Termediator's term list.

When a user selects a term, all of its concepts are displayed underneath it. Clicking on a concept reveals the top three terms that the concept shares the most similarity with. For example, a specific definition of "system" shared the most similarity with the concepts linked to the term "relationship."

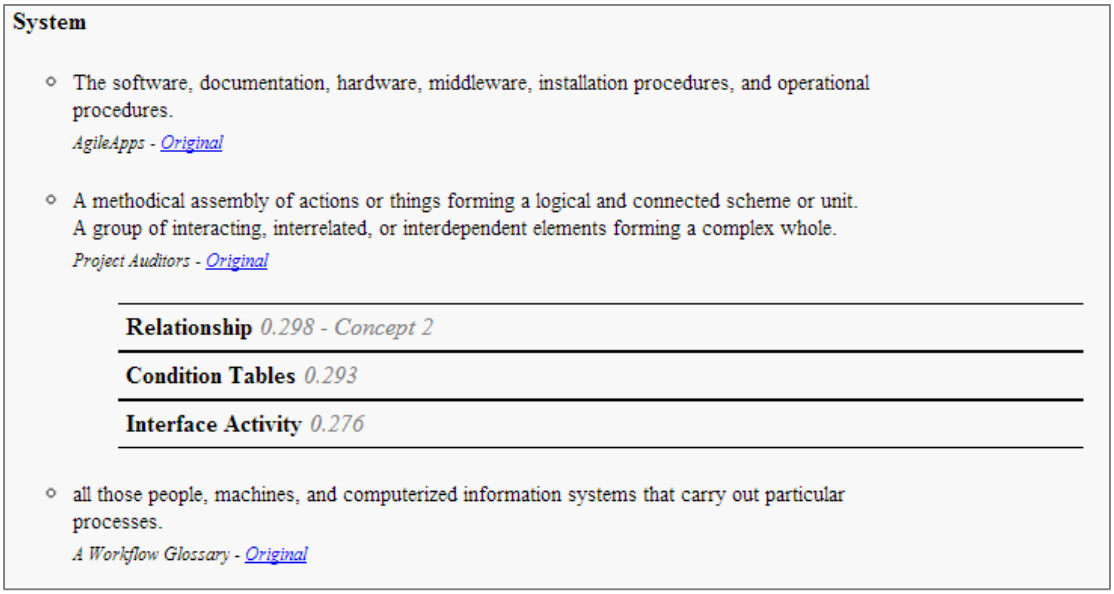


Figure 8. Term, Concept, and Synonyms

Drilling down into one of the top three terms reveals the similarity table. This table is unlikely to be of much use to the lay user, but is a point of interest for developers and those interested in the algorithms behind the tool.

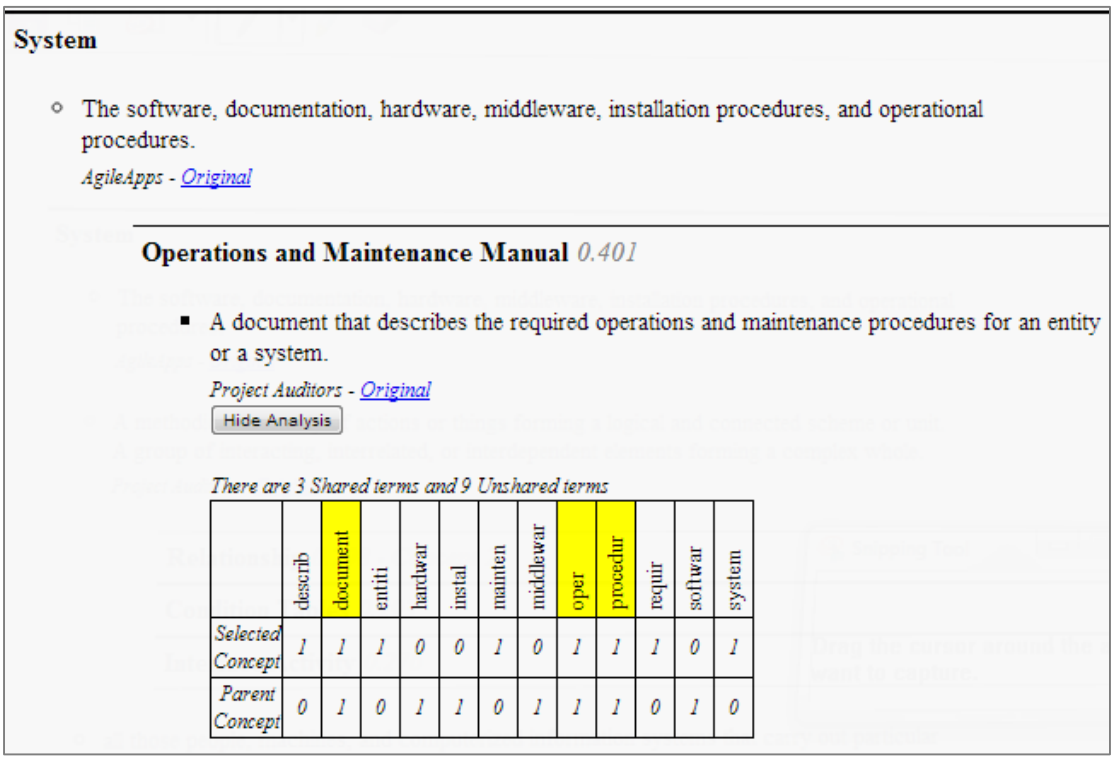


Figure 9. Term, Concept, and Similarity Table

The analysis chart gives the user a look into how the similarity ranking was between that concept and the original concept was derived. This goes back to basic vector creation where each word in a concept has *n* dimensions corresponding to the total number of distinct terms.

The simplest form of vector creation can be understood in the abstract by a basic example. Consider the following two phrases:

P1 = The red cat
P2 = The angry dog

First, we take each distinct term and create vectors of each. The vectors of terms P1 and P2 then become:

Table 3. Basic Vector Creation

	The	**Red**	**Cat**	**Angry**	**Dog**
P1	1	1	1	0	0
P2	1	1	0	1	1

Thus, P1 can be represented as the vector (1,1,1,0,0) and P2 can be represented as (1,1,0,1,1). Once concepts are converted to vectors, we can use similarity measurements to determine how close the two angles made by the vectors are which results with a value between zero and one. A value of zero means there is no similarity (there are no shared dimensions between the vectors) and a value of one means there is perfect similarity (the vectors are the same).

By viewing the analysis chart, a user can see which terms are the key shared terms between concepts. This can further our understanding of how synonymous concepts connect to each other.

4.3 Polysemy Identification

The next step for the Termediator tool (Riley, Richards, Ekstrom, & Tew, 2014) was to attempt to identify polysemy, or when a word or phrase is linked to multiple conflicting concepts. Consider that the intuitive way for a human to find a polysemous term is to look at a term's concepts and sort them into groups by meaning. If there are many groups of meaning, then it may be reasonable to assume that the term is polysemous. If Termediator could automatically sort concepts into these semantic groups, then we could see which terms had the most clusters and therefore the most potential for dissonance.

Termediator uses the hierarchical agglomerative method to create semantic clusters of concepts under a term (Riley O. , 2013). Hierarchical methods were chosen due to their well-documented history in polysemy identification. Such methods needed proximity matrices that would indicate how similar one concept was to another so that Termediator could create accurate clusters. Very similar concepts should be in the same cluster, while highly dissimilar concepts should not. Three different similarity algorithms were used to produce these measurements: cosine, latent semantic indexing (LSI), and latent Dirichlet allocation (LDA). All three of these similarity methods produce concept similarity values between zero and one (higher values indicate more similarity between two concepts). Using these values, Termediator then generated the proximity matrix for each term's concepts.

Although our similarity measurements thus far measured each concept to every other concept, we also needed a measurement of similarity between *clusters* of concepts. We initially looked at three linkage types: single, complete, and average. We chose not to evaluate single linkage because prior research has proven that it "generally gives results that are far inferior to those obtainable when the other hierarchic agglomerative methods are used" (Willett, 2000). We

then evaluated average and complete linkage and determined that both should be included as options in our clustering method.

4.3.1 Discovering the Candidate Threshold

To make the clustering data useful, each term needs a measurement that determines which of its concepts are clustered together. This measurement is called the "candidate threshold". Poor thresholds group dissimilar concepts together, while good thresholds group similar concepts together. At the best threshold, all the concepts underneath a term are categorized in the appropriate semantically related groups.

Grasping the idea of candidate thresholds requires an understanding of agglomerative hierarchical clustering and its visualization in a dendrogram, which is a tree diagram that illustrates the cluster arrangement.

In agglomerative hierarchical clustering, each concept starts out "in a class by itself" isolated in its own cluster. As the threshold value increases, concepts become grouped together and the overall number of clusters linked to a term decreases. Also note that, at some threshold value, a term will eventually collapse all of its child concepts into one singular cluster.

In *Figure 10* there are sample "slice" markings made. The threshold value is intuitively where the dendrogram is horizontally sliced. This slice determines which clusters are produced. Higher thresholds result in less semantic groups. As indicated in the figure, there is a slice point at which all of a term's concepts collapse into a single cluster.

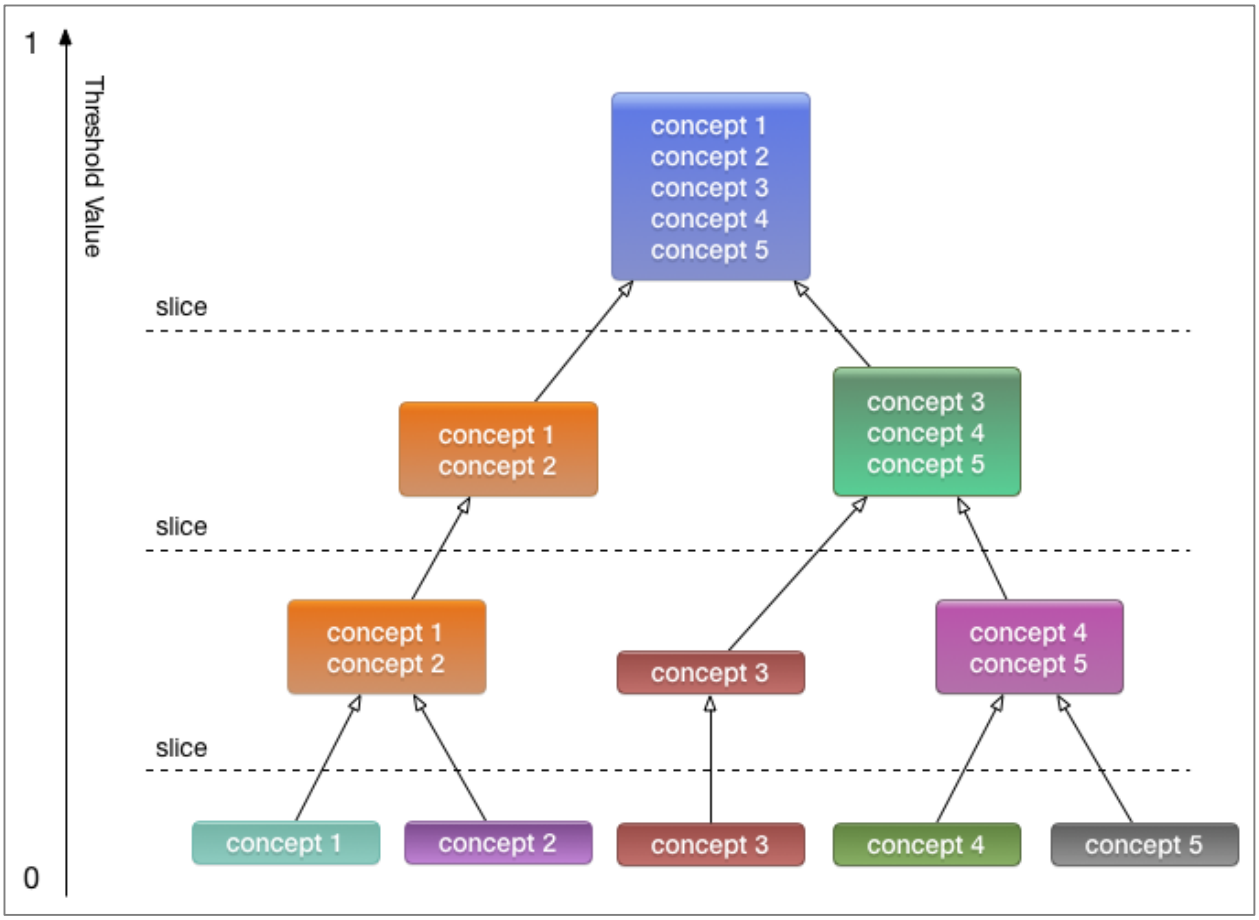


Figure 10. A Sample Dendrogram

We created a web interface for the polysemy tool that allows the user to drag a slider—this slider changes the dendrogram slice and adjusts the clusters of concepts accordingly. The higher the threshold, the fewer clusters were produced. In the figure below we can see that "process" has 9 clusters at a very high threshold.

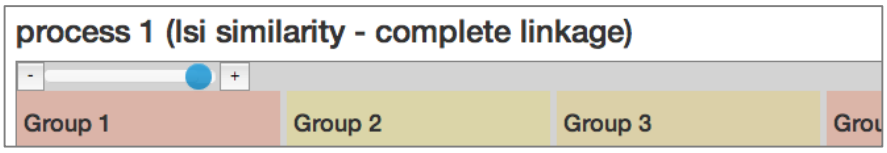


Figure 11. Clusters in the Web Application

4.4 Convergences

The dendrogram slider provided insight into a potential measure of interest. Recall that every term has a threshold at which it converges all text concepts into a single cluster. This is called the "convergence point." Terms that we intuitively identified as "simple" converged at lower threshold values than "complex" terms. In other words, it appeared that highly polysemous terms had *more clusters* at *higher threshold values* than less polysemous terms.

Guided by this insight, we ran all three clustering methods on every term and recorded the corresponding convergence points. To further the analysis, we combined each convergence value with the mean. Graphing all of these convergence points simultaneously (see figure below) revealed the trend persists for all similarity measures and linkage types.

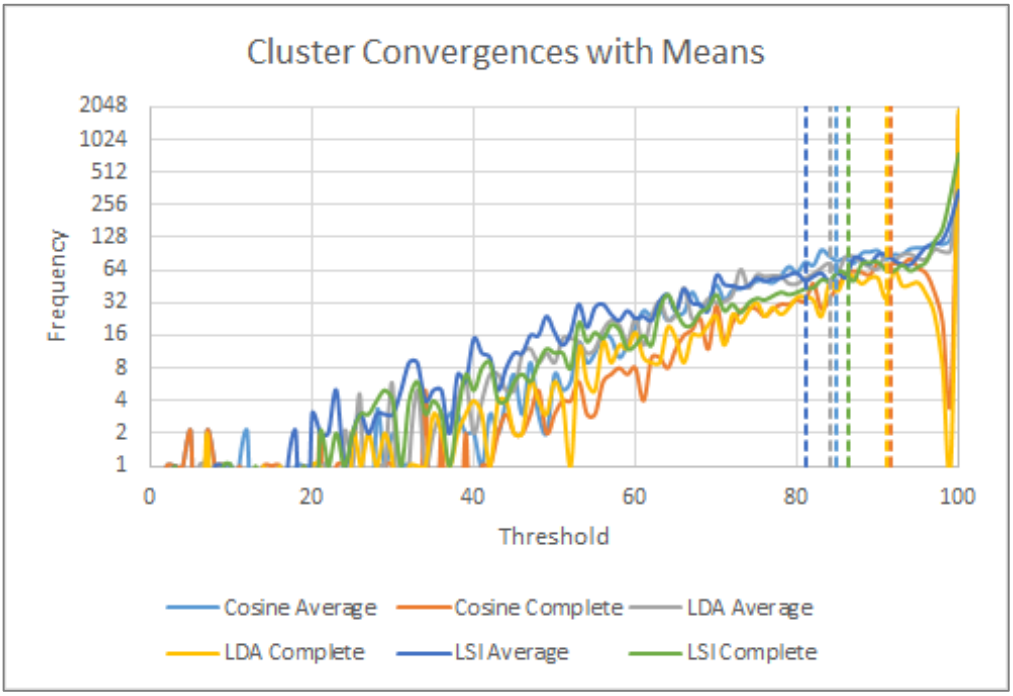


Figure 12. Cluster Convergences

This data was then used to generate an "average convergence point" for each clustering and linkage combination. An average convergence point is the average value at which terms using that particular clustering-linkage combination converge all their concepts into a single cluster.

The matrix of average convergence points was then utilized to perform hierarchical clustering on each term in the compendium. The results were sorted by cluster frequency. This produced table of terms with the most clusters for each of the six clustering-linkage algorithms: LSI complete, LSI average, LDA complete, LDA average, cosine complete, and cosine average. Consider the list of top clustered terms for LSI average:

1. interface (15 clusters)
2. function (11 clusters)
3. object (10 clusters)
4. unit (9 clusters)
5. standard (9 clusters)
6. process (9 clusters)
7. node (9 clusters)
8. link (9 clusters)
9. firewall (9 clusters)
10. system (9 clusters)

The results in this table are interesting because they include many terms that can be intuitively identified as polysemous terms. Words ranked highly in these results, such as "function," "process," and "resource," are fraught with potential for miscommunication.

One may be concerned that this is a simply a list of terms that have a high *number* of concepts, but that is not the case. Terms with even higher numbers of concepts, but a low number of semantic groupings, are generally weeded out by this clustering strategy. The following top twenty list was generated by Termediator's polysemy tool within the Information Technolgoy domain. This particular list categorized as "LSI average"; this means it was generated using the LSI method with the average linkage type.

1. data - 8 clusters - 49 concepts
2. standard - 7 clusters - 24 concepts
3. interface - 7 clusters - 52 concepts
4. graphic - 6 clusters - 22 concepts
5. filter - 6 clusters - 18 concepts
6. archive - 6 clusters- 16 concepts
7. access - 6 clusters - 22 concepts
8. user - 5 clusters - 46 concepts
9. template - 5 clusters - 37 concepts
10. signature - 5 clusters - 29 concepts
11. redundancy - 5 clusters - 16 concepts
12. queue - 5 clusters - 18 concepts
13. process - 5 clusters - 44 concepts
14. post - 5 clusters - 17 concepts
15. parameter - 5 clusters - 16 concepts
16. node - 5 clusters - 35 concepts
17. interactive - 5 clusters - 15 concepts

18. firewall - 5 clusters - 74 concepts

19. feedback - 5 clusters - 20 concepts

20. cut - 5 clusters - 12 concepts

This clearly shows that a high number of concepts does not necessarily mean a high number of clusters. *Firewall* clocks in at 74 concepts but only 5 clusters, while *cut* has a much lower 12 concepts but the same 5 cluster total. The correlation coefficient between the clusters and the concepts in the above data set is approximately 0.2 which is a very low correlation, thus emphasizing that the clustering method is measuring polysemy beyond a term's total number of concepts.

According to the preliminary analysis, terms that still have a high number of clusters at, or above, the average convergence point tend to be more polysemous. Thus this average convergence point may give us an automated method for identifying polysemy, in spite of the accuracy problems associated with analysis of short texts such as glossary concepts.

4.5 Examples of Synonymy Identification

Obvious synonym matches are most often found in concepts that only encompass a "see also" reference to another concept. An example of such a term is "AI" which has the concept "See Also Artificial Intelligence."

The Termediator tool succeeds in this particular instance by listing "Artificial Intelligence" as the most relevant match for "AI" based on the concept "See Also Artifical Intelligence." Such synonym matches were a preliminary indicator that Termediator works on a fundamental level for the purpose of synonymy detection.

Termediator also makes a number of accurate matches between terms that have synonymous concepts that are not quite as obvious as a "see" or "see also" concept match. For example, Termediator correctly identifies “malware” as a synonym to “trojan.”

More examples of correct synonymous matches can be seen in the similarity results table under the following terms: Abstraction, Agile Development, Help Desk Management, Information Processing, Source Code, Terminal, Twisted Pair Cable, URL, User Interface. These are not the only synonymous matches, however, they provide a good starting point for perusal of Termediator's synonymy identification function.

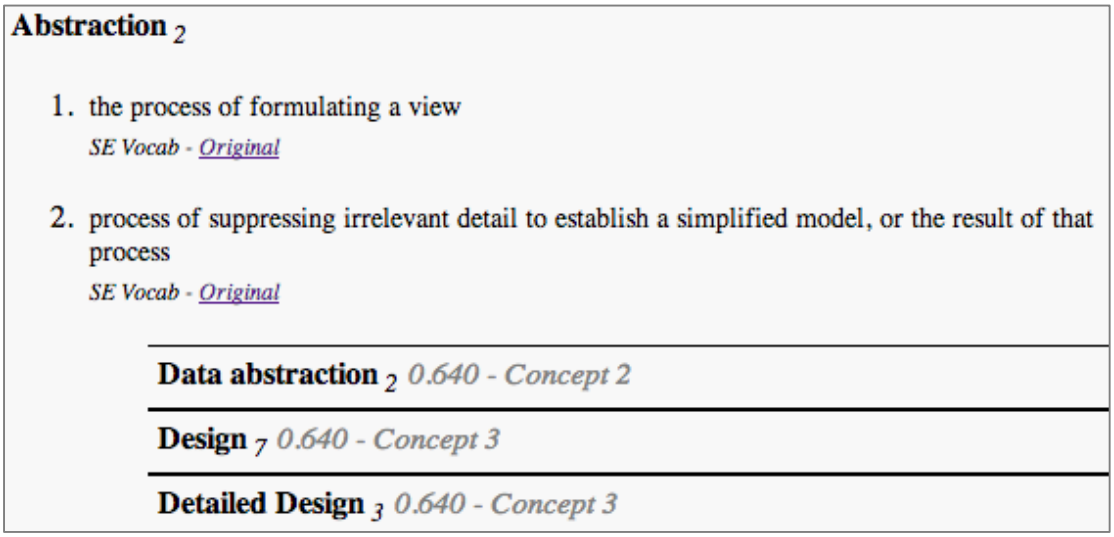


Figure 13. Synonym Matches in "Abstraction"

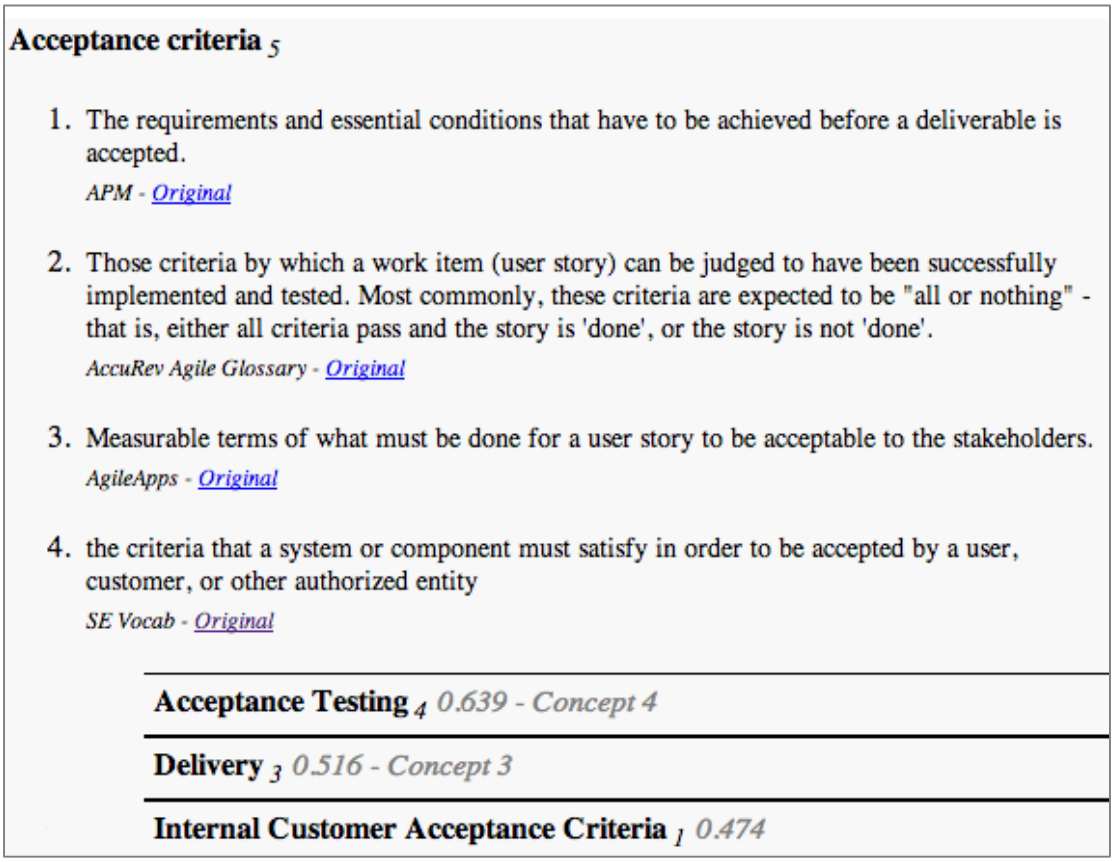


Figure 14. Synonym Matches in "Acceptance Criteria"

Agile Development $_2$

1. Agile development is a way of thinking about software development as expressed in the Agile Manifesto, and acts as an "umbrella" for a group of methodologies. The methodologies are based on process-centric and iterative development, where requirements and solutions evolve through collaboration between self-organizing, cross-functional teams. Agile development is a conceptual framework that promotes evolutionary change throughout the entire life cycle of the project and represents a new, more flexible approach to development than the traditional methods that have previously been the norm for software development.

   AccuRev Agile Glossary - Original

2. software development approach based on iterative development, frequent inspection and adaptation, and incremental deliveries, in which requirements and solutions evolve through collaboration in cross-functional teams and through continuous stakeholder feedback

   SE Vocab - Original

Agile Methodology $_2$ *0.545*

Agile Software Development $_3$ *0.541*

Agile Processes $_1$ *0.473*

Figure 15. Synonym Matches in "Agile Development"

Help Desk Management $_1$

1. Help desk management services provide centralized information and support management service to handle a company's internal or external queries and operational problems about IT-related processes, policies, systems and usage. Services include product support capabilities, including elements of hardware and software support, logging of problems, and results analysis (results analysis means analyzing the results of calls taken to resolution of those calls for entry into self-help database, problem trends to suggest permanent fixes and so forth); dispatch of service technicians or parts; training coordination; and other IT-related issues.

   Gartner - Original

Hardware And Software Maintenance Services $_1$ *0.472*

Product Support Services $_1$ *0.448*

Technical Lead / Technical Point of Contact $_1$ *0.447*

Figure 16. Synonym Matches in "Help Desk Management"

Information Processing 2

1. The handling of information by computers in accordance with strictly defined systems of procedure.
   Iowa State IT Glossary - Original

2. the systematic performance of operations upon information, which includes data processing and may include operations such as data communication and office automation.
   SE Vocab - Original

Data processing (DP) 1 *0.688*
Process 22 *0.662 - Concept 18*
Abstract data type 1 *0.574*

Figure 17. Synonym Matches in "Information Processing"

Source Code 4

- A sequence of instructions, including comments describing those instructions, for a computer system. Also known as program code, program source code, or simply as code.
  AgileApps - Original

Program instruction 1 *0.588*
Computer Programming 1 *0.571*
Computer Technology 1 *0.571*

Figure 18. Synonym Matches in "Source Code"

Terminal *9*

- Computer-like device that includes a screen and a keyboard for both display and input. It connects to the server. It is called “dumb” because it can do very little on its own. It is not a computer by itself. The programs it “runs” are located on the server. Also called a dumb terminal.

  AAO IT Glossary - Original

Servers *1 0.401*
Workstation *3 0.400*
Telnet *2 0.369 - Concept 2*

Figure 19. Synonym Matches in "Terminal"

Trojan *1*

- A harmful application (or routine) that hides it malicious activities by masquerading as a more useful, harmless and desirable application.In many respects a Trojan is like a virus, except that it does not provide a means to copy itself. A Trojan instead relies upon the fact that it masquerades as something useful that people will recommend or copy-on the application. Trojan's are often found in software that is freeware, although it must be stressed that most freeware is free of viruses and Trojans. Anti-virus applications will typically scan for many known Trojans. Anti-virus writers tend to classify a Trojan as one of the following depending on what the Trojan does: Trojan Downloader - downloads and installs new software on the computer. Trojan Dropper - installs new software on the computer. Trojan Proxy - turns the computer into a proxy server providing anonymous access to the internet for the Trojan writer. Trojan PSW - password stealer, hunts the computer for stored usernames and passwords and forwards these to the Trojan writer. Trojan Spy - any form of spyware application, such as keyloggers, password stealers etc. Trojans get their name from the classical Greek story of the Trojan horse, used in the Trojan war. The Trojan horse was a large wooden horse left as a gift from the Greeks to the Trojans, the Greeks then sailed away (apparently admitting defeat). The Trojans moved the horse inside their city walls and then celebrated their apparent victory. Unfortunately for the Trojans the Trojan Horse contains a number of Greek soldiers who then killed the guards and let the remainder of the Greek army into the city who then sacked Troy. A Trojan is also known as a Trojan horse. For more information see: www.cryer.co.uk/resources/antivirus.htm - List of anti virus products that are free for personal use.
  www.historyforkids.org/learn/greeks/religion/myths/trojanhorse.htm - Summary of the Trojan Horse story (for children) http://everything.explained.at/Trojan_horse_(computing)/ - Trojan horse explained.

  Brian Cryer's IT Glossary - Original

Trojan horse *4* *0.640 - Concept 2*

RAT *1* *0.446*

Malware *2* *0.297*

Figure 20. Synonym Matches in "Trojan"

Twisted pair cable *1*

- The type of cable used for most telephone wiring. It has pairs of copper wires twisted together to minimize electrical noise. There are shielded twisted pair (STP) and unshielded twisted pair (UTP) cables. In shielded twisted pair cables, each pair has a metal sheath around it for better protection against interference.

  AAO IT Glossary - Original

STP *4 0.754*
UTP *1 0.699*
Shielded Pair *1 0.514*

Figure 21. Synonym Matches in "Twisted Pair Cable"

URL *4*

- Uniform Resource Locator. A URL provides a standard way to address any resource on the internet.URLs are expressed as a text string of the form: protocol://domain-name/pathname Where protocol is any valid protocol, such as http, ftp, gopher etc. One example of a URL that most people are now familiar with is the web address of each site and page. See also: URI, URN. URLs are defined by RFC1738 and RFC1808. Whilst the HTTP specification does not limit the length of a URL, most browsers do have a limiting maximum size. Whilst this limit is far in excess of most browsing requests, current limits are: Browser / Server Limit Source Internet Explorer (browser) 2083 characters Microsoft Knowledge Base Article 208427 Netscape (browser) No limit Apache (web server) 8190 characters (default) www.jetools.com/content/resources/whitepapers/HTTP_GET_Requests.pdf IIS 6.0 (web server) 16KB For more information see: http://everything.explained.at/Uniform_Resource_Locator/ - URLs explained.

  Brian Cryer's IT Glossary - Original

URI *1 0.499*
URL (uniform Resource Locator) *1 0.369*
HTTP *3 0.358*

Figure 22. Synonym Matches in "URL"

User Interface *6*

- User interface is a part of a computer program in which a human contributes to the operation of the program, either by inputting data, making a decision, or performing other types of work that are dependent on current conditions and priorities. User interfaces are typically specific screens, and are described in subjective terms, such as "friendly", meaning the screens and required actions are highly intuitive and easy to understand.

  ERP Focus - Original

Shell *2 0.413 - Concept 2*
UI *2 0.398*
GUI (graphical User Interface) *1 0.377*

Figure 23. Synonym Matches in "User Interface"

Help Desk Management *1*

1. Help desk management services provide centralized information and support management service to handle a company's internal or external queries and operational problems about IT-related processes, policies, systems and usage. Services include product support capabilities, including elements of hardware and software support, logging of problems, and results analysis (results analysis means analyzing the results of calls taken to resolution of those calls for entry into self-help database, problem trends to suggest permanent fixes and so forth); dispatch of service technicians or parts; training coordination; and other IT-related issues.

   Gartner - Original

Hardware And Software Maintenance Services *1 0.472*
Product Support Services *1 0.448*
Technical Lead / Technical Point of Contact *1 0.447*

Figure 24. Synonym Matches in "Help Desk Management"

4.6 Examples of Polysemy Identification

Termediator is able to successfully identify a number of highly polysemous terms through its warning list generation feature. To recap, the warning list features takes a selected domain and a selected clustering method and generates a list of the "most dissonant" terms in the domain.

This generation feature is also known as a *warning list generator* because the lists are intended to warn users about terms with high conflict potential.

While it is subjective what terms are truly most prone to miscommunication, it seems that Termediator's results match closely with what users would intuitively pick out as highly polysemous terms.

Findings are listed for all eighteen domains in the dataset. The accuracy of the results increases dramatically with a larger data set, therefore the most populated domains will produce better results. The best semantic grouping out of the six similarity measure-linkage type combinations was chosen manually for each domain.

Information Technology (LSI Average)

1. standard
2. firewall
3. access
4. redundancy
5. post
6. interface
7. interactive
8. hierarchy

9. error

10. data

The top term in this set is *standard.* Selected definitions were pulled from the Termediator dataset that illustrated this term's polysemy.

1. An approved model (Department of Education and Communities and Charles Sturt University, 2014).

2. A widely accepted way of doing something (Bleeping Computer LLC, 2014).

3. A mandatory technology, result or procedure to be applied in all appropriate situations (Pepperdine University, 2014).

Information Security (LDA Average)

1. spoofing

2. worm

3. spam

4. risk

5. firewall

6. virus

7. race condition

8. payload

9. zombie

10. whitehat

The top term in this set is *spoofing.* Selected definitions were pulled from the Termediator dataset that illustrated this term's polysemy.

1. Impersonating another person or computer, usually by providing a false email name, URL, domain name server, or IP address (KeyCorp, 2014).

2. A generic label for activities in which trusted relationships or protocols are exploited for mischievous or surreptitious ends especially those cases in which an unknown or unauthorized actor surreptitiously pretends to be a trusted one. The spoofing need not entail personal identification tactics in which a machines identity or address data are usurped are also termed spoofing (El Bucanero, 1996-2014).

3. Spoofing means a router responds to a local host in lieu of sending information across a WAN link to a remote host. The local host thinks the response came from the remote host/ network, when it really came from the router (WestNet, Inc., 2010).

Graphic Design (Cosine Average)

1 . dummy

2. pixel

3. margin

4. font

5. typography

6. typeface

7. thumbnail

8. template

9. resolution

10. register mark

The top term in this set is *dummy*. Selected definitions were pulled from the Termediator dataset that illustrated this term's polysemy.

1. A rough form of any document (1st Impression Printing Waterloo, 2014).

2. A small, detailed page diagram showing where all elements go (Harrower, 2007).

3. A dummy counts as an example of a piece of design work (brochure, ad, book cover etc.) that needs to be approved by the client. Once the client approves the dummy, the designer creates and prints the final design (Conquest Graphics, 2005-2014).

Software Engineering (LDA Complete)

1. abstraction

2. component

3. version

4. risk

5. object

6. domain

7. design

8. database

9 . complexity

10. black box testing

The top term in this set is *abstraction.* Selected definitions were pulled from the Termediator dataset that illustrated this term's polysemy.

1. Generalization, ignoring or hiding details. Examples are abstract data types (the representation details are hidden), abstract syntax (the details of the concrete syntax are ignored}, abstract interpretation (details are ignored to analyse specific properties) (Sommerville, 2010).

2. Parameterization, making something a function of something else. Examples are lambda abstractions (making a term into a function of some variable), higher-order functions

(parameters are functions), bracket abstraction (making a term into a function of a variable) (Sommerville, 2010).

3. A cohesive model of data or an algorithmic procedure (R.S. Pressman & Associates, Inc., 2001-2010).

System Engineering (LSI Complete)

1. constraint

2. interface

3. task

4. system

5. process

6. operation

7. object

8. measure

9. implementation

10. function

The top term in this set is *constraint.* Selected definitions were pulled from the Termediator dataset that illustrated this term's polysemy.

1. Restriction on the value of an attribute or the existence of any object based on the value or existence of one or more others (IEEE, 2010).

2. Restriction on software life cycle process (SLCP) development (IEEE, 2010).

3. Limitation or implied requirement that constrains the design solution or implementation of the systems engineering process and is not changeable by the enterprise (IEEE, 2010).

Business Process Management (LDA Average)

1. process

2. business rule

3. scope

4. resource

5. modeling

6. model

7. event

8. business process execution language

9. business process automation

10. WSDL

The top term in this set is *process.* Selected definitions were pulled from the Termediator dataset that illustrated this term's polysemy.

1. A set of interrelated activities, which transform inputs into outputs (Khosrow-Pour, 2005).

2. The step-by-step sequence of activities (systematic approach) that must be carried out to complete a project (Project Management Institute, 2000).

3. An executable unit managed by an operating system scheduler (IEEE, 2010).

Workflow (LDA Complete)

1. task

2. project

3. baseline

4. XP

5. transparency

6. sprint

7. spike

8. scrum

9. link

10. kanban

The top term in this set is *task.* Selected definitions were pulled from the Termediator dataset that illustrated this term's polysemy.

1. Required, recommended, or permissible action, intended to contribute to the achievement of one or more outcomes of a process (Clarke & O'Connor, 2010).

2. A process that cannot be subdivided any further: an atomic process (Aalst & Hee, 2004).

3. The goals, steps and skills needed to accomplish an activity; a task may comprise a series of sub tasks for instance the task of making tea may involve the sub task of filling the kettle (Amberlight, 2015).

4.7 Multidisciplinary Dissonance Identification

One of the primary use cases considered in Termediator development was the identification of dissonant terms across multiple disciplines. As illustrated in section *4.5 Polysemy Identification,* the tool currently implements a warning list generation feature for polysemous dissonance. These warning lists can possibly help prevent miscommunication in multidisciplinary teams.

Consider a project manager who manages a team of workers who hail from different professional fields. Preventing miscommunication is key to productive output and high morale. The project manager can take note of confusing terms used by team members and check them in Termediator for conflicts. For synonymy, Termediator might help the team realize that Michael uses “task” and Samantha uses “process” to mean the same thing. For polysemy, Termediator could indicate that Gerald and Hannah are not specifying which “interface” or what “system” in their discourse.

Project managers can also prepare their teams at the outset by creating warning lists ahead of time via Termediator. The project manager would select the domains included in the team and generate a list of top potentially dissonant terms. The first team meeting could include a rundown of potentially dissonant terms and definitions from each team member’s domain.

4.7.1 IT + CS + GD

This trio of disciplines (information technology, computer science, and graphic design) is an example of a common interdisciplinary project. Creating a website often involves collaborative teams of information technologists, computer scientists, and graphic designers. Typically the information technologist sets up and maintains the web server, client computers, software across all computers, and any other related computing systems; the computer scientist develops the website backend; and the graphic designer creates the graphic interface and other frontend visual elements. The top dissonant term in this team using “LSI complete” similarity-linkage is "interface."

Table 4. Definitions of "Interface" in Multiple Domains

interface	
Information Technology	A point of connection or junction (Computer Support Group, Inc., 2014).
Computer Science	Class definitions and method signatures provide interfaces. Application program interfaces (APIs) form the interface of a system to applications and often consist of collections of functions or commands in a scripting language. Interfaces may be hidden (available only to the system developer) or exposed (available to others) (LabAutoPedia, 2009).
Graphic Design	The front-end is basically the opposite of the back-end. It's all the components of a website that a visitor to the site can see (pages, images, content, etc.) Specifically, it's the interface that visitors use to access the site's content. It's also sometimes referred to as the user interface (Chapman, 2009).

4.7.2 BPM + WF + ISYS + IT

This quartet of disciplines (business process management, workflow, information systems, and information technology) is another example of Termediator's usefulness in collaborative projects. Creating a functional workflow management system for a big corporation requires the collaboration of workflow analysts and business process managers to create and maintain the flow, as well as information technologists and information systems experts to coordinate the system backend and integration into the company network. The top dissonant term in this team using "cosine average" similarity-linkage is "data."

Table 5. Definitions of "Data" in Multiple Domains.

data	
Business Process Management	Defines the type of information exchanged between business processes (SyBase, 2006).
Workflow	Data elements can be defined by tasks that are accessible only within the context of individual execution of that task (Workflow Patterns, 2010).
Information Systems	Consists of factual elements (or opinions or comments) that describe some object or event. Data can be thought of as raw numbers or text (Post, 2011).
Information Technology	Computer data is information processed or stored by a computer. This information may be in the form of text documents, images, audio clips, software programs, or other types of data (Network Management Solutions, 2014).

4.7.3 Telecommuting

Awareness of potentially dissonant terms is especially relevant in this day and age of telecommuting and global outsourcing. More communication is being performed via text and email rather than in-person or on the phone; even with the advent of video conferencing, it is often more convenient to shoot an email than to Skype with someone five time zones away. For projects where textual communication is dominant, semantic understandings often go unnoticed until a major project flaw occurs. Rather than wait for a communicational crisis, professionals could use Termediator to identify potentially dissonant terms in their collaborative emails.

4.7.4 Education

One of the original intentions of Termediator was to benefit students and educators in Information Technology. Because Information Technology sits adjacent to many other disciplines, it is imperative that IT education include pedagogy that sensitizes students to the potential for misunderstanding because of semantic differences in commonly used terms.

While some more isolated fields still operate under the mindset that "their" definition of a term is canon, someone in IT will work with other fields their entire career and therefore they must recognize the semantic shades of gray. It must also be recognized that when semantic dissonance is encountered frequently, it is not enough to "roll with the punches." Would you tell an Information Security analyst to ignore potential virus threats until one actually infects a machine? Of course not! Clear communication is absolutely essential for the success of IT projects; this is the professional reality that IT students must be prepared to face after graduation. Just as we teach students to prepare for malware or system failure, we should also teach students to prep for effective collaboration and communication with adjacent disciplines. The real problem of miscommunication must be personalized so the student recognizes that "this will be an issue for *me* in my actual career."

As a tool suite, Termediator can be used to sensitize students to the semantic misunderstandings that will occur in their professional careers. Many educational programs already integrate IT with adjacent disciplines in multidisciplinary student projects, and in these cases Termediator can also be used to troubleshoot the miscommunications that occur. Termediator can also help transition faculty from other programs (e.g. Computer Science or Information Systems transfers), create an awareness of synonymy and polysemy in intro level classes, and produce more productive panels in multidiscipline conferences.

5 Discussion

5.1 Future Developments for Termediator

5.1.1 Crowdsourced Data for Polysemy Candidate Threshold

We attempted to crowdsource data from a diverse user set in order to determine the intuitive candidate thresholds for polysemy data. In other words, to find the average threshold value at which dissonant terms tend to separate into the most correct semantic clusters. We made the application publicly available and invited any and all to use the application. The application was integrated into an online feedback form.

In our polysemy web application, each group contained all the concepts deemed to be most similar to each other. Shown in *Figure 25* is a result from the web tool, namely "group 7" from the term "process." Users were expected to experiment with the application's slider, and then select the approximate threshold that produced the most accurate number of groups. If a user saw a concept within a group that was "not like the others" then they needed to change the slider until more similar groups were formed.

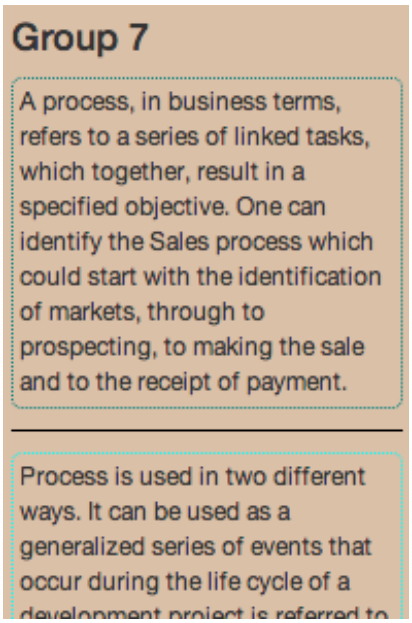


Figure 25. Cluster Definitions

This crowdsourcing experiment failed on multiple accounts. The primary struggle was that initiating interest in this research was difficult as the subject matter requires a great deal of introduction to even understand the problem. It makes the research obscure and generally unattractive to a mass audience. Our marketing strategy was also disorganized. There was not a clear, unified effort to broadcast the existence of this application. We were not sure what groups of people we should introduce the app to and what channels we should communicate on. This limited our overall audience.

When we did get a user to actually use the tool, the online application was overwhelming and confusing to most. The introduction and tutorial content were not adequate. Less than 10% of users enticed to the site actually provided any data. Even three users (two of which identified themselves as technically savvy) who received a personal walk-through of the application in-person still had significant difficulty using the tool. The application was also irredeemably slow which significantly dampened the user experience.

If we were to attempt crowdsourcing data again, there are several changes that should be made. The first is to investigate potential demographics and choose a small number of viable targets. Although it does potentially narrow the total applicant pool, in reality selective advertising may increase the number of total participants because marketing efforts will be more effective. Targeted data will also be more useful in the analysis stage.

Secondly, the testing tool's introduction and tutorial content should be brief yet engaging. Several forms should be provided (text, video, graphical) to provide options for different learning preferences. We may have to direct users of the tool to different introductory content depending on their background. Different strategies are needed for technically oriented and non-technically oriented people. Level of familiarity with communicational linguistics should also be considered. The loading and processing speed should also be drastically improved before another user test is conducted. Lastly, some sort of compensation or other incentive must be provided for participation.

5.1.2 Warning List Generation for Synonymy

The best "one and done" feature we have in Termediator is the warning list generation feature for polysemous terms. In this part of the tool, the user selects one or more disciplines from a list and, with a click of a button, is provided with a list of the top terms with the most potential for communicational conflict. This is a fantastic feature that really provides a practical use for Termediator in the workplace or classroom, yet no feature exists for synonymous terms. As synonymy and polysemy are partners in the ambiguity problem, a synonymy warning list feature is needed so that Termediator can capture the full scope of the dissonance.

5.1.3 Browsing for Polysemous Tools

Although it has this feature for synonymy, the polysemy portion of Termediator does not allow the user to browse all terms in the database with their polysemy data attached. This limits the user's level of engagement as they can only access a term's semantic clusters if they manually type in a term they already know. Much was learned just from browsing the synonymy tool, and adding this feature to polysemy would help both the end user and the developer.

5.1.4 Integration of Synonymy and Polysemy Interfaces

Synonymy and polysemy are two major contributors to terminological dissonance, and they often co-exist in communicational scenarios (e.g. it is not surprising when a synonymous term is also polysemous). In Termediator, these tools were developed alongside each other and currently function in relative isolation. It is imperative that the frontend interfaces be combined. A user should be able to browse through all the terms, pick a term he finds interesting, and then view both synonymy and polysemy data in one clean interaction.

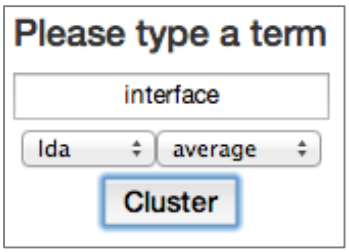


Figure 26. Manual Search

5.1.5 Integration of Synonymy and Polysemy Interfaces

Synonymy and polysemy are two major contributors to terminological dissonance, and they often co-exist in communicational scenarios (e.g. it is not surprising when a synonymous term is also polysemous). In Termediator, these tools were developed alongside each other and currently function in relative isolation. It is imperative that the frontend interfaces be combined. A user should be able to browse through all the terms, pick a term he finds interesting, and then view both synonymy and polysemy data in one clean interaction.

In the interface diagram illustrated in *Figure 27*, there are multiple content box elements that allow the user to browse terms and drill down into data without changing screens. Although changing screens may be useful for an advanced user who has multiple monitors at their disposal, it is less likely that a beginning or intermediate user would intend to have multiple browser windows open. Multiple browser windows would also impede any user on a laptop or mobile device. A possible alternative to this arrangement of multiple frame elements would be pop-up, draggable box elements that would remain on the same screen but could be dragged out of the way or closed when no longer in use.

The following suggested interface combines functions that are currently separated in the most recent version of Termediator. The most notable functions that benefit from this integration are: manual search, alphabetical browsing, polysemy clustering, and synonymy analysis tables. The inline box elements enable full access to these features on one screen. The primary downside of this design is that simultaneous activation of all features will require a full-screen window or horizontal scrolling on smaller monitors.

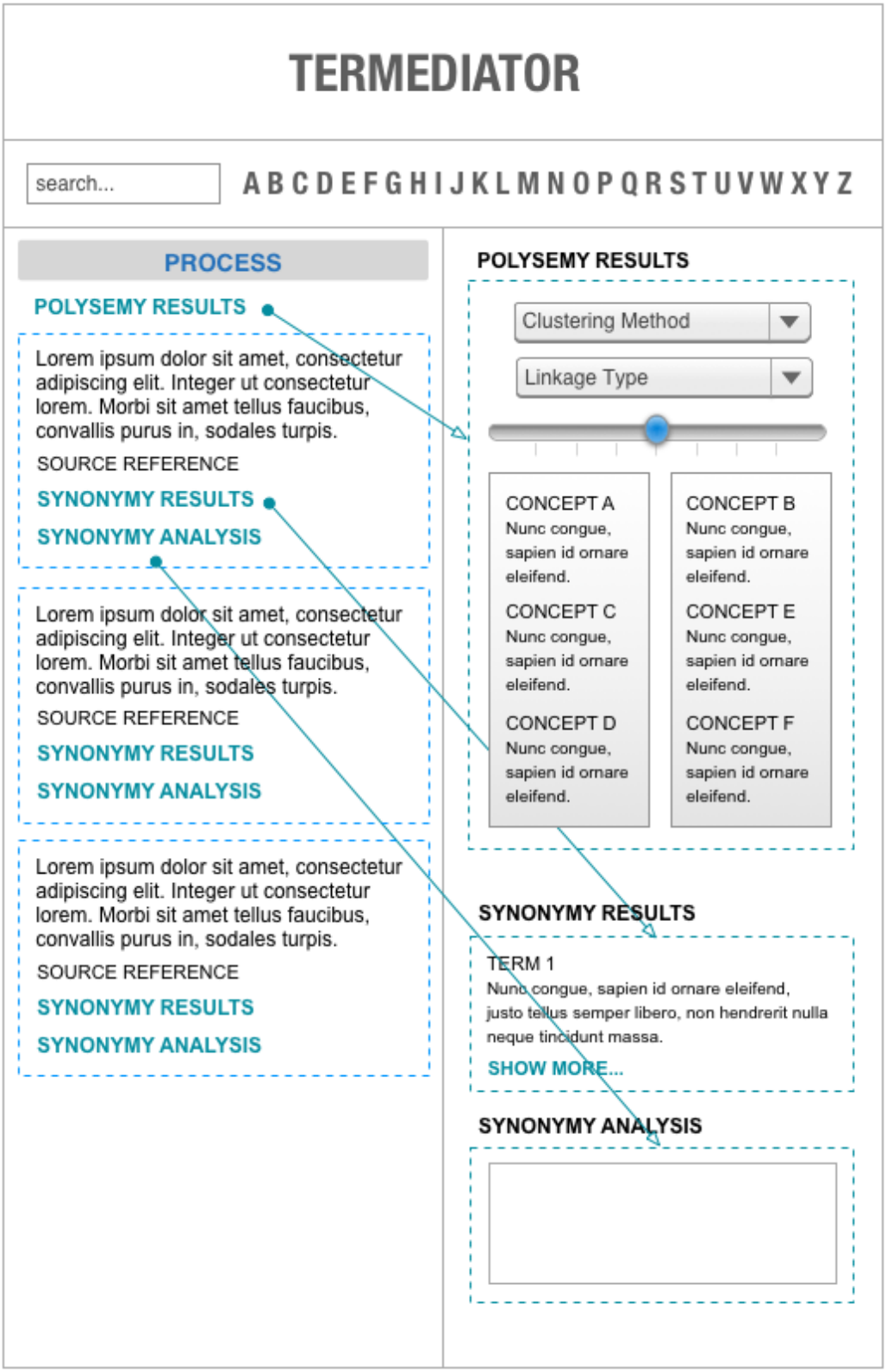


Figure 27. Diagram of Proposed UI

5.1.6 Code Efficiency

The current implementation of the similarity measures, linkage types, proximity matrices, and clustering algorithms can be accurately described as "proof of concept." We did whatever was necessary to show that the product worked as quickly as possible. The code itself is not cleanly written or efficiently implemented.

5.1.7 Mobile Application

Although Termediator has plenty of potential as a practical tool on the computer, there is no mobile application or mobile optimized website. Collaborative meetings are more likely to be populated by tablets and phones than a laptop computer. A mobile interface would help Termediator increase its reach and maintain its practical relevance to interested consumers.

The mobile interface should focus on getting necessary information to the user as quickly as possible. The interface should be devoid of extra experimental features such as the synonymy analysis chart or the polysemy clustering threshold slider.

5.1.8 User Visibility of Polysemy Clustering Methods

Six clustering-linkage methods were used to compute polysemous term warning lists: LSI complete, LSI average, LDA complete, LDA average, cosine complete, and cosine average. Not a single combination was obviously superior to the others in creating these warning lists. Some terms only had accurate warning lists with one combination. Others had nearly identical warning lists with two or three combinations. And then some term lists had almost no discernible difference between all six combinations. Further research should be done to determine if there is

a way to either: 1) choose a combination that is, on average, the best for most terms, or 2) have the app determine which combination is best for each individual term.

As it stands, the polysemy tool allows the user to choose which similarity-linkage type is used when he generates a warning list. While this an engaging feature to some, most lay users have no interest in experimenting with clustering algorithms—they simple want a warning list that will provide the best information for their workplace communication woes. Further revisions of Termediator should remove this and other "testing" options from the general interface, provided there is an intelligent way for the app to choose these algorithms for the user without losing the most relevant analysis.

5.1.9 Sophistication of Current Clustering Methods

The implementation of our current polysemy clustering methods has significant room for improvement. Termediator currently uses several hierarchical agglomerative clustering methods. We chose hierarchical agglomerative methods because of their ease of use and the high level of documentation that accompanied their implementation into our project. However, the hierarchical agglomerative umbrella may not be producing the most accurate results possible for polysemy grouping. Like the vast majority of clustering algorithms, hierarchical agglomerative methods are partitional, meaning that one entity can only be in one cluster at any point in time. A partitional view does not always match up with real datasets, where one entity may be involved in multiple clusters. The newer overlapping clustering model allows hard assignment of data points to multiple clusters (Banerjee, Krumpelman, Basu, Mooney, & Ghosh, 2005). This especially applies to polysemy as the definition of polysemy is something that has multiple usages and multiple group membership is therefore implied. Polysemy's relation to membership

in multiple clusters is confirmed when a user manually sorts concepts into semantic groups—some concepts should be included in more than one cluster to accurately represent that concept's full spectrum of usage contexts (Fukumoto & Tsuji, 1994).

Overlapping clusters may also help us combine the polysemy and synonymy results, for example when a concept is involved in not only multiple concepts underneath a polysemous term, but also linked to another term that is synonymous in some aspect with a polysemous term.

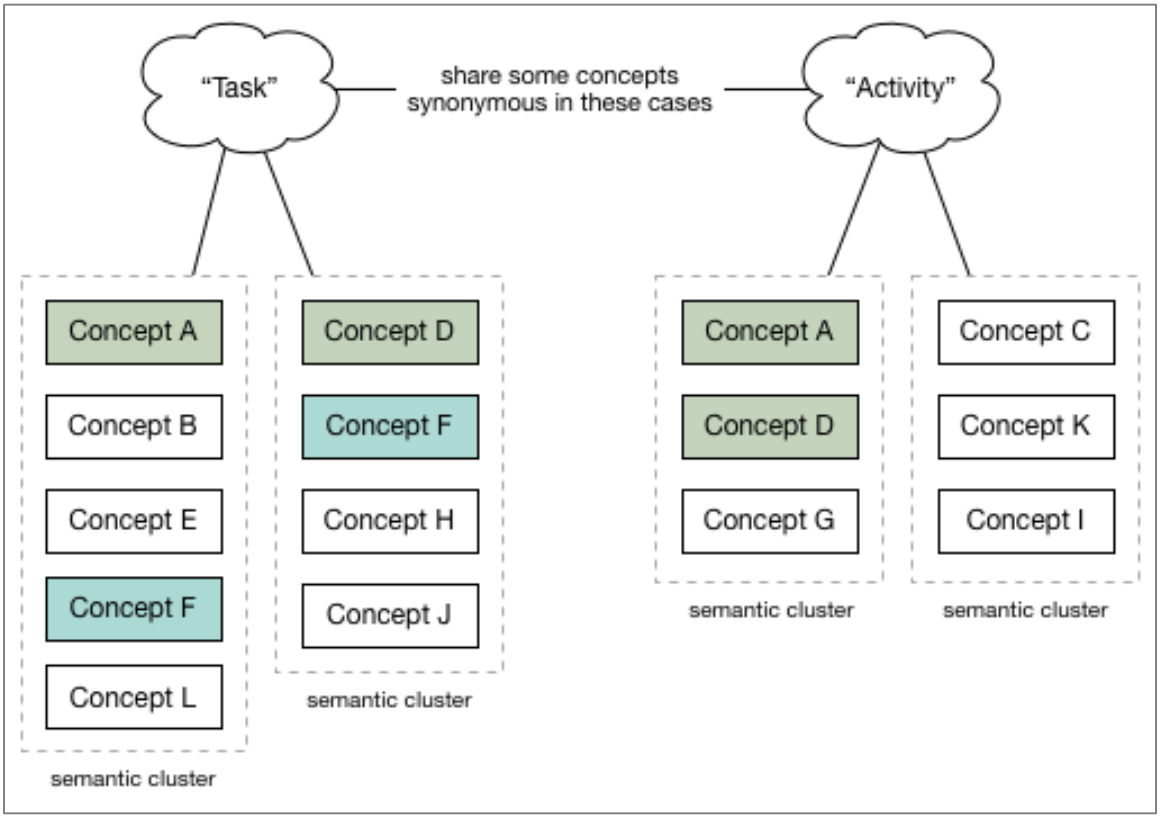


Figure 28. Overlapping Cluster Model

In the above diagram, Concept F has membership in two clusters underneath the term "task". Concept A and Concept D have membership in two clusters, one underneath the term "task" and the other underneath the term "activity." It would be easy for a user to manually

identify these relationships between concepts and semantic groups, however Termediator's current hierarchical clustering methods do not allow these nuanced relationships. Accordingly we may benefit from the exploration and experimental implementation of overlapping clusters in our polysemy research.

5.2 Future Efforts in Terminological Mediation

5.2.1 Awareness of Terminological Dissonance

The likelihood of terminological dissonance needs to be emphasized to the administrators of standards organizations and educational institutions—ideally in all disciplines, but especially in disciplines that have a highly specialized body of terms. When someone is aware of synonymy and polysemy in their own field and in adjacent fields, they may be less likely to create terminological dissonance in their professional communications.

5.2.2 Awareness of Terminological Precision and Information Silos

Many creators of bodies of knowledge understand that ambiguity can create communication problems, and so they attempt to pigeonhole their terms into extremely precise definitions. Without an additional understanding of information silos and their effects, these content creators may not understand how precision in their own limited scope can lead to communication problems in interdisciplinary communication.

5.2.3 Redirection of Effort Toward Mediation Tools

Although the research clearly shows that word-sense disambiguation is an exercise in futility, effort continues to revolve around a "magic machine" that will disambiguate all of the knowledge that the human race has accumulated.

While we should not necessarily discourage research that enhances artificial intelligence and ontology learning systems, some of the effort should be redirected to mediation tools that can effectively manage current terminological dissonance.

5.2.4 Inline Mediation Tools

There are many situations where a stand-alone mediation tool is simply too obtuse to be of any real usefulness. Ideally users could receive assistance as they communicate that would help them prevent miscommunications caused by terminological dissonance.

Text-based communication increases the potential for semantic miscommunication. One way to mediate our text-based messages is an inline tool that highlighted potentially dissonant terms as the message is typed. If the writer clicked on a highlighted term, the tool would reveal several different definitions of the term in question. The writer could then choose to clarify their meaning in their message, or alternatively they could annotate the term with a chosen concept they intended for its meaning. The writer could also add their own annotation if none of the selections were appropriate for the message. If the final sent message contained term annotations, the recipient would be able to click on highlighted terms in the message and see the sender's desired definition for selected terms.

6 Survey

6.1 Survey Format

The first component of the survey was list of top twenty polysemous terms produced by Termediator. This list is as follows, in order of most semantic clusters to least semantic clusters. The LSI clustering-linkage method produced this list.

1. data
2. standard
3. interface
4. graphic
5. filter
6. archive
7. access
8. user
9. template
10. signature
11. redundancy
12. queue
13. process
14. post

15. parameter

16. node

17. interactive

18. firewall

19. feedback

20. cut

The second component of the survey was a list of randomly generated terms. This list and other randomized components were produced by a Python program detailed in *Appendix D.* The criteria for the inclusion of these terms was: that they originated from an Information Technology glossary, and that they had five or more concepts.

1. NTFS (New Technology File System)

2. E-commerce

3. API (Application Program Interface)

4. Kilobyte

5. SDRAM (Synchronous Dynamic Random Access Memory)

6. Typeface

7. ActiveX

8. WEP (Wired Equivalent Privacy)

9. AJAX (Asynchronous JavaScript and XML)

10. AOL (America Online)

11. DHTML (Dynamic HTML)

12. RFID (Radio-frequency Identification)

13. Dial-up

14. Pharming

15. Groupware

16. SQL (Structured Query Language)

17. Boolean

18. Petabyte

19. Wi-fi

20. CPU (Central Processing Unit)

The Termediator list was shuffled into a random order. The randomization was accomplished through a Python program detailed in *Appendix D.* The randomized Termediater list is as follows:

1. node

2. interface

3. feedback

4. firewall

5. signature

6. template

7. process

8. interactive

9. archive

10. cut

11. post

12. queue

13. access

14. graphic

15. data

16. standard

17. redundancy

18. user

19. parameter

20. filter

The above list was then paired with the second list. As the second list's order was already random, that same order was used in the pairing. In the following list of pairs, the term before the dash is from Termediator and the second term is from the random generator.

1. node — NTFS (New Technology File System)

2. e-commerce — interface

3. feedback — API (Application Program Interface)

4. firewall — kilobyte

5. signature — SDRAM (Synchronous Dynamic Random Access Memory)

6. template — typeface

7. process — ActiveX

8. interactive — WEP (wired equivalent privacy)

9. archive — AJAX (Asynchronous JavaScript and XML)

10. cut — AOL (America Online)

11. post — DHTML (dynamic HTML)

12. queue — RFID (Radio-frequency Identification)

13. access — dial-up

14. graphic — pharming

15. data — groupware

16. standard — SQL (structured query language)

17. redundancy — boolean

18. user — petabyte

19. parameter - Wi-Fi

20. filter - CPU (central processing unit)

Finally, the order of each term within its own pair was randomized for the survey. The term pairs position order as presented to survey participants is below.

1. node — NTFS (New Technology File System)

2. e-commerce — interface

3. API (application program interface) — feedback

4. firewall — kilobyte

5. signature — SDRAM (synchronous dynamic random access memory)

6. typeface — template

7. process — activeX

8. WEP (wired equivalent privacy) — interactive

9. AJAX (asynchronous JavaScript and XML) — archive

10. cut — AOL (America Online)

11. post — DHTML (dynamic HTML)

12. queue — RFID (radio-frequency identification)

13. dial-up — access

14. graphic — pharming

15. groupware — data
16. SQL (structured query language) — standard
17. boolean — redundancy
18. user — petabyte
19. Wi-Fi — parameter
20. filter — CPU (central processing unit)

6.2 Results

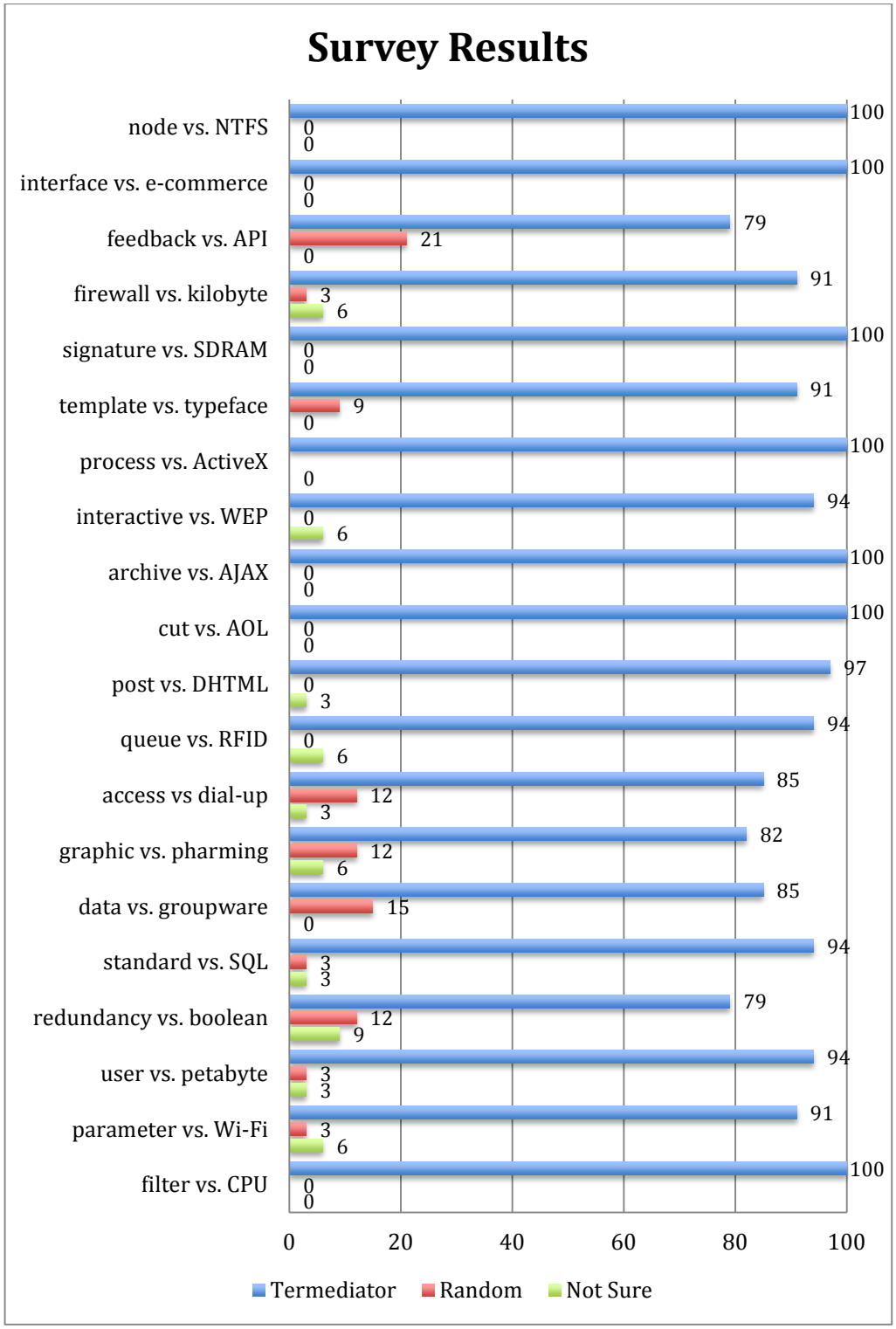
The following graph visually illustrates the results of the survey. Participants who selected the same term as Termediator as "most polysemous" are represented by the blue bars located next to each term. Participants who selected the random term as "most polysemous" are represented by the red bar located next to each term. A blue and red bar pair is given for each term pairing.

The results show survey participants overwhelmingly agree that terms generated by Termediator using LSI average clustering are more polysemous than randomly generated terms from the same source dataset.

The highest vote received for a randomly generated term was 21% for *API* in comparison to the term *feedback*. The highest vote for "not sure" was 9% when comparing the Termediator term *redundancy* versus the random term *boolean*.

A clear majority favors Termediator in every term pairing. Agreement by participants with Termediator ranged from 79% to 100% per term pair.

Table 6. Survey Results



6.3 Survey Conclusions

The survey results strongly suggest that human agents agree with Termediator's polysemy ranking of a term rather than a randomly generated term from the same terminology dataset. Termediator's ranking in the survey is produced by the previously described LSI average semantic clustering algorithm. The scope of this conclusion is limited to terms within a single domain. The domain of said terms were identified by the author of the term's source glossary.

As users overwhelmingly agreed with Termediator's polysemy ranking, this indicates the ability of automated tools to identify terms with a high dissonance potential. This validation of the fundamental mechanism of Termediator paves the way for additional implementations of the tool in real-world academic and professional settings.

7 Summary

The first objective of this thesis was to define *terminology mediation* and *terminology mediation tools* as an area of research within an Information Technology context. The introduction introduced the general problems of miscommunication and then the concept of "terminological dissonance" or "pure miscommunication" was introduced. Terminological dissonance is when miscommunication occurs between educated and sympathetic parties (thus discarding social factors in miscommunication such as hostility or ignorance).

Core terms such as *synonymy, polysemy,* and *knowledge base* were introduced along with their basic definitions. As the Termediator tool's data is extracted from ontological sources such as glossaries and taxonomies, the ontological spectrum was illustrated. The historical usages of ontologies to define, document, and standardize knowledge were discussed.

Once the basic vocabulary was defined, the thesis proceeded to introduce the idea of terminological ambiguity. Factors that produce ambiguity in language were introduced. These factors are summarized as follows: synonymy, polysemy, user enjoyment, user advantage, language evolution, limited vocabulary capacity, and resistance to terminological change.

As the existence of a high level of ambiguity eventually leads to terminological dissonance, the current state of terminological dissonance was introduced in three primary contexts: single domain scope, interdisciplinary work, and text-based communication. It was found that terminological dissonance still occurs when limited to the scope of one domain—this means that practitioners in a field fail to standardize the use of their own terminology. Interdisciplinary work is on a permanent forward trajectory, and this increases the amount of

ambiguity and magnifies the effects of ambiguity as large loads of information is shared between collaborating parties. Text-based communication is now ubiquitous in professional, academic, and personal life, and previous studies have found that text-based messages dramatically increase the number and intensity of miscommunications.

With the acknowledgement that terminological dissonance is an old problem, previously proposed solutions were described, such as: denial of ambiguity, disambiguated language schemes, bodies of terms, ontology creation, automated ontology processing and word-sense disambiguation, and top-down standardization failure. The overriding conclusion from this research was that each solution attempted to either ignore or remove ambiguity. As it was determined that ambiguity is an inevitable, necessary, and even enjoyable feature of language, the conclusion was made that research from a perspective of mediation rather than removal was necessary.

The penultimate section of the literature review introduced the "precision problem". As each discipline attempts to standardize their own language, they often make communication with collaborators outside their disciplinary silo more difficult. When disciplines are increasing terminological precision in parallel, the dissonance in cross-disciplinary communication is magnified.

The final discussion in the literature review introduced tools intended to manage and mediate terminological dissonance without disambiguating the language. These tools focused on automating ambiguity identification processes and assisting the human agent in making communicational decisions. The tools covered were CRTCOL, SDL Multi-Term, and Termediator.

The second objective of this thesis was to verify the contents and implementation of the Termediator tool. The history of the tool was documented from its initial prototype as a term browser to its current state as an identifier of synonymy and polysemy. The original prototype parsed and normalized the ISO/IEC 24765 data into Python 'dict' data structures. A Django interface paired with the dictionaries granted web access to browse the terms and concepts.

The next iteration of the prototype utilized a dataset of hundreds of glossaries sourced from eighteen disciplines of study. The terms and concepts now number in the thousands. The data was parsed through Python data scrapers and normalized into XML format.

Synonymy identification mechanisms were added to identify when a concept was semantically similar to another concept. To identify synonymous terms, a vector model "similarities matrix" was created to compare every concept with every other concept; each relationship was then assigned a similarity ranking. Drilling down into a concept revealed the correlated similarities table.

Polysemy identification mechanisms were implemented soon after the synonymy component. Hierarchic agglomerative methods were utilized to create semantic clusters of concepts under a term. Three different similarity algorithms (cosine, LSI, and LDA) were implemented to produce proximity matrices between terms. These proximity matrices gave an estimation of similarity concept to concept.

Polysemy was further explored by creating clusters of concepts under a term. Building on prior research, average and complete linkage types were combined with LSI, LDA, and cosine to create these semantic clusters. It was hypothesized that the most polysemous terms would also have the most clusters.

The clustering data became useful with the addition of a candidate threshold, which determines where the clustering boundaries occur. At the best threshold, concepts are divided appropriately into their most relevant semantic groups.

A web tool was created that allowed immediate visual feedback as the user changed the candidate threshold. This tool provided insight into a potential measure of interest. Terms easily identifiable as polysemous tended to have more clusters at higher threshold values than less polysemous terms.

Guided by this insight, we ran all three clustering methods on every term and recorded the corresponding convergence points. To further the analysis, we combined each convergence value with the mean. Graphing all of these convergence points simultaneously revealed the trend persists for all similarity measures and linkage types.

This data was then used to generate an "average convergence point" for each clustering and linkage combination. An average convergence point is the average value at which terms using that particular clustering-linkage combination converge all their concepts into a single cluster.

The matrix of average convergence points was then utilized to perform hierarchical clustering on each term in the compendium. The results were sorted by cluster frequency. This produced table of terms with the most clusters for each of the six clustering-linkage algorithms. The subsequent results are interesting because they include many terms that can be intuitively identified as polysemous terms. Words ranked highly in these results, such as "function," "process," and "resource," are fraught with potential for miscommunication.

After defining the methodology of synonymy and polysemy identification, the thesis listed examples of dissonance, in both modes, sourced from Termediator's output. Several

examples of synonymy identification were given with figures showing the actual output from the tool. Polysemy identification was then shown by tables of "top polysemous terms" that Termediator produced for each domain in the dataset. We looked at top polysemous terms for Information Technology, Computer Science, and so on.

The use case of Termediator in multidisciplinary dissonance identification was discussed. This was illustrated using cross-sections of multidisciplinary teams and their commonly used terms. Additional use cases in telecommunication and education were defined and verified.

Chapter 5 overviewed future developments for Termediator and dissonance research. The first suggestion was to attempt a redo at the failed crowdsourcing polysemy survey. Prior failures were documented and improvements, on both tool and human execution sides, were offered.

The second development suggested: extend the "warning list" feature in the polysemy tool to the synonymy tool. This increases Termediator's usefulness to human agents exponentially. Conversely, the browsing feature for synonymy should include results from polysemy data. Overall, the synonymy and polysemy interfaces in Termediator should be seamlessly integrated as the tool moves from experimental to real-world use. The tool should also be made available on mobile devices. Sample UIs and suggestions for functionality of this integrated interface were provided.

The implementation of our current polysemy clustering methods has significant room for improvement. Termediator currently uses several hierarchical agglomerative clustering methods. Hierarchical agglomerative is partitional and does not work as well for polysemy as overlapping cluster models. Further research and experimentation should be done to potentially replace current clustering models with more sophisticated techniques.

Chapter 6 outlined the user survey that was conducted to verify if Termediator produced results in agreement with human intuition. The survey compared two lists of twenty terms each, drawn from the Information Technology dataset. One list was randomly generated while the other was produced by Termediator using the LSI-average clusering method. Juniors and seniors in the Capstone class hosted by BYU IT were asked to compare pairs of terms and rank one term as more polysemous. Appropriate randomization measures were taken to ensure that survey participants did not know the source origin of each term.

The results of the survey showed that participants overwhelmingly agreed that Termediator's chosen terms were more polysemous than randomly generated terms from the dataset. Agreement by participants with Termediator ranged from 79% to 100% per term pair. As users overwhelmingly agreed with Termediator's polysemy ranking, this indicates the ability of automated tools to identify terms with a high dissonance potential. This validation of the fundamental mechanism of Termediator paves the way for additional implementations of the tool in real-world academic and professional settings.

7.1 Conclusion

This research has introduced and defined terminology mediation as an area of research within Information Technology. The contents and implementation of the Termediator have been documented and verified in multiple contexts. The viability of Termediator for dissonance identification was further verified by a user survey where participants consistently chose Termediator's polysemy rankings over randomly generated terms.

REFERENCES

1st Impression Printing Waterloo. (2014). *Printing Glossary*. Retrieved from http://1stimpressionprinting.ca/printing-graphics/printing-glossary

Aalst, W. M., & Hee, K. v. (2004). *Workflow Management: Models, Methods, and Systems* . The MIT Press.

Amberlight. (2015, February 20). *A to Z User Experience Glossary*. Retrieved from User Experience: Business Impact: http://www.amber-light.co.uk/user-experience-glossary.php

Banerjee, A., Krumpelman, C., Basu, S., Mooney, R. J., & Ghosh, J. (2005, August). Model-Based Overlapping Clustering. *International Conference on Knowledge Discovery and Data Mining* , 532-537.

BeautifulSoup API. (2014). *BeautifulSoup*. (Crummy Software) Retrieved from http://www.crummy.com/software/BeautifulSoup/

Bleeping Computer LLC. (2014). *The Computer Glossary*. Retrieved from http://www.bleepingcomputer.com/glossary/

Bromberg, N. (2014). *Michigan Cites 'Miscommunication' in Advertising Free Tickets for Coke Purchases.* Yahoo News.

Buitelaar, P., Cimmiano, P., & Magnini, B. (2005). Ontology Learning From Text: An Overview. In P. C. . Buitelaar (Ed.), *Ontology Learning from Text: Methods, Evaluation and Applications.* Amsterdam: IOS Press.

Byron, K. (2006). Carrying Too Heavy a Load? The Communication and Miscommunication of Emotion by Email. *Academy of Management Review , 33* (2), pp. 309-327.

Cargill, C. F. (2011). Why Standardization Efforts Fail. *Journal of Electronic Publishing , 14* (1).

Chapman, C. (2009). *Web Design Industry Jargon: Glossary and Resources.* Glossary.

Christiansen, M. H., & Kirby, S. (2003). Language Evolution: Consensus and Controversies. *TRENDS in Cognitive Sciences , 7* (7).

Clarke, P., & O'Connor, R. (2010). Harnessing ISO/IEC 2007 to Examine the Extent of SPI Activity in an Organization. *Systems, Software and Services Process Improvement: 17th European Conference. Communications in Computer and Information Science.* (p. 30). EuroSPI 2010.

Cognisco. (2008). *$37 Billion: Counting the Cost of Employee Misunderstanding.* London: International Data Corporation.

Computer Support Group, Inc. (2014). *Computer, Telephone, & Electronics Industry Glossary*. (CSG Networks) Retrieved from http://www.csgnetwork.com/glossary.html

Condamines, A. (2010). Variations in Terminology: Application to the Management of Risk Related to Language Use in the Workplace. *Terminology , 16* (1), 30-50.

Condillac, E. B. (1776). *Le Commerce et le Gouvernement.*

Conquest Graphics. (2005-2014). *Glossary of Graphic Design Terms*. Retrieved from http://www.conquestgraphics.com/Help-Center/Glossary-of-Graphic-Design-Terms

Cushman, J. H. (1990). *Avianca Flight 52: The Delays That Ended in Disaster.* New York Times.

Department of Education and Communities and Charles Sturt University. (2014). *Common Information Technology Terms*. (State of New South Wales) Retrieved from http://hsc.csu.edu.au/info_tech/glossary/2327/common_terms.html

Department of Health and Human Services. (2013, May 22). *Doctors and Hospitals' Use of Health IT More Than Doubles Since 2012*. Retrieved from Department of Health and Human Services: http://www.hhs.gov/news/press/2013pres/05/20130522a.html

Department of Health and Human Services. (2012). *Redesigning Health Professions Education and Practice to Prepare the Interprofessional Team to Care for Populations.* Department of Health and Human Services. Advisory Committee on Interdisciplinary, Community-Based Linkages (ACICBL).

Dictionaries, T. E. (Ed.). (2002). *The American Heritage Stedman's Medical Dictionary.* Houghton Mifflin Company.

Django Project. (2013). *Django.* Retrieved from http://www.djangoproject.com

Ekstrom, J. J. (2012). Experience with a Cross-Disciplinary Aggregated Glossary of Technical Terms. *SIGITE* .

El Bucanero. (1996-2014). *Hacking Glossary*. Retrieved from http://www.bucanero.com.ar/paginas/hacking-glossary/

ElementTree API. (2014). *The ElementTree Library*. (The Python Standard Library) Retrieved from https://docs.python.org/2/library/xml.etree.elementtree.html

Fluit, C., Horst, H. t., Meer, J. v., Sabou, M., & Mika, P. (2003). Spectacle. (J. Davies, D. Fensel, & F. v. Harmelen, Eds.) *Towards the Semantic Web: Ontology-Driven Knowledge Management* .

Fukumoto, F., & Tsuji, J. (1994). *Automatic Recognition of Verbal Polysemy.* UMIST, Centre for Computational Linguistics. Manchester: UMIST.

Goertzel, B. (2013). Lojban++: An Interlingua for Communication between Humans and AGIs . In K.-U. Kuhnberger, S. Rudolph, & P. Wang (Eds.), *Artificial General Intelligence* (Vol. 7999, pp. 21-30). Beijing, China: AGI.

Gong, T., Shuai, L., & Comrie, B. (2014). Evolutionary linguistics: theory of language in an interdisciplinary space. *Language Sciences , 41*, 243-253.

Grice, H. P. (1975). Logic and Conversation. (P. Cole, & J. L. Morgan, Eds.) *Syntax and Semantics , 3*, 41-58.

Griffen, F., Stephens, L., Alexander, J., Bailey, H. R., Maizel, S., Sutton, B., et al. (2007). The American College of Surgeons' Closed Claims Study: New Insights for Improving Care. *Journal of the American College of Surgeons , 204* (4), 561-569.

Gruber, T. (1993). A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition , 5* (2), pp. 199-220.

Harrower, T. (2007). *The Newspaper Designer's Handbook* (6 ed.). McGraw-Hill Humanities/Social Sciences/Languages.

Hessels, L. K., & Lente, H. v. (2008). Re-thinking new knowledge production: A literature review and a research agenda. *Research Policy , 37*, pp. 740-760.

Hjelm, H. (2009). *Cross-Language Ontology Learning: Incorporating and Exploiting Cross-Language Data data in the Ontology Learning Process.* Stockholm University.

IEEE. (2010, 12). SEVOCAB. *IEEE Computer Society* .

Jian, X., & Ah-Hwee, T. (2009). *CRCTOL: A Semantic-Based Domain Ontology Learning System.* Nanyang Technological University, School of Computer Engineering. Wiley InterScience.

Kasser, J. E. (2007). *A Framework for Understanding Systems Engineering.* BookSource Publishing.

Kaufmann, C. (2014). *Emergency Response Delayed After Man's Fatal Collapse.* The Des Moines Register.

KeyCorp. (2014). *Information Security from A-Z*. Retrieved from https://www.key.com/about/security/bank-information-security-glossary.jsp

Khosrow-Pour, M. (Ed.). (2005). *Dictionary of Information Science and Technology* (Vol. 1). IGI Global.

Klepousniotou, E., & Baum, S. R. (2007). Disambiguating the Ambiguity Advantage Effect in Word Recognition: An Advantage for Polysemous but not Homonymous. *Journal of Neurolinguistics* (20), 1-4.

LabAutoPedia. (2009). *LabAutoPedia Glossary.* Glossary.

Legatt, H. (2011, May). *Microsoft: Email Use Continues to Rise*. Retrieved from BizReport: http://www.bizreport.com/2011/05/microsoft-email-use-continues-to-rise.html

Lehrer, A. (1990). Polysemy, Conventionality and the Structure of the Lexicon. *Cognitive Linguistics* (1), 207-246.

Lundrigan, D. (2013). *Moving From Miscommunication to Communication.* High Performance Leadership.

Miller, L. C., Jones, B. B., Graves, R. S., & Sievert, M. C. (2010). Merging Silos: Collaborating for Information Literacy. *The Journal of Continuing Education in Nursing , 41* (6), 267-272.

Miller, L. (2000, November 22-23). *Ontologies and Metadata*. (Semantic Web Technologies Workshop)

Munroe, R. Lojban. *XKCD.* MA.

Nationwide Payment Solutions. (2010). *Glossary of Industry Terms*. Retrieved from http://www.nationwidepaymentsolutions.com/glossary.asp

Nerlich, B., & Clarke, D. (2001). Ambiguities We Live By: Towards a Pragmatics of Polysemy. *Journal of Pragmatics* (33), 1-20.

Network Management Solutions. (2014). *NMS IT Glossary.* Glossary.

Noy, N., & McGuiness, D. (2000). *Ontology Development 101: A Guide to Creating Your First Ontology.* Retrieved from Stanford Protege Project: http://protege.stanford.edu/publications/ontology_development/ontology101-noy-mcguinness.html

Pepperdine University. (2014). *Information Security Glossary*. Retrieved from http://community.pepperdine.edu/it/security/resources/security-glossary.htm

Pidcock, W. (2003, January 15). *What Are the Differences Between a Vocabulary, a Taxonomy, a Thesaurus, an Ontology, and a Meta-Model?* (Boeing) Retrieved from MetaModel: http://infogrid.org/trac/wiki/Reference/PidcockArticle

Post, J. (2011). *Management Information Systems Glossary*. Retrieved from http://www.jerrypost.com/MIS/MISGlossary.html

Project Management Institute. (2000). *A Guide to the Project Management Body of Knowledge.* Newton Square, Pennsylvania.

Puntoni, S., Schroeder, J. E., & Riston, M. (2010). Meaning Matters: Polysemy in Advertising. *Journal of Advertising , 39* (2), 51-64.

Quiroga-Clare, C. (2003). Language Ambiguity: A Curse and a Blessing. *Translation Journal* .

R.S. Pressman & Associates, Inc. (2001-2010). *An Abbreviated Software Engineering Glossary.* Glossary.

Richards, J., Riley, O., Ekstrom, j., & Tew, K. (2013). Termediator: Early Studies in Terminological Mediation Between Disciplines. ACM RiLT.

Riley, O. (2013, April). Termediator-II: Identification of Interdisciplinary Term Ambiguity Through Hierarchical Cluster Analysis". *Brigham Young University* . Provo, UT, USA.

Riley, O., Richards, J., Ekstrom, J., & Tew, K. (2014). Measuring Term Polysemy With Semantic Clustering. ACM SIGITE.

Segen, J. C. (2005). *The Dictionary of Modern Medicine.* McGraw-Hill Medical.

Smiley, P. (1986). *Gender-Linked Miscommunication in "Hills Like White Elephants".* University of Wisconsin. The Hemingway Review.

Sommerville, I. (2010). *Software Engineering Glossary*. (Addison-Wesley) Retrieved from Systems, Software, and Technology: http://ifs.host.cs.st-andrews.ac.uk/Resources/Notes/General%20SE/glossary.pdf

Starmer, A., Spector, N., Srivastava, R., West, D., Rosenbluth, G., Allen, A., et al. (2014). Changes in Medical Errors After Implementation of a Handoff Program. *The New England Journal of Medicine , 371*, 1803-1812.

Subramanian, G., & Minow, M. (2013, January 29). *Interdisciplinary Collaborations on the Rise*. Retrieved from Harvard Law Today: http://today.law.harvard.edu/interdisciplinary-collaborations-on-the-rise/

SyBase. (2006). *SyBooks Online BPM Glossary*. Retrieved from SyBooks: http://infocenter.sybase.com/archive/index.jsp?topic=/com.sybase.stf.powerdesigner.docs_12.1.0/html/bpug/bpugp531.htm

Time Warner Communications. (2014). *Miscommunication Blamed for Fatal U.S. Mistake in Afghanistan.* Time Warner.

Uschold, M., & Gruninger, M. (2004). Ontologies and Semantics for Seamless Connectivity. *ACM SIG-MOD , 33* (4), pp. 58-64.

WestNet, Inc. (2010). *IT Glossary*. Retrieved from http://glossary.westnetinc.com/glossary.php

Willett, P. (2000). Recent Trends in Hierarchic Document Clustering: A Critical Review. *Information Processing & Management , 24* (5), 577-597.

Wong, W., Liu, W., & Bennamoun, M. (2012, August). Ontology Learning From Text: A Look Back Into The Future. *ACM Computing Surveys , 44* (4), p. 20.

Workflow Patterns. (2010). *Workflow Patterns Terms*. Retrieved from http://www.workflowpatterns.com/patterns/data/visibility/wdp1.php

Wright, S. E., & Budin, G. (2001). *Handbook of Terminology Management: Application-Oriented Terminology Management* (Vol. 2). John Benjamins Publishing Company.

APPENDICES

APPENDIX A. EXAMPLE OF A GLOSSARY PARSER FOR PDF

```
#!/usr/local/bin python
import re
import lxml
from lxml import etree as ET
import os
from pprint import pprint as pp
def main():
      # INPUT FILE INSTRUCTIONS
      # Open PDF in Adobe Acrobat Pro
      # File -> Save As Other -> More Options -> XML 1.0
      # Click Settings... button
      # Encoding set to ISO-Latin-1
      # Unchecked: Generate bookmarks
      # Checked: Generate tags for untagged files
      # Unchecked: Generate images and all options below it
      # XML FILE INSTRUCTIONS
      # Compact file in BBEdit

      xml_file = os.path.abspath(__file__)
      xml_file = os.path.dirname(xml_file)
      xml_file = os.path.join(xml_file, "../sourceXML/Business Process
Management Center of Excellence Glossary.xml")

      out_file = os.path.abspath(__file__)
      out_file = os.path.dirname(out_file)
      out_file = os.path.join(out_file, "../resultXML/Business Process
Management Center of Excellence Glossary - STANDARD.xml")

      try:
            parser = ET.XMLParser(remove_blank_text=True, ns_clean=True)
            tree = ET.parse(xml_file, parser)
      except Exception, inst:
            print "Unexpected error opening %s: %s" % (xml_file, inst)
            return

      root = tree.getroot()

      counter = 0

      Glossary = ET.Element("Glossary")

      for OriginNameTemp in
root.iter('{http://purl.org/dc/elements/1.1/}title'):
            for SpecificTemp in
OriginNameTemp.iter('{http://www.w3.org/1999/02/22-rdf-syntax-ns#}li'):
                  Glossary.set("OriginName", SpecificTemp.text)
```

```
      Glossary.set("OriginURL",
"http://www.ftb.ca.gov/aboutftb/projects/itsp/bpm_glossary.pdf")

      for OriginAuthorTemp in
root.iter('{http://purl.org/dc/elements/1.1/}creator'):
            for SpecificTemp in
OriginAuthorTemp.iter('{http://www.w3.org/1999/02/22-rdf-syntax-ns#}li'):
                  Glossary.set("OriginAuthor", SpecificTemp.text)

      Glossary.set("OriginDomain", "Business Process Management")


      # Keeps track of iterations because Terms and Concepts are paragraphs
one after the other
      counter = 0
      even = False;

      TermsAndConcepts = {}
      TermTemp = ET.Element("Nothing")

      for paragraph in root.iter('P'):
            if paragraph.text:
                  paragraphLength = len(paragraph.text)
                  # Skips letter titles and blank paragraphs, or paragraphs
containing multiple terms
                  if paragraphLength != 1 and paragraphLength != 0:
                        # Skip "no terms" entry for letter titles
                        if "No terms at this time" in paragraph.text:
                              continue;
                        # Check counter. Even is a Concept, Odd is a Term
                        counter += 1
                        if counter%2 == 0:
                              even = True
                        else:
                              even = False
                        if even == False:
                              # Store Term for next iteration
                              TermTemp = paragraph
                              #For "see also" Terms that don't have their own
concept
                              if ", see" in TermTemp.text:
                                    Entry = ET.Element("Entry")
                                    Term = ET.Element("Term")
                                    TermAnnotation =
ET.Element("TermAnnotation")
                                    Concept = ET.Element("Concept")
                                    Term.text = TermTemp.text
                                    Concept.text = ''
                                    # Group 1 is Term
                                    # Group 2 is SeeAlso
                                    match = re.search(r'([^.]*), see
([^.]*)', TermTemp.text)
                                    if match:
                                          Term.text = match.group(1)
                                          TermAnnotation.set("type",
"SeeAlso")
```

```
                                          TermAnnotation.text =
match.group(2)
                                          Concept.text = "See " +
match.group(2)
                                    Term.append(TermAnnotation)
                                    Entry.append(Term)
                                    Entry.append(Concept)
                                    Glossary.append(Entry)
                                    counter += 1
                                    continue
                              else:
                                    Entry = ET.Element("Entry")
                                    Term = ET.Element("Term")
                                    Concept = ET.Element("Concept")
                                    Term.text = TermTemp.text
                                    Concept.text = paragraph.text
                                    Entry.append(Term)
                                    Entry.append(Concept)
                                    Glossary.append(Entry)
                  else:
                        #Multiple concept terms
                        counter += 1
                        Entry = ET.Element("Entry")
                        Term = ET.Element("Term")
                        Term.text = TermTemp.text
                        Entry.append(Term)
                        for   concept in paragraph.iter('LBody'):
                              Concept = ET.Element("Concept")
                              Concept.text = concept.text
                              Entry.append(Concept)
                        Glossary.append(Entry)


            NewTree = ET.ElementTree(Glossary)
            NewTree.write(out_file)

def check(number):
    if number%2==0:
        even = True;
    else:
        even = False;

if __name__ == "__main__":
      # Someone is launching this directly
      main()
```

APPENDIX B. EXAMPLE OF A GLOSSARY PARSER FOR HTML

```
# coding: utf-8
#!/usr/local/bin python

import re
import lxml
from lxml import etree as ET
from lxml.html import fromstring, tostring
import os
from pprint import pprint as pp
from lxml import html as HT
from bs4 import BeautifulSoup, NavigableString
from bs4 import UnicodeDammit

def main():

      # INPUT FILE: HTML saved from Google Chrome as Web Page, HTML Only

      xml_file = os.path.abspath(__file__)
      xml_file = os.path.dirname(xml_file)
      xml_file = os.path.join(xml_file, "../sourceXML/Usability 247.html")

      out_file = os.path.abspath(__file__)
      out_file = os.path.dirname(out_file)
      out_file = os.path.join(out_file, "../resultXML/Usability 247 -
STANDARD.xml")

      try:
            #HTML Parse
            soup = BeautifulSoup(open(xml_file), "lxml", from_encoding="utf-
8")
      except Exception as inst:
            print "Unexpected error opening %s: %s" % (xml_file, inst)
            return

      #root = tree.getroot()

      Glossary = ET.Element("Glossary")
      Glossary.set("OriginName", "Usability 247")
      Glossary.set("OriginURL",
"http://www.usability247.com/resources/usability-glossary/")
      Glossary.set("OriginAuthor", "Usability 247")
      Glossary.set("OriginDomain", "User Experience Design")

      Entry = ET.Element("Nothing1")
      Term = ET.Element("Nothing2")
      Concept = ET.Element("Nothing3")
```

```
	glossNode = soup.find(id="content")

	for child in glossNode.findChildren():
		if child.name == "h3":
			Entry = ET.Element("Entry")
			Glossary.append(Entry)
			Term = ET.Element("Term")
			Entry.append(Term)

			parenMatch = re.search("(.*) \((.*)\)", child.text)
			if parenMatch:
				Term.text = parenMatch.group(1)
				hold = parenMatch.group(2)
				TermAnnotation = ET.Element("TermAnnotation")
				TermAnnotation.set("type", "SeeAlso")
				Term.append(TermAnnotation)
				TermAnnotation.text = hold
			else:
				Term.text = child.text
		elif child.name == "p":
			Concept = ET.Element("Concept")
			Entry.append(Concept)
			Concept.text = child.text

		"""if child.find("span"):
			temp = ""
			for pos,string in enumerate(child.stripped_strings):
				if pos == 0:
					Entry = ET.Element("Entry")
					Glossary.append(Entry)
					Term = ET.Element("Term")
					Entry.append(Term)

					parenMatch = re.search("(.*) \((.*)\)", string)
					commaMatch = re.search(",", string)

					# Are there synonyms in parentheses next to the
first term definition?
					if parenMatch:
						Term.text = parenMatch.group(1)
						hold = parenMatch.group(2)
						pcomMatch = re.search(",", hold)
						# Is there a comma delimited list inside
the parentheses?
						if pcomMatch:
							split = re.split(",", hold)
							for item in split:
								TermAnnotation =
ET.Element("TermAnnotation")
								Term.append(TermAnnotation)
								TermAnnotation.set("type",
"SeeAlso")
								TermAnnotation.text = item
						else:
							TermAnnotation =
ET.Element("TermAnnotation")
```

```
                                          TermAnnotation.set("type",
"SeeAlso")
                                          Term.append(TermAnnotation)
                                          TermAnnotation.text = hold
                                    # Is the term definition itself a comma
delimited list?
                                    elif commaMatch:
                                          split = re.split(",", string)
                                          for num,item in enumerate(split):
                                                # set first one in list as main
term definition
                                                # set others to TermAnnotations
                                                if num == 0:
                                                      Term.text = item
                                                else:
                                                      TermAnnotation =
ET.Element("TermAnnotation")
                                                      Term.append(TermAnnotation)
                                                      TermAnnotation.set("type",
"SeeAlso")
                                                      TermAnnotation.text = item
                                    else:
                                          Term.text = string
                              elif pos == 1:
                                    Concept = ET.Element("Concept")
                                    Entry.append(Concept)
                                    temp += string
                              else:
                                    temp += string
                        Concept.text = temp.lstrip(", ")"""

      NewTree = ET.ElementTree(Glossary)
      NewTree.write(out_file, encoding='utf-8')


if __name__ == "__main__":
      # Someone is launching this directly
      main()
```

APPENDIX C. MERGER PROGRAM

```
import os
from lxml import etree
import re

def createShortReference(reference):
    return reference.replace(" ","").replace("'","")

errorDescriptions = {
    "noneTerms" : "empty term tags",
    "emptyTerms" : "term tags with all text in annotations",
    "noneConcepts" : "empty concept tags",
    "emptyConcepts" : "concept tags with all text in annotations",
    "duplicateConcepts" : "concepts that are identical",
    "noneAnnotationType" : "annotations with missing 'type' attribute",
    "noneAnnotationText" : "empty annotation tags",
    "missingConcepts" : "terms with no concepts",
}
annotations = {}
concerns = {}
compendium = {}
SEvocab_sources = []
compendiumElement = etree.Element("GlossaryCompendium")
for filename in os.listdir("resultXML"):
      # merger fails if there is a & in OriginName

    print filename
    if filename.startswith('.'):
        continue
    tree = etree.parse("resultXML/"+filename)
    root = tree.getroot()
    originName = root.attrib["OriginName"]
    originURL = root.attrib["OriginURL"]
    originAuthor = root.attrib.get("OriginAuthor")
    originDomain = root.attrib["OriginDomain"]
    originID = createShortReference(originName)

    #Limit scope to terms in IT
    if not re.match("Information Technology", originDomain):
      continue

    concerns[originID] = {
        'noneTerms' : 0,
        'emptyTerms' : 0,
        'noneConcepts' : 0,
        'emptyConcepts' : 0,
        'duplicateConcepts' : 0,
        'noneAnnotationType' : 0,
```

```
        'noneAnnotationText' : 0,
        'missingConcepts': 0,
    }

    add_node = etree.SubElement
    glossaryRef = add_node(compendiumElement, "GlossaryRef", id=originID)
    originNameElement = add_node(glossaryRef, "OriginName")
    originNameElement.text = originName
    originURLElement = add_node(glossaryRef, "OriginURL")
    originURLElement.text = originURL
    originDomainElement = add_node(glossaryRef, "OriginDomain")
    originDomainElement.text = originDomain
    for x in tree.findall("Entry"):
        term = x.find("Term")
        termText = term.text
        if termText is None:
            concerns[originID]['noneTerms'] += 1
            continue
        # Limit to one word terms
        # Accepts words with variable characters, such as ".htaccess" or
"net-domain"
        if not re.match("^[\w.\-]+$", termText):
            continue

        termText = termText.strip()

        #Remove trailing periods
        if termText[-1] == ".":
            termText = termText[:-1]

        #Replace hyphens with spaces
        #termText = termText.replace("-"," ").strip()

        termKey = termText.lower()
        if not termKey:
            concerns[originID]['emptyTerms'] += 1
            continue
        if termKey not in compendium:
            compendium[termKey] = {}
            compendium[termKey]["term"] = termText.capitalize()
        if len(term) > 0:
            annotations[termKey] = []
            for annotation in term:
                annotationType = annotation.attrib.get("type")
                if annotationType is None:
                    concerns[originID]['noneAnnotationType'] += 1
                    continue
                annotationText = annotation.text
                if annotationText is None:
                    concerns[originID]['noneAnnotationText'] += 1
                    continue
                annotationText = " ".join(annotationText.strip().split())
                annotations[termKey].append((annotationType, annotationText))
        conceptsExist = False
        for y in x.findall("Concept"):
            conceptsExist = True
            conceptText = y.text
```

```
            if conceptText is None:
                concerns[originID]['noneConcepts'] += 1
                continue
            cleanText = conceptText.encode('ascii','ignore').strip()
            cleanText = " ".join(cleanText.split())
            if not cleanText:
                concerns[originID]['emptyConcepts'] += 1
                continue

            if originID == "SEVocab" and len(y) > 0:
                for annotation in y:
                    annotationType = annotation.attrib.get("type")
                    annotationText = "
".join(annotation.text.strip().split())
                    originName2 = annotationText
                    originID2 = createShortReference(originName2)
                    if annotationText not in SEvocab_sources:
                        SEvocab_sources.append(annotationText)
                        originAuthor2 = annotationText[:annotationText.find("
")]
                        if originAuthor2 == "A":
                            originAuthor2 = "PMI"
                        originURL2 = originURL
                        originDomain2 = originDomain
                        glossaryRef = add_node(compendiumElement,
"GlossaryRef", id=originID2)
                        originNameElement = add_node(glossaryRef,
"OriginName")
                        originNameElement.text = originName2
                        originURLElement = add_node(glossaryRef, "OriginURL")
                        originURLElement.text = originURL2
                        originDomainElement = add_node(glossaryRef,
"OriginDomain")
                        originDomainElement.text = originDomain2
                    if originName2 not in compendium[termKey]:
                        compendium[termKey][originName2] = [cleanText]
                    else:
                        if cleanText not in compendium[termKey][originName2]:

compendium[termKey][originName2].append(cleanText)
                        else:
                            #concerns[originID]['duplicateConcepts'] += 1
                            pass
            elif originName not in compendium[termKey]:
                compendium[termKey][originName] = [cleanText]
            else:
                if cleanText not in compendium[termKey][originName]:
                    compendium[termKey][originName].append(cleanText)
                else:
                    #concerns[originID]['duplicateConcepts'] += 1
                    pass
        if not conceptsExist:
            concerns[originID]['missingConcepts'] += 1
            continue

for term in compendium:
    if term[-1] == "s" and len(term) > 5 and term[:-1] in compendium:
```

```
        newTerm = term[:-1]
        if term.lower() in annotations:
            if newTerm.lower() in annotations:
                annotations[newTerm.lower()] += annotations[term.lower()]
            else:
                annotations[newTerm.lower()] = annotations[term.lower()]
            del annotations[term.lower()]
        for x in compendium[term]:
            if x == "term":
                continue
            if x in compendium[newTerm]:
                compendium[newTerm][x] += compendium[term][x]
            else:
                compendium[newTerm][x] = compendium[term][x]
        compendium[term] = [0]

for term in sorted(compendium):
    if len(compendium[term]) == 1:
        del compendium[term]
        continue
    entryElement = add_node(compendiumElement, "Entry")
    termElement = add_node(entryElement, "Term")
    termElement.text = compendium[term]["term"]
    del compendium[term]["term"]
    """
    if term in annotations:
        for annotation in annotations[term]:
            annotationElement = add_node(termElement, "TermAnnotation",
type=annotation[0])
            annotationElement.text = annotation[1]
    """
    for origin in compendium[term]:
        for concept in compendium[term][origin]:
            conceptElement = add_node(entryElement, "Concept")
            conceptElement.text = concept
            referenceElement = add_node(conceptElement, "ConceptAnnotation",
                                        type="Reference")
            referenceElement.text = createShortReference(origin)
output = '<?xml version="1.0"?>\n' + etree.tostring(compendiumElement,
                                                    pretty_print=True)
with open("glossary_it.xml",'w') as out_file:
        out_file.writelines(output)
error_output = "The following issues were detected:\n"
for origin in concerns:
    for error in concerns[origin]:
        if concerns[origin][error] != 0:
            error_output += "There were " + str(concerns[origin][error]) + "
issues with " + errorDescriptions[error] + " in " + origin + "\n"
with open("compendium_issues.txt","w") as out_file:
    out_file.writelines(error_output)
```

APPENDIX D. RANDOMIZATION PROGRAM

```
import re
import lxml
from lxml import etree as ET
from lxml.html import fromstring, tostring
import os
from pprint import pprint as pp
from lxml import html as HT
from bs4 import BeautifulSoup, NavigableString
from bs4 import UnicodeDammit

import random
from random import shuffle
import numpy

#glossary = etree.parse("glossary.xml")

soup = BeautifulSoup(open("glossary_it.xml"), "lxml", from_encoding="utf-8")

results = []

for f_entry in soup.find_all("entry"):
      counter = 0
      for f_concept in f_entry.find_all("concept"):
            counter += 1
      if counter > 4:
            f_term = f_entry.find("term")
            results.append(f_term)
      #print f_term

results_trimmed = []
for x in range(20):
      #print (random.choice(results)).text
      results_trimmed.append((random.choice(results)).text)

for item in results_trimmed:
      print item

print " "
print " "

termediator = [["data", ""], ["standard", ""], ["interface", ""], ["graphic",
""],
["filter", ""], ["archive", ""], ["access", ""], ["user", ""], ["template",
""],
["signature", ""], ["redundancy", ""], ["queue", ""], ["process", ""],
["post", ""],
["parameter", ""], ["node", ""], ["interactive", ""], ["firewall", ""],
```

```
["feedback", ""], ["cut", ""]]
random.shuffle(termediator)

#randomly pair the two lists
for i in range (20):
      termediator[i][1] = results_trimmed[i]

#shuffle column position
map(numpy.random.shuffle, termediator)

for elem in termediator:
      print elem
```