Integrating Mathematical Thinking, Abstract Thinking, and Computational Thinking

Kirby McMaster
St. Mary's College of MD, kmcmaster@smcm.edu

Brian Rague
Weber State University, brague@weber.edu

Nicole Anderson
Winona State University, nanderson@winona.edu

Abstract – **In recent years, several groups of Computer Science educators have made a sustained effort to capture the essence of CS apart from programming. Three of these approaches are Mathematical Thinking, Abstract Thinking, and Computational Thinking. Each approach tries to clarify areas of CS that are not directly tied to writing computer programs. In a separate line of research, the current authors have been examining different ways to teach mathematics to CS students. We developed a Computational Math scale that measures the level of problem-solving *gestalt* exhibited by textbook authors. In this study, we relate our Computational Math framework to current research on Mathematical, Abstract, and Computational (MAC) Thinking. We counted words used frequently in research articles and compared them to words that form the Computational Math scale. Our results suggest an overall MAC Thinking framework that integrates a wide range of topics relevant to computing and programming.**

Index Terms – Abstraction, Algorithm, Computational Math, Model, Thinking.

INTRODUCTION

In its early years, the field of Computer Science (CS) was not afraid to declare an affinity for computer programming. As evidence of a life-long love of programming, Knuth wrote (and is still writing) a well-known series of books called *The Art of Computer Programming*. From Knuth's [1] viewpoint, "the notion of an algorithm or a computer program provides us with an extremely useful test for the depth of our knowledge about any given subject." In a recent interview, Knuth [2] made a similar statement: "The truth is you don't really understand something until you've taught it to a computer, until you've been able to program it."

Along an alternate timeline, many authors have argued that CS should place more emphasis on theoretical and design issues and rely less on programming. In spite of these efforts, the myth persists that "Computer science equals programming" [3].

In recent years, renewed efforts have been made to capture the essence of CS apart from programming. Several of these attempts replace the historical terms "Algorithmic Thinking" and "Computational Science" with alternative concepts such as "Mathematical Thinking" [4], "Abstract Thinking" [5], and "Computational Thinking" [6]. Each of these overlapping approaches tries to clarify areas of CS that are not directly tied to writing computer programs.

There are several reasons for this current interest in "re-centering" CS away from programming [7]:

1. To reverse the declining number of students majoring in CS by placing less emphasis on teaching of "industrial languages" and more emphasis on computer theory and software engineering. In an extreme case, Computer Science Unplugged [8] claims that "computer science isn't really about computers at all!"
2. To provide special introductory CS courses for non-majors that will stimulate their interest in computation. Examples include "Pander to Ponder" [9] and "CS for Non-Majors Using Principles of Computation" [10].
3. To indicate ways in which CS can contribute to other disciplines. Examples include "Renaissance Computing" [11] and "A Multidisciplinary Approach Towards Computational Thinking for Science Majors" [12].

In a separate line of research, the current authors have been examining different ways to teach mathematics to CS students. Many students in CS (and most other fields) suffer from "math anxiety" and have difficulty learning math concepts. Part of this difficulty may be due to the mental framework, or *gestalt*, in which math is commonly presented to students, with substantial emphasis on theorems and proofs.

In our previous research [13], we were able to characterize two frameworks for mathematics, one based on *proving theorems* and the other based on *solving*

978-1-4244-6262-9/10/$26.00 ©2010 IEEE

problems. Our methodology made the assumption that words used frequently in a book indicate the gestalt of the author. By examining word frequencies in various traditional and applied mathematics books, we developed two scales for measuring the framework preferred by an author. A Logical Math scale measures theorem-proving gestalt, and a Computational Math scale measures problem-solving gestalt. Only the Computational Math scale is utilized in this study. Our choice of the word "computational" in this context was influenced by the Program in Applied and Computational Math (PACM) at Princeton University, and not by recent research on Computational Thinking.

The purpose of this study was to compare three frameworks—Mathematical Thinking, Abstract Thinking, and Computational Thinking—with our Computational Math gestalt. We examined words used frequently in current articles on the three "Thinking" frameworks and obtained counts for words listed on the Computational Math scale. We also summarized other Computer Science words that appeared in the articles in order to detect differences between the approaches. Our results lead to an overall "Thinking" framework, which integrates a variety of CS topics relevant to computing and programming. In this paper, we refer to this combined framework as *MAC Thinking* (Mathematical + Abstract + Computational).

COMPUTATIONAL MATH

The methodology for creating our Computational Math gestalt scale is detailed in a previous paper [13]. A brief outline of this methodology is given below to provide a background for later sections of this paper.

From the Amazon web site, we selected a broad sample of 56 Applied Math books, each having a *concordance*. An Amazon concordance provides a list of the 100 most frequently used words, with many common English words excluded. We converted some words so that the scale contribution of a word would not depend on the particular noun form or verb tense an author favored. We also combined two or more similar words (nouns and verbs, synonyms) into a *word-group* (e.g. "solution/solve", "algorithm/method").

Constructing the Computational Math scale (referred to as CMATH) was an iterative process. We searched for words that are used *frequently* within each book, and *consistently* across these books. We generated a tentative CMATH scale, and then calculated CMATH scores for each book. We removed the lowest scoring books, and repeated the process. After several iterations, we obtained a CMATH scale constructed from a reduced sample of 25 Applied Math books.

The CMATH scale consists of 9 word-groups and weights. The weights are based on word frequencies, and the sum of the weights over all word-groups in the scale is 100. The details of this scale are presented in Table 1.

TABLE I
COMPUTATIONAL MATH SCALE (CMATH)

WORD-GROUP	WEIGHT
problem	19.28
algorithm/method	16.40
solution/solve	14.29
value/variable	11.14
equation/inequality	11.02
function/mapping	10.90
model/modeling	8.24
system/subsystem	4.48
condition/constraint	4.25
Total Weight	100.00

The most frequent word (highest weight) for the CMATH scale is "problem"; the third most frequent word-group is "solution/solve". Thus, problem solving is a central theme in the Computational Math framework.

Computational Math, as described by the CMATH scale, is concerned with how to use mathematics to solve real world problems. The word-groups "model/modeling" and "algorithm/ method" describe the main approach to solving problems. Words like "variable", "equation", "function", and "constraint" are components of mathematical models and algorithms. Figure 1 provides a visualization of the primary concepts and associated word-groups for the Computational Math framework in terms of three "worlds"—Real World, Math World, and Computer World.

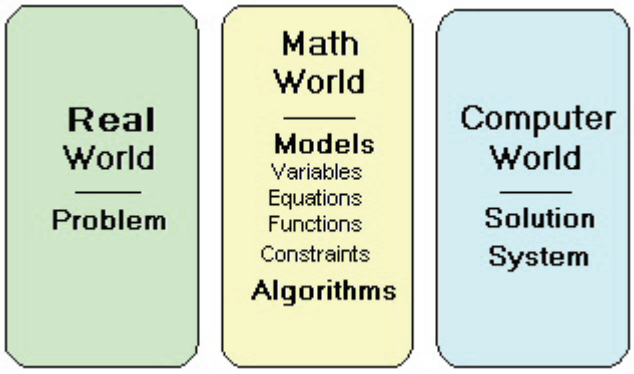

FIGURE 1
COMPUTATIONAL MATH FRAMEWORK

The word "system" placed in Computer World refers to the implemented computer system that provides the solution. However, "system" also applies to Math World when the models and algorithms are developed as an abstract mathematical system.

MAC THINKING AND CMATH WORDS

This section of the paper summarizes word frequencies in a sample of 15 articles on Mathematical Thinking (MT), Abstract Thinking (AT), and Computational Thinking (CT). The articles were drawn from publications such as CACM, SIGCSE Bulletin, and several conference proceedings. Title keywords such as "mathematical", "abstract", "computation", and "thinking" were used in selecting the papers. Table 2 presents a brief summary of the sample.

TABLE II
SUMMARY OF SAMPLE ARTICLES

	MT	AT	CT
Articles	4	5	6
Total Words	14275	16466	15818
Most Freq: "the"	647	995	729
Publication Dates	2001-03	2006-08	2006-09
References	[4,14, 15,16]	[1,5,18, 19,20]	[6,10,12, 21,22,23]

The articles were downloaded from the Internet, mostly in PDF format. The text in the articles, excluding references, was transferred to MS Word documents. The freeware program TextSTAT (courtesy of Matthias Hüning) was used to obtain word counts and then export the frequency distributions to MS Excel. Words and their frequencies were combined into word-groups, consisting of similar noun forms, verb tenses, adjectives, and synonyms (e.g. "abstract/abstraction"), repeating the approach used to construct the CMATH scale.

We first examine the concentration of CMATH scale words in the sample articles. From our view, the five most important words (and associated word-groups) on the CMATH scale are "problem", "solution", "model", "algorithm", and "system". Table 3 gives the frequency for each of these word-groups in the 15 articles. Results are presented separately for MT, AT, and CT articles. Because the number of words is slightly different for the three sets of articles, we restate (in parentheses) each frequency as a rate per 1000 words. For example, in the MT articles, "problem" appears 89 times out of 14275 total words. The MT occurrence rate for "problem" is 6.2 times per 1000 words. By comparison, the most frequent MT word is "the", which appears 647 times, for a rate of 45.3 per 1000 words.

In Table 3, the largest frequency (and rate) for a single CMATH word-group within a set of articles is 157 (rate = 9.5), which occurs for "solution/solve" in the AT articles. For the five main word-groups, we display the largest frequency and rate in **bold**. The word-groups "problem", "algorithm/method", and "system/subsystem" occur most often in the MT articles. Word-groups "solution/solve" and "model/modeling" appear most often in the AT articles. No CMATH word-group has its highest frequency in the CT articles.

TABLE III
COMPUTATIONAL MATH (CMATH) WORDS

WORD-GROUP	MT	AT	CT
problem	**89 (6.2)**	56 (3.4)	52 (3.3)
solution/solve	61 (4.3)	**157 (9.5)**	37 (2.3)
model/modeling	42 (2.9)	**51 (3.1)**	12 (0.8)
algorithm/method	**80 (5.6)**	21 (1.3)	49 (3.1)
system/subsystem	**40 (2.8)**	9 (0.5)	29 (1.8)
other CMath words	49 (3.4)	27 (1.6)	47 (3.0)
Total CMath words	361 (25.3)	321 (19.5)	226 (14.3)
Total Words	14275	16466	15818

Word-groups having a fairly low frequency and rate are "model/modeling" for CT articles and "algorithm/method" and "system/subsystem" for AT articles. This suggests that CT places less emphasis on modeling, while AT is less concerned with algorithms and systems. Overall, the MT articles make the most use of CMATH words (rate = 25.3), which is not unexpected, given that the CMATH scale was developed from Applied Math books. On the other hand, the CT articles make the least use of CMATH words (rate = 14.3). In the next section, we will discover words that are employed more often in CT articles.

MAC THINKING AND CS WORDS

The three Thinking approaches are not limited to mathematical concepts. They also include a large number of Computer Science and other words. In this section, we summarize the frequency of various CS words that appear in the MT, AT, and CT articles. We have divided these words into two lists. The first list includes words that describe the type of "thinking" proposed by each article. The second list looks at common Software Development words that appear in the articles.

I. Type of Thinking Words

Table 4 presents frequencies and rates for word-groups that are directly related to the type of thinking endorsed in each article category. In this table, we have marked in **bold** any word-group frequency and rate that is distinctly larger than the other frequencies on the same row. This indicates when a particular word-group predominates for one category of articles.

For example, the word-group "abstraction/abstract" has a frequency of 401 for AT articles, which is more than 13 times larger than the frequencies for MT and CT articles.

Similarly, the frequency of "mathematics/mathematical" for MT is 346, which is over 10 times the frequencies for AT and CT. A third case is the frequency of 199 for the word-group "computation/ computational" in CT articles, which exceeds the other row frequencies by a factor of 12. It is not surprising that word-groups having the three largest modal frequencies and rates match the adjectives commonly used to describe the corresponding frameworks.

TABLE IV
MAC THINKING WORDS

WORD-GROUP	MT	AT	CT
abstraction/abstract	30 (2.1)	**401 (24.4)**	17 (1.1)
computation/ computational	4 (0.3)	16 (1.0)	**199 (12.6)**
computing/ compute	18 (1.3)	27 (1.6)	101 (6.4)
computer	127 (8.9)	99 (6.0)	154 (9.7)
mathematics/ mathematical	**346 (24.2)**	22 (1.3)	33 (2.1)
science/scientific	119 (8.3)	100 (6.1)	190 (12.0)
thinking/think	82 (5.7)	116 (7.0)	133 (8.4)
Total MAC Thinking words	726 (50.9)	781 (47.4)	827 (52.3)
Total Words	14275	16466	15818

The word-groups "computing/compute" and "computer" are very similar. We kept these word-groups distinct, because the usage patterns differ across the articles. "Computing/compute" has a much higher frequency and rate among CT articles (as does "computation"). "Computer", on the other hand, appears frequently in all types of articles, although somewhat more often for CT. This suggests that computers are a foundation concept in all three Thinking frameworks.

For the "science/scientific" word-group, each set of articles has a frequency of at least 100, indicating that all types of thinking apply to scientific applications. The largest frequency on this row appears for CT, which may be partly due to their use of the historical term "computational science". The final word-group "think/thinking" is used most often in CT articles, and least often by MT. This does not imply that thinking is less important in mathematics. In fact, if the word "reasoning" is added to the "thinking/think" word-group, the usage rate is almost identical for MT, AT, and CT.

The total usage of Thinking words is fairly even for the three sets of articles, with rates per 1000 words ranging from 47.4 for AT to 52.3 for CT. Beyond the three "modal" word-groups, the remaining word-groups moderately favor CT.

II. Software Development Words

The next list is shown in Table 5, which summarizes word-groups that include basic Software Development terms. We have marked a frequency in **bold** if it is at least twice the next highest frequency on the same row. Note that five of the eight word-groups have a bold entry.

TABLE V
SOFTWARE DEVELOPMENT WORDS

WORD-GROUP	MT	AT	CT
analysis	23 (1.6)	22 (1.3)	16 (1.0)
data/information	17 (1.2)	21 (1.3)	**56 (3.5)**
design	14 (1.0)	**67 (4.1)**	27 (1.7)
develop/ development	66 (4.6)	112 (6.8)	45 (2.8)
engineer/ engineering	**107 (7.5)**	24 (1.5)	12 (0.8)
program/ programming/code	70 (4.9)	62 (3.8)	**167 (10.6)**
requirement/ specification	69 (4.8)	43 (2.6)	38 (2.4)
software	**124 (8.7)**	37 (2.2)	10 (0.6)
Total SD words	490 (34.3)	388 (23.6)	371 (23.5)
Total Words	14275	16466	15818

The three largest bold frequencies are 167 for "program/ programming/code" (CT), 124 for "software" (MT), and 107 for "engineer/engineering" (MT). The other two bold frequencies are 67 for "design" (AT) and 56 for "data/information" (CT). For the remaining word-groups, "develop/development" was used most by AT, "requirement/specification" appeared most often in MT articles, and "analysis" was used sparingly in all articles. Overall, the MT articles contained the most Software Development words (490; rate = 34.3).

MAC THINKING FRAMEWORK

We now expand our Computational Math gestalt into a broader framework for MAC Thinking. In Figure 2, we start with the primary concepts of "problem", "solution", "model", "algorithm", and "system" from Computational Math. We again allocate these concepts to three worlds.

Real World activities focus on defining the problem and determining requirements for the proposed solution.

We renamed Math World as Abstract World (or Design World) to reflect the important role of abstraction at this stage. We also mention that in CS we do not always express models and algorithms mathematically. Instead, we often prefer visual representations using diagrams and prototypes. In Abstract World, we develop models for programs (software architecture), for data (from

preliminary models to normalized relational models), and for systems (such as state diagrams and network communications). Algorithms describe the computations that are to be performed in moving data through the system.

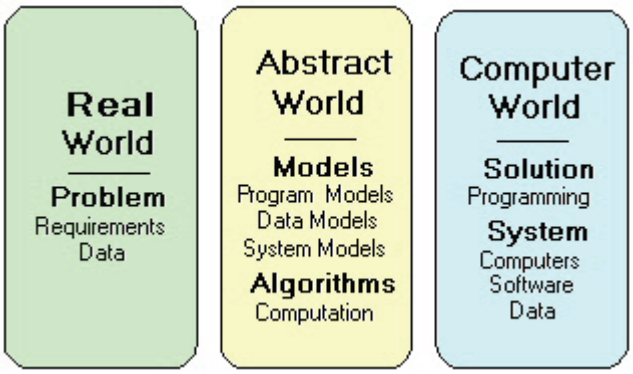

FIGURE 2
MAC THINKING FRAMEWORK

In Computer World, the actual system that provides the solution is constructed. This system includes computers and other hardware, software, and managed data. Programming is one of the main activities (but not the only activity) in this World.

The primary focus in MT, AT, and CT is on Real World and Abstract World, which emphasize the broad range of non-programming activities in CS. But CS cannot ignore Computer World. Programming and other aspects of system implementation are neither trivial nor routine. When we "teach our computers", as Knuth might say, they do not always obey us in the manner we expect. One of the main advantages in CS is that we can observe how our solutions behave when our code executes (and make changes when necessary).

SUMMARY AND CONCLUSIONS

In this study, we compared three frameworks—Mathematical Thinking (MT), Abstract Thinking (AT), and Computational Thinking (CT)—with Computational Math gestalt. We examined words used frequently in current articles on the three Thinking frameworks and obtained counts for words listed on the Computational Math scale (CMATH). Overall, the MT articles included the most CMATH words, which is consistent with the fact that the scale was developed from Applied Math books. Two of the CMATH words most relevant to CS are "model" and "algorithm". Contrasting CT and AT, CT articles rarely used the word "model", while AT articles used the word "algorithm" infrequently. Perhaps CT and AT adherents should combine their efforts on these concepts.

We also counted other Computer Science words in the articles to detect differences between the MT, AT, and CT frameworks. The three "modal" words "mathematical", "abstract", and "computational" matched up well with the three frameworks. The words "computer", "science", and "thinking" were used frequently in all three sets of articles, but most often in CT articles. Software Development words that appeared relatively often include "software" and "engineering" in MT articles, "development" in AT articles, and "programming" in CT articles.

Our results were combined into an overall *MAC Thinking* framework, which integrates a variety of CS topics relevant to computing and programming. This framework allocates the concepts into three worlds—Real World, Abstract World, and Computer World. MT, AT, and CT articles emphasize the first two worlds, which consist of activities that precede most of the coding for computer programs.

In a recent discussion about CT, Denning [24] stated: "Computational thinking is one of the key practices of computer science. But it is not unique to computing and is not adequate to portray the whole of the field." The MAC Thinking framework is broader than CT. Instead of deemphasizing programming, this integrated framework clarifies its important role in CS. Even so, this expanded framework does not encompass every important area and issue in CS. Today's use of computers involves communication of information more that computation. Nevertheless, underneath all popular computer applications lie the worlds of problems, models and algorithms, and software.

FUTURE RESEARCH

Our main reason for examining and building frameworks is to improve CS education. The MT, AT, and CT papers share this focus by including several words related to education, such as "school", "student", "teacher", "course", "study", and "learn". CT spends more words talking about the educational implications of their approach (rate = 36.4) than do MT (26.0) and AT (26.1).

Having an overall MAC Thinking framework for CS does not preclude developing more specific frameworks for various CS topic areas and courses. We have been defining and applying frameworks for Programming, Database Systems, and Software Engineering. We would like to create frameworks for other areas such as Computer Architecture, Operating Systems, Data Communications, and Computer Theory.

All of the frameworks we have developed so far have been constructed from words used frequently by authors of textbooks and research articles, which strongly influence the subject matter taught in the classroom. We are now measuring students' perceptions of framework concepts to determine what is important to them. We have not begun the more difficult task of measuring how effective these frameworks are in improving learning.

We gain an appreciation for the content of a puzzle by fitting together the pieces. Frameworks should enable

students (and teachers) to combine "pieces of CS" into patterns that can be viewed as a whole. Hopefully, this will promote deep learning and understanding. According to Bain [25], if we do not provide meaningful frameworks to students, they will attempt to form their own (with uncertain results).

REFERENCES

[1] Knuth, Donald 1974. Computer programming as an art. CACM, Vol 17, No 12.

[2] Shustek, Len (ed.) 2008. Donald Knuth: A life's work interrupted, part 2. CACM, Vol 51, No 8.

[3] Denning, Peter 2004. The field of programmers myth. CACM, Vol 47, No 7.

[4] Henderson, Peter, et al. 2001. Striving for mathematical thinking. ITiCSE 2000 Working Group Report, SIGCSE Bulletin - Inroads, Vol. 33, No. 4.

[5] Kramer, Jeff. 2007. Is Abstraction the Key to Computing? CACM, Vol 50, No 4.

[6] Wing, Jeannette 2006. Computational thinking. CACM, Volume 49, No. 3.

[7] Denning, Peter, and Andrew McGettrick 2005. Recentering Computer Science. CACM, Vol 48, No 11.

[8] CS Unplugged. http://csunplugged.com.

[9] Astrachan, Owen 2009. Pander to ponder. Proceeding of SIGCSE '09, Chattanooga, TN.

[10] Cortina, Thomas 2007. An introduction to computer science for non-majors using principles of computation. Proceedings of SIGCSE '07, Covington, KY.

[11] Soh, Leen-Kiat, et al. 2009. Renaissance Computing: An Initiative for Promoting Student Participation in Computing. Proceedings of SIGCSE '09, Chattanooga, TN.

[12] Hambrusch, Susanne, et al. 2009. A multidisciplinary approach towards computational thinking for science majors. Proceedings of SIGCSE '09, Chattanooga, TN.

[13] McMaster, Kirby, et al. 2008. Two gestalts for mathematics: logical vs. computational. ISEDJ, 6 (20).

[14] Gries, David, et al. 2001. How mathematical thinking enhances computer science problem solving. ACM SIGCSE Bulletin, Vol 33, No. 1.

[15] Henderson, Peter 2003. Mathematical reasoning in software engineering education. CACM, Vol 46, No. 9.

[16] Henderson, Peter, et al. 2003. Materials development in support of mathematical thinking. ACM SIGCSE Bulletin, Vol. 35, No. 2.

[17] Armoni, M. and 2006. Reduction – an abstract thinking pattern. Proceedings of SIGCSE '06, Houston, TX.

[18] Hill, Jonathan, et al. 2008. Applying abstraction to master complexity. ROA '08, Leipzig, Germany.

[19] Kramer, Jeff, and Orit Hazzan, 2006. The Role of Abstraction in Software Engineering. ICSE Companion 2008.

[20] Mishali, Oren, et al. 2008. Towards IDE support for abstract thinking. ROA '08, Leipzig, Germany.

[21] Astrachan, Owen, et al. 2009. The present and future of computational thinking. Proceedings of SIGCSE '09, Chattanooga, TN.

[22] Guzdial, Mark 2008. Paving the way for computational thinking. CACM, Vol 51, No. 8.

[23] Lu, James, and George Fletcher, 2009. Thinking about computational thinking. Proceedings of SIGCSE '09, Chattanooga, TN.

[24] Denning, Peter 2009. Beyond computational thinking. CACM, Vol 52, No 6.

[25] Bain, Ken 2004. What the Best College Teachers Do. Harvard University Press.

AUTHOR INFORMATION

Kirby McMaster, Visiting Professor, Math & CS Dept., St. Mary's College of Maryland, kmcmaster@smcm.edu

Brian Rague, Associate Professor, CS Dept., Weber State University, brague@weber.edu

Nicole Anderson, Assistant Professor, CS Dept., Winona State University, nanderson@winona.edu