Volume 10, Issue 5
October 2012
ISSN: 1545-679X

INFORMATION SYSTEMS EDUCATION JOURNAL

In this issue:

The **Information Systems Education Journal** (ISEDJ) is a double-blind peer-reviewed academic journal published by **EDSIG**, the Education Special Interest Group of AITP, the Association of Information Technology Professionals (Chicago, Illinois). Publishing frequency is six times per year. The first year of publication is 2003.

ISEDJ is published online (http://isedjorg) in connection with ISECON, the Information Systems Education Conference, which is also double-blind peer reviewed. Our sister publication, the Proceedings of ISECON (http://isecon.org) features all papers, panels, workshops, and presentations from the conference.

The journal acceptance review process involves a minimum of three double-blind peer reviews, where both the reviewer is not aware of the identities of the authors and the authors are not aware of the identities of the reviewers. The initial reviews happen before the conference. At that point papers are divided into award papers (top 15%), other journal papers (top 30%), unsettled papers, and non-journal papers. The unsettled papers are subjected to a second round of blind peer review to establish whether they will be accepted to the journal or not. Those papers that are deemed of sufficient quality are accepted for publication in the ISEDJ journal. Currently the target acceptance rate for the journal is about 45%.

Information Systems Education Journal is pleased to be listed in the 1st Edition of Cabell's Directory of Publishing Opportunities in Educational Technology and Library Science, in both the electronic and printed editions. Questions should be addressed to the editor at editor@isedj.org or the publisher at publisher@isedj.org.

INFORMATION SYSTEMS EDUCATION JOURNAL

Editors

ISEDJ Editorial Board

Problem Solving Frameworks for Mathematics and Software Development

Kirby McMaster
kmcmaster@weber.edu
CSIS Dept, Fort Lewis College
Durango, CO 81301, USA

Samuel Sambasivam
ssambasivam@apu.edu
CS Dept, Azusa Pacific University
Azusa, CA 91702, USA

Ashley Blake
ablaketx@hotmail.com
Scribblin' Sisters
Houston, TX 77094, USA

Abstract

In this research, we examine how problem solving frameworks differ between Mathematics and Software Development. Our methodology is based on the assumption that the words used frequently in a book indicate the mental framework of the author. We compared word frequencies in a sample of 139 books that discuss problem solving. The books were grouped into three categories: Traditional Math, Applied Math, and Software Development. We obtained a list of the most frequent words in each category, and used these lists to describe three problem solving frameworks. Applied Math uses models and algorithms to solve problems. Traditional Math is more concerned with proving theorems. In the Software Development framework, customers provide the problem, and models and algorithms are used to create a software solution. Our findings have relevance in the development of approaches for teaching problem solving in Mathematics and Software Development courses.

Keywords: problem, solution, framework, model, algorithm, mathematics, software.

1. INTRODUCTION

A monkey and a banana are placed in a room. The monkey desires the banana, but the banana is high overhead. The room also contains a box. If the monkey moves the box and climbs on it, the banana can be reached. This is one version of a classic *problem solving* situation in Artificial Intelligence (Bratko, 2001).

Scientific activities that demonstrate problem solving have been performed for centuries. Archimedes was able to determine if a king's crown was solid gold. Newton developed Calculus, anecdotally to explain why an apple fell on his head. Attributes of problem solving have been studied in fields such as Psychology, Medicine, Warfare, Management, and Engineering.

One issue often mentioned is whether problem solving can be expressed in terms of a single set of general principles, or if unique processes are required in different knowledge domains. In this paper, our primary focus is on problem solving in Mathematics (Math) and Software Development (SD). Does a single framework for problem solving apply to Math and SD, or do these academic disciplines solve problems in different ways?

Problem Solving in Math

Mathematics encompasses a large number of subject matter areas, such as Algebra, Calculus, Geometry, Differential Equations, and Number Theory. Within these areas, there are different levels of emphasis on problem solving and theorem proving. In his classic book *How to Solve It*, Polya (1945) promotes methods of solving problems in Math:

> Studying the methods of solving problems, we perceive another face of mathematics. Yes, mathematics has two faces; it is the rigorous science of Euclid, but it is also something else. Mathematics presented in the Euclidean way appears as a systematic, deductive science; but mathematics in the making appears as an experimental, inductive science. Both aspects are as old as the science of mathematics itself.

Almost fifty years later, Velleman (1994) wrote a book called *How to Prove It*, in which he discusses the same two faces of mathematics, but with a preference for constructing proofs:

> This textbook will prepare students to make the transition from solving problems to proving theorems by teaching them the techniques needed to read and write proofs.

The priority in each Math field can be on solving Math problems, or it can be on using Math to solve *real world* problems. Polya's book and Velleman's book spend most of their coverage on solving (or proving) Math problems. Several current Math books on problem solving provide students with techniques to help them compete in Math exams, such as the Mathematical Olympiads (Zeitz, 2006; Andreescu & Gelca, 2008). These books focus almost entirely on solving Math problems, not real world problems.

On the other hand, the recent book entitled *How to Solve It: Modern Heuristics* by Michalewicz and Fogel (2004) leans toward the use of Math to solve real world problems. Books on Statistics and Operations Research are often obligated to deal with real world problems. This is especially true for Applied Statistics, with its attention to the collection and analysis of real world data.

Given the diversity of content and form within Math, it seems reasonable to expect that more than one mathematical framework could be applicable to problem solving. A framework for solving problems is not equivalent to a framework for proving theorems. Also, a framework for solving Math problems might differ from a framework for solving real world problems.

Problem Solving in Software Development

Software Development has rapidly evolved into an extensive discipline that attempts to solve a variety of computation and communication problems. Coursework areas include Programming, Operating Systems, Databases, Networks, Software Engineering, and Electronic Commerce.

The earliest use of computers to perform repetitive calculations could be considered a form of problem solving. The field of Artificial Intelligence has specifically targeted problem solving in software. An early example is the General Problem Solver program for proving theorems, developed by Newell, Shaw, and Simon (1959). Current versions of Microsoft Excel have a Solver *add-in* that can search for solutions to a wide range of numerical problems.

Over a decade ago, IBM developed the Deep Blue computer system to play chess, and reached the Grand Master level. Recently, IBM's Watson computer competed on the TV game show *Jeopardy* and defeated two human champions.

Some areas of computing are more explicit about their desire to solve problems, especially topics which are heavily dependent on Math. Software Development areas such as Programming, Database, and Software Engineering solve problems with less reliance on Math. Most SD students prefer to be exposed to as little Math as possible in their courses.

We do not expect to find a single problem solving framework that is appropriate for all of Software Development. SD areas may share

some features of the Math frameworks, but each SD field contains domain-specific concepts that are difficult to combine into a common framework.

Plan of this Research

In this paper, we examine how problem solving frameworks differ between Math and Software Development. Measurement of mental concepts is always difficult. Our methodology is based on the assumption that the words people use are suggestive of their mental state. In particular, we assume that words used frequently in a book indicate the mental state, or framework, of the author.

Certainly, a framework is more than a list of words. A framework must provide a way to combine the words into a unified "whole". However, we need the individual words to describe the relevant concepts that form the overall framework.

In this study, we compare word frequencies in a sample of Math and SD books that discuss problem solving. After organizing the books into subject matter categories, we list the most frequent words in each category. We then synthesize these results to propose a problem solving framework for each book category. Our findings have relevance in the development of approaches for teaching problem solving in Math and SD courses.

2. METHODOLOGY

The methodology used to examine problem solving frameworks is described in this section. The methodology involved the following steps:

1. Choose a broad sample of Mathematics and Software Development books.
2. Record frequencies for words used often in the books.
3. Convert nouns, verbs, adjectives, and adverbs to a consistent form.
4. Transform the word frequencies to make data from different books comparable.
5. Combine synonyms into *word groups*.
6. Determine the most frequent word groups in each category of books.

Sampling

By design, a wide variety of Math and SD books were sought for our sample. We needed books for which we could determine word usage frequencies. Because we did not have full text files, we selected books from the Amazon web site that included a *concordance* (a list of frequently used words). Our need for a concordance hindered our ability to obtain a random sample of books. However, Amazon does provide a concordance for many of its books, so we were able to get a diverse sample. The majority, but not all, of our sample books are suitable for use as college textbooks.

Books were chosen from three broad categories:

1. *Traditional Math* (TRM) includes books in fields such as Algebra, Analysis, Geometry, Number Theory, and Topology, along with some Probability and Statistics books. Books with the word *Theory* in the title were usually placed in this category. For example, the book entitled "Course in Probability Theory" was classified as Traditional Math.

2. *Applied Math* (APM) includes books with the words *Applied*, *Computational*, *Numerical,* or *Engineering* in the title. For example, the book with the title "Applied Engineering Mathematics" was classified as Applied Math. This category also contains Operations Research and Simulation books, along with Probability and Statistics books that are more applied than theoretical.

3. *Software Development* (SD) includes books on Object-Oriented Programming (OOP), Database (DB), and Software Engineering (SE). These books are used in core SD courses that teach students how to design and implement software systems.

Our complete sample consisted of 53 Traditional Math books, 59 Applied Math books, and 110 Software Development books. The SD sample contained 36 OOP books, 37 DB books, and 37 SE books. The total number of books in the sample was 222.

Data Collection

The Amazon concordance for a book provides a list of the 100 most frequently used words. These concordances screen out many (but not all) common English words, such as "the" and "of". For each concordance word, we recorded

the book code, word, and frequency. Frequency is the actual number of times the word occurs in the book.

Convert Words to a Consistent Form

One problem with using words to infer an author's framework is that words can take more than one form. For example, nouns and verbs may be singular or plural. Verbs can have various tenses. Adjectives and adverbs can have related syntax. To alleviate this problem, we converted many words to a consistent form. We did not want the relative frequency of a word to depend on the particular form an author favored. The following types of word conversions were performed:

1. Convert plural nouns to singular form ("elements" becomes "element").
2. Make verbs refer to plural subjects ("exists" becomes "exist").
3. Change verbs to present tense ("defined" becomes "define", "solving" becomes "solve").
4. Remove endings such as "al" and "ly" from some adjectives and adverbs ("computational" becomes "computation", "finitely" becomes "finite").

Transform Frequencies

Word frequencies were rescaled (or standardized) to allow comparisons between books of different lengths. We rescaled word frequencies within a concordance as follows:

1. We removed all words that are in the list of Top 100 Common English Words (Fry, 1993). Fortunately, Amazon had already removed most of these Top 100 words. Otherwise, we would have had few words left to analyze.
2. For the remaining (approx. 90) words, we calculated the average word frequency for the concordance.
3. We then restated each individual word frequency (Freq) relative to the average frequency (avgFreq) using the formula:

   StdFreq = (Freq / avgFreq) * 100

   With this calculation, a standard frequency (StdFreq) score of 100 represents the transformed frequency for the "average word" in the reduced concordance. A word with a StdFreq value of 300 would appear three times as often as the average concordance word in the same book.

Combine Synonyms into Word Groups

A special complication with assembling words into frameworks is that different words can have similar meanings. When relevant, we combined two or more synonyms into a concatenated *word group*. For example, *algorithm* and *method* became *algorithm/method*. We applied this step *after* standardizing the word frequencies (StdFreq) because we wanted the average frequency for a concordance to be based on individual words. When synonyms are combined into word groups, the StdFreq score for the group is the sum of the StdFreq scores of the words in the group.

3. PROBLEM SOLVING BOOKS

The primary approach in this study of problem solving frameworks in Mathematics and Software Development was to examine frequently used words in our sample of books. But which books in the sample discuss problem solving? Polya's "How to Solve It" is certainly a candidate. However, only four books in our sample contain the words *problem* and/or *solve* in the title. Instead, we chose to focus on books that include *problem* or *solution/solve* in their concordances. We assumed that these books would be more likely to involve problem solving, even though these words are often used in other contexts.

Table 1: Math and SD Books by Category

Category	All Books	*Problem* Books	*Problem+ Solution* Books
APM	59	53	48
TRM	53	26	13
SD (OOP) (DB) (SE)	110 (36) (37) (37)	60 (12) (17) (31)	8 (2) (0) (6)
Total	222	139	69

Starting with a sample of 222 Math and SD books, the number of concordances containing the word *problem* is 139. This initial constraint removes one-third of our sample. If we then eliminate books that do not include *solution* or *solve* in their concordances, the remaining

sample has only 69 books--about 1/3 of our original sample.

The main difficulty with limiting our analysis to these 69 "problem + solution" books is that the reduction does not apply equally to all book categories. Table 1 summarizes how the book counts are reduced in each category as we successively apply the *problem* and *solution* filters.

Requiring Applied Math concordances to contain the *solution* keyword in addition to *problem* is not an issue. The resulting sample has 48 of the 53 *problem* books. The drop is more precipitous for Traditional Math (from 26 to 13 books) and Software Development (from 60 to 8 books). Note that none of the Database books and only 2 of the Programming books include both keywords in their concordances.

Many authors of Traditional Math are more concerned with *proofs* than with problem solving. This can partially explain the reduced number of books in this category that contain *solution* or *solve*, but it doesn't explain the remarkably small number of Software Development books that mention *solutions*.

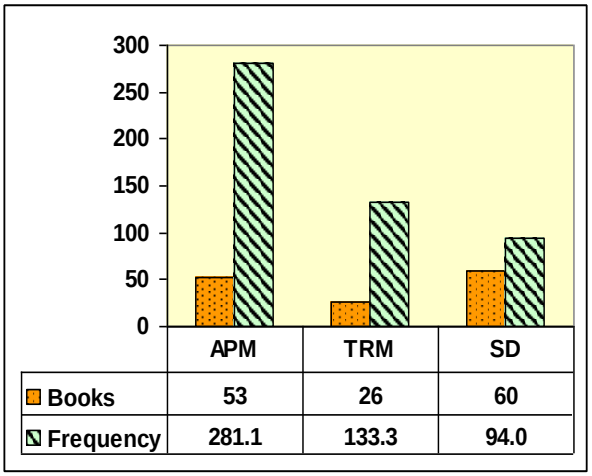
Problem solving is an important part of Software Development, as stated by McConnell (2004):

> Problem solving is the core activity in building computer software.

Programming, Database, and Software Engineering books use an alternative terminology for problem solving. From a Software Development perspective, *requirements* define the problem, and *software* is the solution. Programs and databases are essential components of the solution. The goal of Software Engineering is to effectively build software systems that meet customer requirements.

Because of the extreme sample size reduction that would result from requiring both *problem* and *solution* to be concordance words, we decided to impose the less restrictive constraint that only *problem* must be in the concordance. Figure 1 shows the resulting sample of 139 books used in the analysis that follows.

Across the three book categories, the average standard frequency (avgStdFreq) for the word *problem* varies widely. For the 53 Applied Math concordances that contain *problem*, the avgStdFreq of 281.1 indicates that this word occurs almost three times as often as an average concordance word. At the other extreme, in 60 Software Development books, *problem* occurs less often (94.0) than an average concordance word.



	APM	TRM	SD
Books	53	26	60
Frequency	281.1	133.3	94.0

Figure 1: "Problem" frequency by Category.

4. PROBLEM SOLVING FRAMEWORKS

This section describes how we obtained the words that form the problem solving frameworks for each category. We looked for words that are used *frequently* within each book and *consistently* across books in the same category. Given a category (e.g. Applied Math) and a word in at least one of the concordances, we calculated the number of books containing that word, plus the avgStdFreq for the word. We retained the words that appear in most of the category books and had a high avgStdFreq.

Our principal methodology decision was the choice of cutoff points for number of books and avgStdFreq. After some trial and error, we set the minimum number of books at 70% of the sample size. For the 53 APM books, 70% is 37 books (rounded). For avgStdFreq, we chose a cutoff point of approximately 150, with some judgment reserved for words near this point.

To qualify as a framework word, we wanted most of the books in the category to agree on the importance of the word. Some words had a high frequency, but appeared in only a few of the books. For example, the word *simulation* appears in 8 Applied Math books, with an

avgStdFreq of 219.7. This word is important in those 8 books, but is not used regularly throughout the Applied Math category.

Other words appear in most books, but with low frequencies. For example, the word *result* appears in 44 APM concordances, but the avgStdFreq value is a below-average 88.1. Of passing interest, this word could be considered a synonym for *solution*.

Applied Math Framework

Using the methodology explained in the preceding paragraphs, we generated a list of the 10 most frequent words and word groups for Applied Math books. This list is presented in Table 2. We include the word group *set/element*, even though its frequency value is slightly below 150.

Table 2: Most frequent words in Applied Math books (N=53).

Word Group	Books	Avg StdFreq
problem	53	281.1
algorithm/method	47	280.5
function	50	248.1
solution/solve	48	239.9
value/variable	52	250.9
model/modeling	38	227.7
equation/inequality	43	220.0
system	47	173.2
point/line	52	161.8
set/element	48	146.7

The most frequent word is *problem*. It is not surprising that this word is in all of the APM books, since this condition was used to generate the sample. What is unusual is that the frequency of *problem* and *solution/solve* is relatively high in this category. This suggests that the framework for Applied Math does emphasize problem solving.

The word groups *model/modeling* and *algorithm/method* describe this category's approach to solving real world problems. Models are used to abstract relevant aspects of the real world problem. Algorithms describe the computational effort needed to obtain a solution.

The Applied Math framework includes several widely-used mathematical objects--*function*, *variable*, *equation*, *point*, *line*, and *set*. These words appear frequently in most of the Applied Math books. However, other familiar Math concepts, such as *matrix*, *polynomial*, and *vector* do not appear in this general framework. These domain-specific words are in the concordances of some Applied Math books, but absent from many others.

Traditional Math Framework

Repeating the same methodology used for Applied Math, we obtained a list of the 12 most frequent words and word groups for Traditional Math books. The list is shown in Table 3. We include the word group *definition/define* in this list, since its avgStdFreq value is almost 150.

Table 3: Most frequent words in Traditional Math books (N=26).

Word Group	Books	Avg StdFreq
point/line	19	411.6
theorem/lemma/corollary	25	326.8
function	25	278.5
proof/prove	23	258.7
let	26	228.9
set/element	26	225.1
value/variable	20	197.3
show/shown	24	181.9
hence/thus/therefore	26	172.1
follow/following	24	163.3
equation/inequality	21	163.1
definition/define	23	149.3

The most frequent word group is *point/line*, which appears in 19 (73%) of the Traditional Math books. Points and lines--along with functions, sets, variables, and equations--are Math objects that are also in the Applied Math framework, but with different frequencies.

Two high-frequency word groups are *theorem/ lemma/corollary* and *proof/prove*. This reveals that the primary goal of Traditional Math is proving theorems. The words *model* and *algorithm* are not part of this framework.

Instead, this framework prefers the use of logic to solve Math problems.

Not everyone agrees that theorem proving is equivalent to solving problems. Concerning the "problem of proving things", Michalewicz and Fogel (2004) state:

> ... if you ask someone to *find* some solution to a problem, they'll typically find this much easier than if you had asked them to *prove* something about the solution, even when the two tasks are exactly the same mathematically.

For example, which of the following statements require problem solving?

1. Find the largest prime number less than 100.
2. Prove that 97 is the largest prime number less than 100.

Which task is more difficult? Are the tasks equivalent mathematically?

We could take the view that, in Traditional Math, the *problem* is to verify or refute a theorem. The proof or counterexample is the *solution*.

The remaining words in the Traditional Math framework--such as let, show, hence, follow, and define--are common terminology used in stating theorems and expressing proofs.

Software Development Framework

The Software Development category includes books on Object-Oriented Programming, Database, and Software Engineering. Table 4 lists 9 of the most frequent word groups for this category.

The top four word groups are *object/class*, *system*, d*ata*, and *program/code*. The framework formed by these words is very different from the frameworks for the two preceding categories of books. The word groups *problem* and *solution/solve* do not appear on this list. However, *model/modeling*, *algorithm/method*, and *system* are shared with Applied Math.

This is the only framework that includes the real world concept *data*. Note that no Math objects are on this list.

Table 4: Most frequent words in Software Development books (N=60).

Word Group	Books	Avg StdFreq
object/class	50	412.6
system	54	293.1
data	57	243.5
program/code	54	219.2
process/processing	48	211.9
user/client/customer	48	200.1
model/modeling	49	190.9
algorithm/method	44	182.7
design	46	151.1

Several common Software Development concepts that almost made the list include *software*, *requirement*, and *development*. These words have high frequencies, but are not in enough books (< 42) to qualify for this framework.

The Software Development framework uses models and algorithms to design software systems that integrate programs, data, and users.

5. COMPARING FRAMEWORKS

In the previous section, we presented problem solving frameworks for three book categories in Mathematics and SD. The frameworks are described by of lists of words used frequently in Applied Math, Traditional Math, and Software Development books. The frameworks are not independent, since some words appear on more than one list.

Mathematical Frameworks

The Applied Math and Traditional Math frameworks share 5 word groups that represent widely used mathematical objects--*set*, *function*, *variable*, *equation*, and *point/line*. The remaining words indicate the different nature of the two frameworks. The Applied Math list includes the words *problem*, *solution*, *model*, *algorithm*, and *system*, which describe an approach for solving problems in real world systems. The Traditional Math list includes the words *definition*, *theorem*, and *proof*, along with several common terms used in presenting

theorems and proofs. The emphasis in this framework is on solving mathematical problems through the use of logic.

Software Development and Applied Math

The Software Development framework presents a different approach to problem solving. This list includes *model*, *algorithm*, and *system* from Applied Math, and adds terms that are used in the software development process. In particular, *design*, *class*, *program* and *data* are highlighted. In this framework, users supply the problem, and the completed software product represents the solution. Thus, Software Development combines important Applied Math methods with specific components of the final system.

The relationships between the three frameworks are summarized visually in the Appendix. This figure is a Venn diagram that displays the frameworks as overlapping sets of word groups. No word group appears in all three sets. Moreover, Traditional Math and Software Development have no words in common.

6. DOMAIN-SPECIFIC FRAMEWORKS

We have described the commonalities and differences in the problem solving frameworks for the three book categories. Within each category, several subfields, or domains, are represented. Each of the main frameworks are based on concepts that apply to most of the books in the category. Domain-specific concepts are masked at this level of analysis.

The number of books in each area of Applied Math and Traditional Math is relatively small, so the ability to make domain-specific comparisons is limited. We do highlight the Operations Research framework within Applied Math.

The Software Development book sample covers three domains--Programming, Database, and Software Engineering. Word lists for each of these domains are presented below.

Operations Research Domain

The Applied Math (APM) sample includes 7 books on Operations Research (OR). The top word groups for the OR books, including domain-specific (New) words, are listed in Table 5.

The OR books present a classic variation of the Applied Math framework. OR includes six essential Applied Math word groups--*problem*, *solution/solve*, *model/modeling*, *algorithm/method, value/variable*, and *system*. All but *algorithm/method* have higher average frequencies in the OR domain than in the larger sample of Applied Math books.

The OR framework shares *program/code* with Software Development. It also adds *condition/constraint* and *cost*, which are important in optimization problems.

Table 5: Most frequent words in Operations Research books (N=7).

Word Group	Books	Avg StdFreq	vs. APM
problem	7	443.1	281.1
model/modeling	7	376.5	227.7
value/variable	7	357.9	250.9
solution/solve	7	324.3	239.9
system	6	272.1	173.2
algorithm/method	7	223.7	280.5
program/code	6	198.4	SD
condition/constraint	7	191.1	New
cost	7	178.4	New

Programming Domain

The Software Development sample includes 12 books on Programming (OOP). The top word groups for the OOP books, including one domain-specific word, are presented in Table 6.

Table 6: Most frequent words in Programming books (N=12).

Word Group	Books	Avg StdFreq	vs. SD
object/class	12	674.9	412.6
program/code	12	340.7	219.2
algorithm/method	11	330.9	182.7
value/variable	11	192.4	Math
type	10	186.1	New
set/element	11	170.3	Math
function	9	164.0	Math

The top three Programming word groups--*object/class*, *program/code*, and *algorithm/ method*--are shared with Software Development. These word groups have much higher frequencies in this domain than for the general SD framework.

The single new Programming word is (data) *type*. Three other word groups are borrowed from the Math frameworks. The word *model* is not included in this list because it appeared in the concordances of only 4 OOP books.

Database Domain

The Software Development sample includes 17 Database (DB) books. The top word groups for the DB books are shown in Table 7.

Table 7: Most frequent words in Database books (N=17).

Word Group	Books	Avg StdFreq	vs. SD
data	17	473.7	243.5
object/class	14	445.0	412.6
relation/table	17	360.6	New
database	17	334.9	New
system	17	213.0	293.1
user/client/customer	14	208.4	200.1
query	14	201.8	New
model/modeling	17	201.7	190.9
attribute/column	16	177.0	New
set/element	13	161.2	Math

The Database framework includes four new concepts--*database*, *relation/table*, *query*, and *attribute/column*. Not surprisingly, these words indicate an emphasis on relational databases. The word *data* has a much higher frequency in the DB domain than in the Software Development framework. Also, exactly one Math word group (*set/element*) is on this list.

Software Engineering Domain

The Software Development sample includes 31 Software Engineering (SE) books. The top word groups for the SE books are shown in Table 8.

New SE domain-specific words include *software*, *project*, *requirement*, *development*, and *product*. For most of the word groups shared with the Software Development framework, the average frequency for the SE books is close to the value for the larger SD sample. The likely reason for this similarity is that SE books comprise over half the sample of SD books.

Two exceptions are *object/class* and *system*. In the Programming and Database books, the frequency of *object/class* is substantially higher than in the SE books. This pattern is reversed for *system*.

Table 8: Most frequent words in Software Engineering books (N=31).

Word Group	Books	Avg StdFreq	vs. SD
software	31	402.4	New
system	30	363.8	293.1
process/processing	31	277.8	211.9
object/class	24	262.5	412.6
project	27	243.0	New
requirement	28	242.6	New
program/code	28	219.4	219.2
user/client/customer	27	218.9	200.1
development	31	208.6	New
model/modeling	28	189.1	190.9
product	23	174.0	New
design	30	168.8	151.1
data	28	157.4	243.5

7. SUMMARY AND CONCLUSIONS

The general objective of this study was to examine how problem solving frameworks differ between Mathematics and Software Development. Our approach assumes that words used frequently in a book indicate the mental framework of the author.

We started with a sample of 222 books drawn from three categories: Traditional Math, Applied Math, and Software Development. We chose books that had an Amazon concordance that lists the 100 most frequently used words. Because this research involved problem solving, we eliminated books that did not include *problem* in their concordances, leaving us with 139 books for further study.

We modified the concordance words to compensate for syntactic differences in nouns, verbs, adjectives, and adverbs. We also standardized the word frequencies in each book to make books of various lengths comparable. Finally, we collected words having similar meanings into word groups. Then we generated a list of the most frequent words and word groups in each book category. Based on these lists, we described problem solving frameworks for the categories.

Our results indicate that the frameworks for Traditional Math and Applied Math are fundamentally different. Applied Math uses *models* and *algorithms* to solve real world problems. Traditional Math is more concerned with *theorems* and *proofs*, with the application of logic to solve Math problems.

The Software Development framework shares an emphasis on *models* and *algorithms* with Applied Math, but includes many domain-specific features. Problem solving in Software Development is aimed at creating a successful software product. The methodology involves the design of models and algorithms for *programs* and *data*. Often these models and algorithms are represented visually rather than mathematically, before being implemented in software.

Our findings suggest ways to teach problem solving in Traditional Math, Applied Math, and Software Development courses. In Traditional Math courses, the instructor should introduce an appropriate amount of rigor in theorem proving, consistent with the level of the course. Math majors eventually acquire the mental fortitude to appreciate well-crafted theorems and proofs.

For Applied Math (e.g. Engineering) courses, students prefer to solve real world problems ("story problems") using abstract models and computational algorithms. When presented, proofs can be more informal and descriptive.

For Software Development courses, problems are expressed in terms of models and algorithms that can be used to create software solutions. Here, the nature of the problem and the solution depend on the application. Programming courses involve models for software architecture, as well as ways to specify algorithms. Database courses spend more time on data models, along with query algorithms written in non-procedural SQL. The framework for Software Engineering courses must include the entire life cycle of programming, database, and management activities that lead to the final system.

Computational thinking enthusiasts (Wing, 2006) seem to promote algorithms and computation at the expense of modeling. Conversely, in a recent article on abstract thinking, Kramer (2007) gives greater emphasis to modeling and abstraction:

> Modeling is the most important engineering technique; models help us to understand and analyze large and complex problems.

Teachers of Software Development courses should provide students with substantial exposure to both models and algorithms during the journey from user problems to the eventual software destination.

8. REFERENCES

Andreescu, T., and Gelca (2008), R. Mathematical Olympiad Challenges (2nd ed). Birkhäuser Boston.

Bratko, I. (2001). Prolog Programming for Artificial Intelligence (3rd ed). Addison-Wesley.

Fry, E., et al (1993). The Reading Teacher's Book of Lists (3rd ed). Center for Applied Research in Education.

Kramer, J. (2007). Is abstraction the key to computing? *CACM*, Vol 50, No 4.

McConnell, S. (2004). Code Complete (2nd ed). Microsoft Press.

Michalewicz, Z., and Fogel, D. (2004). How to Solve It: Modern Heuristics (2nd ed). Springer.

Newell, A.; Shaw, J.C.; Simon, H.A. (1959). Report on a general problem-solving program. *Proceedings of the International Conference on Information Processing*.

Polya, G. (1945). How To Solve It. Princeton University Press.

Velleman, D. (1994). How to Prove It: A Structured Approach. Cambridge University Press.

Wing, J. (2006). Computational thinking. *CACM*, Volume 49, No. 3.

Zeitz, P. (2006). The Art and Craft of Problem Solving (2nd ed).

9. APPENDIX

Mathematics and Software Development Frameworks