

Sponsoring Committee: Susan M. Merritt, PhD, Dean, Committee Chair
Bernice J. Houle, PhD, Associate Dean
Allen Stix, PhD, Professor of Computer Science

**Applying Abstraction to Master Complexity:
The Comparison of Abstraction Ability in Computer Science Majors with Students
in Other Disciplines**

Jonathan H. Hill

Submitted in Partial Fulfillment
of the requirements for the degree of
Doctor of Professional Studies in Computing
The Seidenberg School of Computer Science and Information Systems
Pace University
New York, USA
2007

UMI Number: 3430590

All rights reserved

INFORMATION TO ALL USERS

The quality of this reproduction is dependent upon the quality of the copy submitted.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if material had to be removed, a note will indicate the deletion.



UMI 3430590

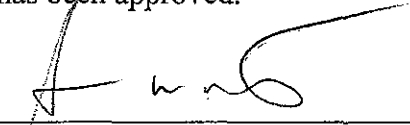
Copyright 2010 by ProQuest LLC.

All rights reserved. This edition of the work is protected against unauthorized copying under Title 17, United States Code.



ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

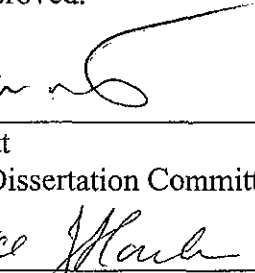
We hereby certify that this dissertation, submitted by Jonathan H. Hill satisfies the dissertation requirements for the degree of Doctor of Professional Studies in Computing and has been approved.



Dr. Susan Merritt
Chairperson of Dissertation Committee

9/24/07


Date



Dr. Bernice Houle
Dissertation Committee Member

9/24/07

Date



Dr. Allen Stix
Dissertation Committee Member

9/24/07

Date

Seidenberg School of Computer Science and Information Systems
Pace University 2007

I hereby guarantee that no part of the dissertation which I have submitted for publication has been heretofore published and/or copyrighted in the United States of America, except in the case of passages quoted from other sources; that I am the sole author and proprietor of said dissertation; that the dissertation contains no matter which, if published, will be libelous or otherwise injurious, or infringe in any way the copyright of any other party; and that I will defend, indemnify and hold harmless Pace University against all suits and proceedings which may be brought and against all claims which may be made against Pace University by reason of the said publication

A handwritten signature in black ink, reading "Jonathan H. Hill". The signature is written in a cursive style with a large, sweeping initial 'J'.

Jonathan H. Hill

ABSTRACT

Aptitude for managing abstraction may be a distinguishing characteristic of computer science majors. If this is so, and if this aptitude can be recognized among potential majors, those who are well suited for computer science but have not considered it as a major can be made aware of the possibility. Abstraction, as a human ability, is comprised of two complementary aspects: clearing away details to build simplifications and deriving generalizations that illuminate essentials. Agreement exists that this ability *may be nurtured through instruction and experience, but that it rests upon a natural aptitude that is possessed by few.* Agreement exists that this natural aptitude is assessable, although no instrument yet exists for measuring it efficiently among *prospective computer science majors who have not begun computer science* coursework. Nor has one existed to test abstraction skills among a general population of undergraduate students. This dissertation is focused on a study done at New York's Pace University to test undergraduate students across a range of majors for abstraction ability.

Acknowledgements

The completion of a doctoral dissertation marks the milestone in a journey. My own journey in academia is currently two decades long and, having reached this milestone, I gratefully acknowledge the mentorship, love and support of some wonderful people:

Dr, Stuart Schulman of the City University of New York started me on this journey and believing that because he could do it so well, anybody else could, gave me the chance to teach and learn and develop into a 'professor'. I am indebted to a generation of students at Kingsborough Community College and Queens College of the City University of New York for letting me learn how to teach in their presence. I am also grateful for the opportunities I was given to build things – programs, curricula, relationships, internships, seminars – that brought learning and self realization to some very fine people.

I have been blessed with the opportunity to continue my journey at Pace University and to have the opportunity to work with many marvelous students, staff and faculty there at the Seidenberg School of Computer Science and Information Systems. Under the direction of Dr. Susan M. Merritt, Dean and the chair of the committee for this dissertation, I have been able to grow in many

ways. Sue is a constant example of how much the combination of a fine intellect and a great heart can accomplish.

I have, likewise, been blessed with a 'dream' committee who gave me far more constructive time and input than any doctoral candidate has the right to expect. This study would not have been possible without the talents, insights and encouragement of the other members of my committee. Bernice J. Houle, PhD, Associate Dean of the Seidenberg School who is always at the top of her game and has been a dear friend and supporter professionally and personally. Her knowledge of many things has informed this study. Allen Stix, PhD, Professor of Computer Science is a man of great ideas who patiently encouraged me to rethink, rewrite and say things in a more elegant way. Working with him has been a great pleasure and a valuable learning experience.

Dr. Jack Schwarz, Emeritus of New York University, provided input and modeled what it takes to be a great scholar. Dr. OCZ Gotel of Pace University also modeled the effort it takes to create a quality scholarly work. I am much better for my time with these fine people.

The staff and faculty of the Seidenberg School have been an important source of encouragement for me. Freddy Peña, Kim Brazaitis, Carey Miller and Fran O'Gara have picked up the slack and created the space for me to do this work.

As a doctoral candidate, I was further blessed by my colleagues in the DPS 2007 cohort. My teammates, Anne Manette-Wright, Matt Ganis and Henry Wong were talented and collegial teammates who did much to make the course work meaningful.

My dear family was, and remains, the bedrock of my inspiration. My parents, Jesse and Audrey, instilled a love of learning and created a home where reading and intellect were prized for me and my dear sisters; Margaret Ellen and Cassie Ann, who have always encouraged my desire to succeed. I do hope that this may make them proud of me.

My grandmother, Cassie Ann Hill, whose sheer brilliance was passed down to a small extent to me, and in larger part to my children. Her daughter, my recently departed Aunt, Orvetta D. Hill, who had the capacity for both hard work and inner peace helped to make this – and much in my life – possible, and I think she would be pleased to know that I have earned my doctorate. Her unfailing belief in me has always been a source of strength.

My children, Luisa Maria Orvetta, Jesse Alejandro and James Cuthbert are my delight and inspiration and, in their curiosity about what I was writing, have contributed clarity to what you will read here.

Especially to my dear wife Ana Maria, who has been my friend and companion through this journey and encouraged me through multiple degree programs, my eternal love and gratitude. The journey has been fun and it is going to get even better!

Kendall Park, New Jersey

September 25, 2007

**Applying Abstraction to Master Complexity:
The Comparison of Abstraction Ability in Computer Science Majors with
Students in Other Disciplines**

TABLE OF CONTENTS

ABSTRACT		i
ACKNOWLEDGEMENTS		ii
TABLE OF CONTENTS		iv
DEFINITION OF TERMS		vii
LIST OF TABLES		ix
LIST OF FIGURES		xii
CHAPTER		
I	Introduction	
1.1	Introduction	1
1.2	Statement of the Problem	2
1.3	The Research Objective	4
1.4	Rationale for the Study	5
1.5	Definitions of Abstraction	6
1.6	Conceptual Framework	10
1.7	The Research Questions	13
1.8	Limitations of the Study	14
1.9	Summary	15
II	Review of the Literature and History of Abstraction	
2.1	Introduction	16
2.2	Abstraction as a Cognitive Function	16
2.3	Abstraction in the Computer Science Literature	21
2.3.1	Historical Foundations of Abstraction In Computer Science	21
2.3.2	Abstraction in the High Level Programming Languages	24
2.3.2	Abstraction in Object-Oriented Programming	26

	2.3.3	Recent Research in Computer Science and Abstraction	28
	2.3.4	Abstraction in Computer Science Pedagogy	29
2.4		Kramer's Model and Question Patterns for Measuring Abstraction	31
2.5		Discussion and Summary of the Literature	34
III		Research Instrument Development and Deployment	
	3.1	Introduction	36
	3.2	The Setting	37
	3.3	The Population for the Study	38
	3.4	Theoretical Underpinnings of the Instrument	39
	3.4.1	Testing the Three Scales of Abstraction	41
	3.4.2	Developing the Questions	42
	3.5	Data Collection	44
	3.6	Steps in Instrument Development	45
	3.7	Statistical Analysis	46
	3.8	Additional Questions Addressed in the Study	48
	3.9	Summary	48
IV		Analysis of Data	
	4.1	Introduction	50
	4.2	Comparison of Population with Respondents	50
	4.3	The Survey Instrument and the Survey Questions	52
	4.3.1	Section One – Identifiers	52
	4.3.2	Section Two – Conceptual Abstraction	56
	4.3.3	Section Three – Activities	58
	4.3.4	Section Four – Formal Abstraction	60
	4.3.5	Section Five - Descriptive Abstraction	66
	4.3.6	Section Six – Individual Information	70
	4.4	Grouping of Majors	71
	4.5	Experimental Results in SPSS	73
	4.5.1	Comparing Discipline with Abstraction Abilities	73
	4.5.2	Comparing Abstraction Abilities	77
	4.5.3	Comparing the Effect of Gender on Abstraction	80
	4.5.4	Comparing the Effect of Program on Abstraction	83
	4.6	Summary	86

V	Findings, Interpretations and Implications of This Study	
5.1	Introduction	87
5.2	Review of the Study	87
	5.2.1 Impact of One's Discipline on Abstraction Skills	89
	5.2.3 Results of Studying the Three Scales of Abstraction	90
5.3	Testing the Hypotheses	93
	5.3.1 Additional Questions	95
5.4	Key Findings of the Study	98
5.5	Implications of the Study	98
5.6	Recommendations for Future Research	100
5.7	Opportunities for the Computer Science Discipline	102
5.8	Conclusions	103
APPENDICES		
	A. Survey	104
	B. Letter from Bob Birrer of Harris Interactive	113
	B. References	114

DEFINITION OF TERMS

Abstraction – an evaluative skill that enables the user to identify, focus on and feature the most important parts of an entity, while ignoring or minimizing the less important parts.

Computer Science – The academic discipline of computer science as it is commonly taught and understood at the undergraduate level in general and at Pace University in particular. It involves deriving symbolic representations of processes (algorithms) and systems (software design).

Conceptual Abstraction – The name for the scale that measures a self-reported inclination toward abstract thinking (i.e. a self-reported assessment of abstraction aptitude). Scale scores are based on items 5, 6, and 7 in the survey instrument and range from 3 to 15.

CS 1 – An all-encompassing term referring to the Introduction to Programming course that is a standard requirement for majors in most computer science programs at American Universities.

CS 2 – The second, or follow-on course to CS 1, in programming that is a standard requirement for majors in most computer science programs at American Universities.

Descriptive Abstraction – The name for the scale formulated by the author that measures the ability to discern the characteristics of chief importance in a situation and, therefore, to formulate accounts that are meaningful though abbreviated. This same skill underlies the representational modeling required for implementing algorithms and building software systems.

Discipline – An aggregation of majors requiring similar problem solving skills and focused on content of a similar nature, for instance the social sciences and education in contrast to finance and accounting. Majors were aggregated into disciplines for the purpose of statistical analysis

Formal Abstraction – The name for the scale formulated by the author that measures the ability to perceive and reason about the structure of a symbolic system and, therefore, to deduce simplifications allowing for previously obscure properties to be evident and/or streamlined manipulative possibilities. Scale scores are based on items 9 – 14 in the survey. Scores range from a low of 0 to a possible high of 6. No respondent attained the theoretical maximum.

Major – A traditional undergraduate academic major such as computer science, philosophy or nursing.

Program – A rough grouping of Pace students by academic qualification. Students at the lowest level are in the Challenge to Achievement at Pace (CAP) program,. Students at the highest level are in the Honors program. Students at the middle level are termed traditional.

Scales of Abstraction – The three key aptitude scales measuring abstraction formulated by the author and described above: the conceptual abstraction scale, the descriptive abstraction scale, and the formal abstraction scale.

List of Tables

Table	Title	Page
1	Respondents by Gender	51
2	Year in School	51
3	Program Affiliation of Respondents	52
4	Course Sections	52
5	Majors of Respondents	54
6	Double Majors	55
7	Responses of Question 5	56
8	Responses of Question 6	57
9	Responses of Question 7	58
10	Responses of Question 8	59
11	Responses of Question 9	60
12	Responses of Question 10	61
13	Diagram from Question 11	62
14	Responses of Question 11	63
15	Responses of Question 12	64
16	Responses of Question 13	65
17	Responses of Question 14	66
18	Responses of Question 15	67
19	Responses of Question 16	68
20	Responses of Question 17	69

21	Responses of Question 18	70
22	Grouping of Academic Majors by Discipline	72
23	Results of Discipline and Conceptual Abstraction Crosstabulation	74
24	Results of Discipline and Conceptual Abstraction Chi-square analysis	75
25	Results of Discipline and Formal Abstraction Crosstabulation	75
26	Results of Discipline and Formal Abstraction Chi-square Analysis	76
27	Results of Discipline and Descriptive Abstraction Crosstabulation	77
28	Results of Discipline and Descriptive Abstraction Chi-square Analysis	77
29	Results of Formal Abstraction and Conceptual Abstraction	78
30	Results of Formal Abstraction and Conceptual Abstraction Chi-square Analysis	78
31	Results of Conceptual Abstraction and Descriptive Abstraction Crosstabulation	79
32	Results of Conceptual Abstraction and Descriptive Abstraction Chi-square analysis	80
33	Results of Formal Abstraction and Conceptual Abstraction Crosstabulation	80
34	Results of Formal Abstraction and Descriptive Abstraction Chi-square Analysis	81
35	Results of Conceptual Abstraction and Gender Crosstabulation	81
36	Results of Conceptual Abstraction and Gender Chi-square	82
37	Results of Formal Abstraction and Gender	82

Crosstabulation

38	Results of Formal Abstraction and Gender Chi-square	82
39	Results of Descriptive Abstraction and Gender Crosstabulation	83
40	Results of Descriptive Abstraction and Gender Chi-Square	83
41	Conceptual Abstraction and Program Crosstabulation	84
42	Conceptual Abstraction and Program Chi-square	84
43	Formal Abstraction and Program Crosstabulation	85
44	Formal Abstraction and Program Chi-square	85
45	Descriptive Abstraction and Program Crosstabulation	85
46	Formal Abstraction and Program Chi-square	86

List of Figures

Figure	Title	Page
1.1	Three Scales of Abstraction	12
2.1	The London Underground	32
2.2	Naked Blue IV by Henri Matisse	33
4.1	Graphic for High-Q Game	65
5.1	Adapted Three Scales of Abstraction	100

**Applying Abstraction to Master Complexity:
The Comparison of Abstraction Ability in Computer Science Majors with
Students in Other Disciplines**

Chapter I - Introduction

"We (computer scientists) have become so good at defining and manipulating abstractions that we hardly notice how skillfully we create abstract "objects," which upon deployment in a computer, perform useful actions. Students need to see from the beginning how to connect abstractions to actions."

Peter J Denning [28]

1.1 Introduction

The context of, and prime motivation for, this study is the decline in the number of students electing to major in computer science. This decline has been taking place over the past five years. It is well documented [6, 26, 31] and a source of concern to academia, industry and government. If there is a distinguishing aptitude characteristic of the discipline, and if this aptitude could be recognized among new freshmen, perhaps students with potential, who may not have considered majoring in computer science, could be encouraged to do so.

The study was based on a concept, increasingly mentioned in the literature, that the "key" factor (Jeff Kramer's term) [58] required for success in computer science is an aptitude for working with abstractions [12, 19, 29, 45, 46, 75, 81, 94, 96, 106]. In one sense, this is an ability, in a software design setting, to discern and create simplicity in the face of complexity. In another sense, it is an ability to conceptualize up the abstraction ladder to create software hierarchies of different varieties [21].

The intent of this investigation was to discover whether computer science students do, in fact, possess a stronger aptitude for abstraction than students in other majors. Researchers have investigated the aptitude for abstract thinking among computer science majors with exercises using concepts from computer science. No other researchers have attempted to gauge this ability across majors as is done here.

Because work up to now investigated the aptitude for abstract thinking among computer science majors with measures using concepts and terminology from computer science, it cannot be directly harnessed to measure the same aptitude among students who are pursuing other majors. This study has devised an instrument to measure abstraction aptitude among undergraduate students in all majors. The investigation attempts to determine whether computer science students score higher on this measure than students in other disciplines.

A secondary contribution from this study is the creation of a model of abstraction based on the work of Jeff Kramer [46, 58] that supports research focusing on the relationship between computer science and abstraction. This model is described and illustrated in this chapter in section 1.6, Conceptual Framework and evaluated in chapter 5, section 5.3, Results of Hypotheses.

1.2 Statement of the Problem

As the number of undergraduate college students electing to study science, engineering and, particularly, computer science, has continued to decline, the need to identify both inherent traits and skill sets that are potential

indicators of success for computer science students has intensified. This study has been undertaken to extend the exploratory work that has begun in the field of computer science. Since abstraction is a core skill in computer science that may be a measurable skill, this study investigates whether students of computer science have better abstraction skills than students in other majors. This work is important because abstraction is an important part of the study of computer science and of the software development process. So important that abstraction is a concept addressed in the curriculum of CS 1 and CS 2 courses in many college and university computer science departments [5, 10]. If abstraction skill can be measured in any student, not just computer science majors, it would create opportunities to identify students who have strong abstraction skills for recruitment to computer science programs. It could also identify students who could benefit from remediation in their abstraction skills, thus creating more successful students of computer science and therefore increasing retention.

Abstraction has been explored extensively in the computer science literature and has received a lot of attention from industry practitioners as a theoretical tool for parsing information and as a practical problem solving technique [27, 45, 46, 58, 76, 88]. As a result, abstraction appears to be widely accepted as a heuristic that can be applied in the discipline.

Recent literature in the field has discussed the need to test an individual's aptitude for finding and making abstractions, as well as measuring abstraction abilities among populations of students and practitioners [12, 45, 58]. However, the movement to test abstraction remains in a developmental stage. In recent

work, Bennedsen and Caspersen say that “to our knowledge, no research has been conducted to verify whether abstraction ability is actually a predictor of success....” [12, p. 157]

This study focuses on the hypothesis that students who have declared computer science as a major will test higher on all three scales of abstraction than students who have declared other majors. This research was undertaken with the intent that findings would assist teachers of computer science, students, guidance counselors, computer science departments, university recruiters and other stakeholders in the quest for measurable skill sets in the computer science discipline.

1.3 The Research Objective

The role of abstraction is central to the study and practice of computer science. From this premise, this study involves an analysis of the role of abstraction in computer science, the exploration of three measures of abstraction aptitude, the analysis of their operationalization, and the test of several hypotheses relating to the abstraction aptitude of computer science and other majors. This study was, in part, an effort to measure abstraction ability in three areas among a broad cross section of undergraduate students and to compare their relative abstraction abilities.

This study developed a research tool to test the abstraction skill levels among a cross section of undergraduate students at Pace University in New York, including both the New York City and Westchester campuses. The results

of the survey have been examined to explore the relationships between students in different majors and their relative abstraction skills. Recommendations for continuing work and further research have been made based on the results.

1.4 Rationale for the Study

There are not enough students interested in studying computer science in the United States to support the computing needs of the U.S. economy [31]. Along with the troubling shortage of entering college and university undergraduates with the mathematical skills required for success in many disciplines, the shortfall of students enrolling in computer science, engineering and the sciences has received considerable attention in the U.S media in recent years [41, 67]. Some have speculated that the shortfall of students encourages lower standards, which then discourages those with the greatest aptitude and strongest disposition for challenge and hard work [6, 8, 14, 60 96, 105]. That this decline has also been experienced in the sciences and engineering is well documented in the literature [8, 31, 59]. So is the concern from the corporations who need a predictable stream of well-trained computer science graduates, as voiced by Nick D'Onofrio of IBM [31]. While high school students who are strong in mathematics (and who have been advised into advanced placement (AP) computer science courses), or science have been the source for many computer science majors at the undergraduate level [6], students with these strengths in recent years have been going into other fields such as business and finance

where students expect their quantitative skills to bring better compensation upon graduating [50, 63].

With the decline in the numbers of computer science students, it becomes imperative to identify those factors that may indicate that an individual is particularly suited to study in the discipline. Leading computer scientists, including Peter Denning and David Lorge Parnas [82], agree that the ability to do abstraction is integral to success in computer science. If that aspect of intelligence relating to abstraction is the "key to computing" as argued by Kramer [58, p. 38], then being able to identify it in prospective students is integral to concentrating recruitment efforts and incentives that will attract the most qualified students, students who will prosper in the major, enable standards to rise, and replenish the talent needed by industry.

The research includes a feasibility study on the potential for measuring abstraction skills in a test population of undergraduate students. To this end a survey was developed with questions modeled on, among other sources, the ground breaking work of Kramer and Hazzan [45, 50] in abstraction, featuring a series of questions designed to measure an individual's ability to think and work abstractly.

1.5 Definitions of Abstraction

Abstraction is an evaluative skill that enables the user to identify, focus on and feature the most important parts of an entity, while ignoring or minimizing the less important parts. Skilled abstractionists realize that the relative importance of

the individual parts of an object is dependant on the context in which they are viewed or which problems they will be called upon to solve. As a result, abstraction is a flexible tool and, when employed strategically, is used to identify, separate and use the most important ideas, features or concepts. Abstraction is particularly appropriate to computer programs because they are, by nature, not concrete but rather, aggregations of one dimensional strings of code (or objects) that are abstract representations of the tangible problems that they are meant to solve.

The role of abstraction in the teaching and practice of computer science is well recognized in the literature and pedagogy of the field [12, 19, 29, 45, 46, 76, 81]. However, the field acknowledges that the concept of “abstraction” is amorphous and can be applied to a variety of contexts, situations, problems, objects and programming practices. A defining feature of abstraction is its appropriate use in many different contexts.

Two researchers who have worked together to further define abstraction are Jeff Kramer of Imperial College in London and Orit Hazzan of the Technion in Haifa, Israel. Kramer, in his recent work on the topic for the *Communications of the ACM* (April, 2007), defines abstraction as “the act of withdrawing or removing something and the act or process of leaving out of consideration one or more properties of a complex object so as to attend to others [58, p. 38].” Orit Hazzan, defines abstraction as “...a cognitive means according to which, in order to overcome complexity at a specific stage of a problem solving situation, we

concentrate on the essential features of our subject of thought, and ignore irrelevant details [46, p. 158].”

The Computing Technology Association defines abstraction in the following way:

Abstraction - 1) “A process of identifying which details in a given context are essential, and should be visible, and which are non-essential and can be hidden “behind the scenes.” Abstraction also helps to identify common components of objects or processes that could be grouped in a way that makes those components or processes re-usable in more than one context without having to repeat all the details in every context. Abstraction is very useful when developing standards for business documents, where blocks of information like contact information, address information, etc. are repeated in many document definitions [25].”

2) “Another type of abstraction is the process of creating an abstract class. This is convenient when several types of a thing share common attributes or components, and you only want to define the components once. For example, you could define an abstract class `BasicForecastData` in which you describe attributes that are used in all types of forecasts, but you would never create an actual “`BasicForecastData`” document or file. Instead you would create a type of forecast, such as a `Material Release Replenishment Plan`. The `Replenishment Plan` class is a specialization of `BasicForecastData`; it uses all the attributes in `BasicForecastData` plus some additional attributes [25].”

Abstraction is a method of problem solving, a strategic tool applied in other disciplines. The Complete Wargames Handbook defines abstraction as a “Key concept in wargames whereby complex procedures in an historical event are much simplified in a wargame of the same event [33, p.76].”

Abstraction is often discussed in the context of abstract art where an object is reduced to its simplest form and most basic, recognizable qualities in the artist’s representation. The use of abstraction in art is comparable to the way in which computer scientists use abstraction in their discipline. The online site for

the Tate Museum defines abstract art this way: “the word abstract, strictly speaking, means to separate or withdraw something from something else. In that sense, it applies to art in which the artist has started with some visible object and abstracted elements from it to arrive at a more or less simplified or schematised form. A term also applied to art using forms that have no source at all in external reality [103].”

ProgressiveArt.com describes abstract art as “A 20th century style of painting in which nonrepresentational lines, colors, shapes, and forms replace accurate visual depiction of objects, landscape, and figures. The subjects often stylized, blurred, repeated or broken down into basic forms so that it becomes unrecognizable. Intangible subjects such as thoughts, emotions, and time are often expressed in abstract art form [86].” While we recognize many of the descriptions of abstract art in computer science, the thought that a piece of software might be written to be “blurred... intangible...or unrecognizable [86]” misses the point. Rather, abstraction in computer science is closer to what George A. Miller termed “recoding [73],” substituting one larger chunk of information (e.g. a subprogram name) for many smaller chunks (e.g. its set of statements) and what Charles Simonyi describes as layers of software, where each layer represents processes and information for the layers above and below [17].

Certain misconceptions about abstraction as a tool come from a casual misunderstanding of the concept of abstract art. Some observers associate abstract art with non-representational works that give little attention to the re-

creation of form. Rather, in computer science, abstraction draws inspiration from the disciplined, representational art created by masters such as Henri Matisse and Katsuhika Hosukai who made memorable art via disciplined, representational works that used form and color sparingly to represent images at their most abstract [103].

1.6 Conceptual Framework

This study was guided by recent research on abstraction as it is applied to computer science. Computer scientists, particularly those who focus on the teaching of computer science at the introductory level, have been working to develop an understanding of abstraction that would lead to an effective way of teaching students how to think abstractly [5, 10, 14]. Since abstraction is discussed in the literature as a thinking process, but also as an applied skill, how, then, does a student move from a conceptual understanding of abstraction to applying abstraction in the traditional Model, View, Control (MVC) levels utilized in object-oriented programming [54]? How can a researcher test and evaluate a student's conceptual grounding in abstraction and ability to use abstraction in an applied way?

Orit Hazzan and Jeff Kramer are two researchers from the field of computer science who have written extensively on the need to develop an instrument to test abstraction [45, 46, 58]. Their work has examined the presence of abstraction in the computer science curricula as well as the challenges in trying to teach it at the undergraduate level. Their work has focused on the identification of the kinds of tasks that are particularly suited to

abstraction and may therefore enhance students' abstract thinking. They are working from the recommendations of the ACM/IEEE-CS Task Force on Computing Curricula 2001 for Computer Science, beginning with the premise that "computer science draws its foundations from a wide range of disciplines all computer science students must learn to integrate theory and practice, to recognize the importance of abstraction and to appreciate the value of good engineering design [45, p. 3 – 4]."

The conceptual framework for this study has roots in the study of abstraction as a cognitive ability, based upon the tests developed by Adey and Shayer in the 1980's [1,12] in the United Kingdom and manifested in the recent work from Kramer and Hazzan [46, 58]. Their work has led to the identification of three scales of abstraction: conceptual abstraction, formal abstraction and descriptive abstraction.

Conceptual abstraction measures abstraction as it may be understood from an individual's orientation within a big picture vs. small detail context. An individual with strong abstraction skills can move effectively between both orientations. Formal abstraction and descriptive abstraction flow from conceptual abstraction, as shown in figure 1, and are specializations in problem solving and language manipulation, respectively.

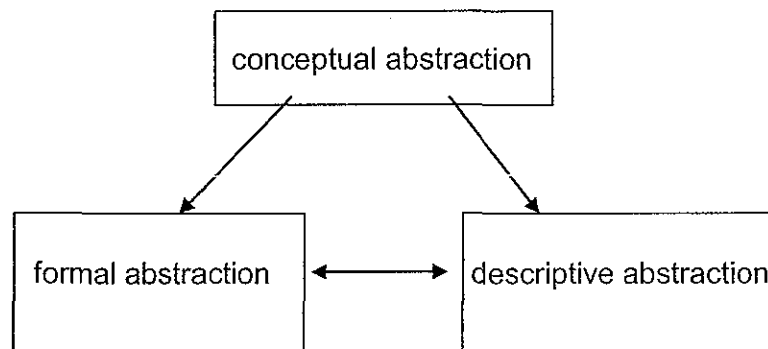


Figure 1.1. Three Scales of Abstraction

According to Cohn, [23], conceptual abstraction is a foundation of scientific process and inquiry, providing, “commonly understood ‘benchmarks’” that allow terminologies to be “concept oriented [23, p.299].” Cohn says of conceptual abstraction; “the first of these that we decided to apply was conceptual abstraction, namely, the elements of the terminology or coded concepts, possibly multiple synonymous text representations and hierarchical or definitional relationships to other concepts. There are no redundant, ambiguous or vague concepts in terminology, or none that are not recognized [23].”

Formal abstraction is the ability to reason about symbolic situations or symbolic constructions in order to derive, deduce, or perceive the underlying structure and/or a more straightforward (or simplified) view. The simplification may allow for new or streamlined manipulative possibilities or factual properties that were previously too obscure to be evident. It must avoid operative distortion or the introduction of error. It is often tested with traditional quantitative or word *problem-type questions* [50, 54, 83].

Descriptive abstraction is the ability to discern characteristics of chief importance and, therefore, to construct generalized accounts or models that are meaningful yet abbreviated. It identifies the highlights, focuses on incisive analysis, cuts right to the core and perceives essences. When a collection of entities is grouped because of similarities or by way of contrast, descriptive abstraction enables salient unifications and/or differentiations. It is manifested in word problems as well as in descriptive ways like writing.

1.7 The Research Questions

The main focus of this study was to determine if computer science majors have stronger abstraction skills than other majors. The study will also test the model used to illustrate abstraction. The hypotheses are the following:

Hypothesis 1: Students who have declared computer science as their major will score higher than students in other majors on measurements of all three scales of abstraction.

Hypothesis 2: Conceptual abstraction (a self-reported inclination toward abstract thinking), is significantly related to formal abstraction (removing detail in order to simplify), operationalized as the ability to reason about symbolic situations or symbolic constructions in order to derive, deduce, or perceive underlying structure or a more straightforward (simplified) view.

Hypothesis 3: Conceptual abstraction is significantly related to descriptive abstraction (generalization so that one representation portrays multiple examples), operationalized as the ability to discern characteristics of chief

importance and, therefore, to construct generalized accounts or models that are abbreviated and meaningful.

Hypothesis 4: The relationship between formal abstraction and descriptive abstraction is weaker than the relationship between either of these and conceptual abstraction. The rationale is that these are distinct manifestations of conceptual abstraction and the inclination toward either could account for a high score on the scale of conceptual abstraction.

1.8 Limitations of the Study

In the realm of computer science, abstraction is a broadly defined term relating to a 'soft skill' that is widely held to be an important skill in software development and, thus, an area of focus in computer science pedagogy. As a result, definitions of abstraction are quite context specific and thus the definitions used in the study may not be a fit in other contexts.

In addition, the subjects who responded to the survey instrument were all undergraduate students at Pace University. Therefore, the tests returned might not be representative of the larger population of undergraduate college and university students. As a result, care should be used when applying the findings of this study to other populations.

The researcher has endeavored to pilot a set of scales for recognizing abstraction aptitude in individuals. Those seeking to replicate these results need to be cognizant of the fact that the focus of this study is exploratory.

1.9 Summary of Chapter 1

Aptitude for managing abstraction may be a distinguishing characteristic of computer science majors. If this is so, and if this aptitude can be recognized among potential majors, those who are well suited for computer science but have not considered it as a major can be made aware of the possibility.

While opinion of abstraction's centrality in computer science and software design is widely shared, Jeff Kramer strove to define it and to provide the theoretical underpinnings. Working with Orit Hazzan, Jeff Kramer worked to specify how an aptitude for framing abstractions might be tested. Their testing regimen, however, relied upon background that only computer science majors would have.

Extending Hazzan's work, this study is focused on testing if computer science majors have stronger abstraction skills than other majors among a general group of undergraduate students. As all research, this study has defined limitations. Abstraction aptitude is found to be unusually prevalent among computer science students, and recognizing this ability among undergraduate students, with no prior courses in computer science, is possible.

CHAPTER II

REVIEW OF THE LITERATURE AND HISTORY OF ABSTRACTION

2.1 Introduction

This study was built upon the role of abstraction as a foundation in the teaching and practice of computer science. Abstraction is a recognized and important cognitive ability that has been identified by a number of psychologists as a tool in the discipline of computer science [73, 75, 90]. Much of the literature on abstraction can be divided into two categories: that which examines the cognitive aspects of abstraction and that which deals with abstraction as it is applied in the discipline of computer science and the field of software development. This chapter discusses the role that abstraction plays in computer science as it was initially conceived by great theoreticians in the discipline represented by Donald E. Knuth, David Lorge Parnas, and Edsger W. Dijkstra. Also described are recent investigations of abstraction ability in the context of computer science education with attention to aspects of methodology that contrast with, or bear upon, the design of this study.

2.2 Abstraction as a Cognitive Function

"Is abstraction the key the key to computing?" Jeff Kramer posed this rhetorical question as the title of an article featured in the influential *Communications of the ACM* [58]. His thesis, based upon over 30 years of experience teaching computer science and software engineering, is that "the

ability to perform abstract thinking and to exhibit abstraction skills" [58, p. 38] is the catalyst responsible for making certain students great computer scientists.

The nature of this ability we call abstraction has a long history in the *intellectual discourse*. The English philosopher John Locke discussed abstraction in detail more than two centuries ago. The Swiss interdisciplinarian Jean Piaget developed concepts of abstraction as a cognitive skill that are widely accepted [18] and inform the computer science-specific work of Kramer [58]. John Locke discussed his own abstraction skills when he said, "whether others have this wonderful faculty of abstracting their ideas, they can best tell; for myself, I find indeed I have a faculty of imagining, or representing to myself, the ideas of those particular things I have perceived, and of variously compounding them. I can imagine a man with two heads, or the upper parts of a man joined to the body of a horse, I can consider the hand, the eye, the nose, each by itself abstracted or separated from the rest of the body. But then whatever hand or eye I imagine, it must have some particular shape and colour [90, p.125]." According to Roecklein, Locke's work contributed to the common understanding of how abstract concepts such as shapes and color are perceived, "Locke's concepts of abstraction were controversial with his contemporaries: The controversy about how the brain or the mind represents abstract ideas such as the general concept of a color or a circle, square, or triangle is older than psychology as an independent scientific discipline [90, p.149]." This work subsequently influences the definition of the three scales of abstraction discussed in this study.

Kramer [58] examines the process of cognitive development in human beings and draws on the work of Jean Piaget (1896 – 1980), the Swiss psychologist, biologist and philosopher whose work has been fundamental in many disciplines that study human development including psychology, education, biology and sociology to examine the process of cognitive development from childhood to adulthood. Piaget's work identified four stages of cognitive development: sensorimotor in infancy, pre-operational in early childhood, concrete operational in mid-childhood (ages 7 – 12) and formal operational from age 12 through adulthood. An individual's learning ability, particularly in the formal operational stage, is manifest in the way computer science students gather and process information and the way they apply that information to computer science problem solving processes [18, 58].

Piaget made significant contributions to the field of cognitive science. Piaget's work is a foundation for the way that abstraction is understood as a skill in the computer science literature. Piaget found that only 30% - 35% of adolescents reached the fourth, or formal operations stage of cognitive development – and that some adults never do. Development during this fourth stage is critical to the development of abstraction skills in people (and an abstraction deficiency may be a reason why many people don't 'get' computer science). Reasoning abilities developed during this stage include hypothetico-deductive reasoning, scientific-inductive reasoning and reflective abstraction. In a posthumous 1981 publication, Piaget [73] described the capacity for hypothetico-deductive reasoning as the ability to be able to deal with not only

objects and experiences but with hypotheses as well, with "the possible as well as the real" This allows a person to draw conclusions from a hypothesis – not just a tangible object. The ability to reach conclusions by going from general to specific is called deductive reasoning.

The Austrian philosopher Ernst Von Glaserfeld, in his 1991 treatise on reflection and abstraction in the work of Locke and Piaget states that "a program is the fixed itinerary of an activity that can guide and govern the sequence of its reenactment. But there are two points to be stressed. First, a program may specify what material on which to act, but it does not supply the material. Second, a program may specify what acts are to be performed, but it supplies neither the acting agent nor the action [109, p. 12]." Von Glaserfeld summarizes the work of Locke by stating that "Locke believed that the sensory source of ideas, the 'impressions' generated by 'outward objects' provide the mind with some sort of picture of an outside world [109, p. 36]." Piaget saw perception as the result of the subjects actions and mental operations aimed at providing, not a picture of, but an adaptive fit into the structure of that outer world [109].

The 20th Century Austrian economist Friedrich Hayek talked about the "primacy of the abstract" interpreted by him to be a set of abstract rules that human societies live by that are inherently understood by individual members of that society. Hayek goes on to introduce 'abstraction boundaries' which allow for specialized knowledge to be gathered in areas of specialization. Hayek describes abstractions as "a means to cope with the complexity of the concrete which our mind is not capable of fully mastering [109]." Hayek did significant

work in developing a theory of the human mind, which he saw as a complex system of classifications and abstract rules.

This concept of abstraction boundaries has been adapted to the realm of software engineering where abstraction as a tool for plan coordination has become a norm. Hayek's work helped to inspire the computer scientist turned economist Don Lavoie (1950 – 2001) to launch the Agorics Project [61], a means to utilize computer models to explore abstractions in market forces and economics.

Don Norman, the engineer, computer scientist and cognitive psychologist at Northwestern University has done seminal work to describe the role that abstraction plays in learning, memory, analysis and, importantly for this study, the effective application of cognitive learning to problem solving [76, 78]. Norman's work is particularly useful in that it examines human cognitive development within the context of technology. The work on cognitive development and the role of representation in both individual and collective learning provide significant analysis of the role of abstraction in learning and indicate that abstraction is an important tool. In Norman's analysis, individuals' minds develop cognitive 'artifacts' from the environment that develop 'representations' that capture the essential elements of the event or concept by 'abstracting away' the irrelevant details [76].

The lack of abstraction skill can be a serious impediment to full-functioning by an adult. A *New York Times Magazine* article discussed Williams Syndrome: a genetic disorder where "The resulting cognitive deficits lie mainly in the realm of

abstract thought. Many with Williams have so vague a concept of space, for instance, that even as adults they fail at six-piece jigsaw puzzles, easily get lost, draw like a pre-schooler and struggle to replicate a simple T or X shape built with a half-dozen building blocks. Few can balance a checkbook [30].”

2.3 Abstraction in the Computer Science Literature

Much of the literature on abstraction can be divided into two categories; that which examines those aspects of abstraction that are part of human cognitive development and those that deal with abstraction as an applied skill in the discipline of computer science and the field of software development. In developing this study, the researcher has explored the historical underpinnings of abstraction in computer science, its application in the area of the early ‘high level’ programming languages and its current role in the object oriented paradigm.

2.3.1 Historical Foundations of Abstraction in Computer Science

The earliest computing people tended to be quantitative types (e.g. John von Neumann, Alan Turing, Grace Murray Hopper, John Warner Backus, and Donald Edwin Knuth) and the earliest applications tended to be numerical (ballistics, engineering, science, and statistics). The foundation literature in computer science and software design looked mathematical, from treatises on algorithm analysis and compiler construction to the classic piece on structured programming by Bohm and Jacopini [17]. Mathematical aptitude was even used by computer science departments for screening prospective majors. As late as 2003, the year that Ventura's investigation found that verbal SAT score was a

better predicted of success in computer science than math SAT, the math SAT remained in widespread usage for screening [103]. This may have been because mathematical ability was presumed prerequisite for parts of the curriculum and math SAT was the standardized indicator of mathematical ability on applications.

Donald Knuth's Turing Award Lecture of 1974, titled "Computer Programming as an Art," can be read as a discussion on abstraction in building software. The essence of Knuth's argument is that software is an art because "it is devised by the intellect," as opposed to something derived from nature. He elaborates by describing how the process of software construction is dependent upon ingenuity and accumulated expertise and not on general laws from a logically systematized body. Although the word "abstraction" is not highlighted, it is clear that what "ingenuity" is "devising" are abstractions.

Somewhat later Knuth addressed head-on the question of the kinds of thought processes that distinguish computer scientists. In fact, the keynote address he delivered in 1979 to an international symposium on Algorithms in Modern Mathematics and Computer Science [56] presaged the work of Ventura, Kramer, and Hazzan and others who later strove to discern that aptitude which is uniquely effective at predicting success in software. As an aside, it is curious that none of the later workers attempted to build upon his ideas, let alone even cite his foray into their territory.

Knuth's work was sparked by the finding that "only about 2 out of every 100 students enrolling in introductory programming courses really resonate with

the subject and seem to be natural born computer scientists [57, p. 87]." This led him to a study by Gerrit DeYoung, "a psychologist-interested-in-computer-science whom I met at the University of Illinois," that very credibly demonstrated a "lack of correlation between quantitative ability and programming performance." DeYoung's finding squared with Knuth's long-held feelings that mathematical thinking and computer-science are different and prompted him to conduct an investigation of his own, a content analysis of math treatises. As a result, Knuth stated the following:

"I think it is generally agreed that mathematicians have somewhat different thought processes from physicists, who have somewhat different thought processes from chemists, who have somewhat different thought processes from biologists. Similarly, the respective mentalities of lawyers, poets, playwrights, historians, linguists, farmers, and so on seem to be unique. Each of these groups can probably recognize that other types have a different approach to knowledge; and it seems likely that a person gravitates to a particular kind of occupation whenever a choice is possible [57, p.117]."

While the content analysis employed a typology of mathematical and computing features, nothing useful emerged. It is worth noting that while Knuth never returned to the problem of what distinguished those "2 out of every 100 students", he repeatedly dwelled on the differences between the mentalities of computer scientists and mathematicians.

Edsger W. Dijkstra also discussed these same issues in his 1975 essay, "Craftsman or Scientist?"

Sticking to the technology of the scientist means being as explicit as we possibly can about as many aspects of our trade as we can. Now the teaching of programming comprises the teaching of facts -- facts about systems, machines, programming languages, etc. -- and it is very easy to be explicit about them, but the trouble

is that these facts represent about 10 percent of what has to be taught: the remaining 90 percent is problem solving and how to avoid unmastered complexity, in short: it is the teaching of thinking, no more and no less. ...This, of course, raises the question of the feasibility of the teaching of thinking. Experience has made me a firm believer that this newer aspect of thinking, i.e. how to avoid unmastered complexity, can indeed be taught. [29, p. 107-108]

Dijkstra asserts that computer science is 90% art. Its distinctive thought processes revolve around reducing complexity. And this kind of problem-solving can be taught. Learning the art of mastering complexity is accomplished through experience, as in the apprenticeship to a craftsman.

2.3.2 Abstraction in the High Level Programming Languages

If abstraction is the removal of unnecessary detail in order to master complexity, early examples may be seen in FORTRAN and COBOL, the so called "high level" programming languages from fifty years ago. Interestingly, what was "high level" about them was precisely their level of abstraction relative to the operations on memory and registers offered by the hardware. These languages removed unnecessary detail by offering data abstraction and control abstraction.

Data abstraction refers to declaring variables and using their name, as opposed to an address and the bit string stored in a program. This simplified programming enormously. This entails much more than the substitution of a mnemonic for a number. It is the idea that a data type, such as an integer, a real, or a String, can be simulated, with all the detail hidden away, and the programmer given the symbolic construct of variables that obey algebraic

intuitions. It is important to emphasize that data types and variable do not exist naturally inside computers. They only exist as abstractions.

At the same time that data abstractions greatly simplify the mechanics of programming and the logical representation of input, assignment, and output; they add a cognitive burden. FORTRAN's float is not a real, it is an abstraction. While an algorithm may be capable of computing the square root of 2 to any number of places, a float will never give more than eight.

In addition, high level languages offered software developers the opportunity to create their own abstractions through the ability to write subroutines, units of code that could be activated by typing their name in the flow of control. A subroutine functions as a program for a program.

The challenge for subroutines was to perform effective module decomposition. The software developer needed to take a very general task, such as building a text editor, and figure out what the top level subroutines should be, what subroutines each of these would need to do their job, the subroutines needed by these subroutines, and so on. In his influential book, *The Mythical Man Month*, Frederick Brooks says "The programmer, like the poet, works only slightly removed from pure thought-stuff. Programmers build castles in the air, from air, creating by exertion of the imagination. Few media of creation are so flexible, so easy to polish and rework, so readily capable of realizing grand conceptual structures [16, p. 7]."

2.3.2 Abstraction in Object-Oriented Programming

The move from procedural programming to object-oriented programming can be seen as a move from concrete plans to abstract plans. Objects do not coordinate their actions with the specific plans of other objects; they coordinate their actions to the abstract aspects of those plans.

- Bill Tulloch, George Mason University [105]

In the late 1990's, computer science instructors began to shift their teaching methods to feature object-oriented programming languages such as Java. Bergin and others proposed paradigm-changing curricula using object oriented programming as the foundation to learning computer science [13, 14]. Object-oriented languages particularly lend themselves to the teaching of abstraction because of the need to identify data types, data structures and functions that can be applied to the data structure. In this way, by means of the abstraction process, data structures become objects that include data and functions. The efficiencies gained by the software developer who works in an object-oriented programming language have been demonstrated repeatedly [13, 21, 52, 63]. Abstraction makes this efficiency possible and is a key factor in how well the developer or software engineer uses objects to meet the goals of the project.

All objects are abstractions, like FORTRAN's floats. The chief difference is that they are vastly more elaborate, and there are vastly more objects in standard Java, around three thousand, than constructs in any procedural language. The abstractions for basic computation, information storage and retrieval, graphical user interfaces, data communications, security, and the like

are more real and important to programmers than the physical reality of the machine. They are the building blocks. They are the things one needs to learn about.

Capretz presents abstraction as an important tool in dealing with complexity in software systems, saying that “forming abstractions of an application in terms of classes and objects is one of the fundamental tenets of the object-oriented paradigm [21, p. 2].” He goes on to say that, like abstraction, object-orientation has been interpreted and defined in different ways, resulting in different methodologies. Capretz presents a model of object-orientation that is composed of four supporting elements including system simulation, operating systems, data abstraction, and artificial intelligence. He attributes this classification, in part, to Cardelli and Wegner’s work in the area of formal methods which helped to define the role of abstraction and abstract data in the object-oriented paradigm [21].

Liu, et al, in their 2005 discuss how object abstraction is essential in enabling the construction of complex systems from components [64], and note the efficiencies – and potential cost savings from a corporate perspective – that abstraction in the object-oriented development of software services offers. They also discuss the concept of aspect-oriented programming (AOP) as an excellent means of structuring abstraction.

In related work, The Agorics Project [61], a team of researchers at the *Center for the Study of Market Processes at George Mason University*, has been engaged in a comprehensive study of the software components industry. One

area of their work, which is related to this study, is the important set of technical advances in abstraction mechanisms, modularization techniques and object oriented technologies. Much of this group's work has focused on reusable software elements, the market commoditization of objects and the value of abstraction mechanisms in making software component reusable and resalable. Encapsulation of domain abstractions and information hiding, another element of abstractions are the ways in which these software objects can be reused and programmers can benefit from one another's experience.

2.3.3 Recent Research in Computer Science and Abstraction

Noteworthy precursors to Kramer's and Hazzan's work in abstraction includes the study by Or-Bach and Lavy [81] and the study by Bennedsen and Caspersen [12]. Or-Bach and Lavy's contribution is a typology for rating the quality of a derivation hierarchy designed by third year college students following the completion two semesters of coursework in object-oriented programming and design. The lowest quality design entailed an abstract class at the top of a Java derivation hierarchy that included only instance variables. In the middle quality design, instance variables and concrete methods had been factored into the top class. In the best design, the top class was an abstract class with instance variables, concrete methods, and abstract methods. Of the 33 participants, 4 constructed designs of top quality and 7 drew-up designs of middling quality. 20 of the remaining 22 could only do a design of the lowest quality, and 2 could not even do this. Their recommendation was for instructors to increase the number of examples seen in class along with more drill and practice. For Or-Bach and

Lavy, "abstraction" lies in the graduated factoring of classes up the design hierarchy, and they claim that the ability to accomplish this competently is teachable.

The Bennedsen and Caspersen study is interesting on several counts [12]. The first is the hypothesis, that abstraction ability is correlated with course grade in CS1. The second is the means used to measure abstraction ability, a test with eight gradations of "cognitive development" rooted in the developmental stages described by Jean Piaget. The third is the finding of no correlation *between abstraction ability and course grade coupled to the explanations proffered to account for this.*

Unlike Or-Bach and Lavy, for whom "abstraction ability" was literally tied to the design of an abstract class; for Bennedsen and Caspersen it pertained to a general mentality or thinking process, as in the interpretation of Knuth. Their study was the first empirical effort to explore the relationship of abstraction to students' success in CS1.

2.3.4 Abstraction in the Computer Science Pedagogy

Linda Wilkens lists abstraction as first among the most important concepts for first year computer science students to learn (she lists 'logical thinking' and 'programming as problem solving' as 2nd and 3rd) [112]. In the movement to make the first year programming class for undergraduates a more meaningful experience, many faculty have made the CS1 & CS2 courses as much about problem solving than programming. To further this end, Bergin [13] suggests introducing the object-oriented programming paradigm before the procedural

one. The fundamental principles of object-orientation, such as inheritance, polymorphism and encapsulation are supported by the concepts of abstraction.

Much of the literature addresses the valuable role that abstraction plays in the classroom teaching of programming. Sprague and Schahozenski discuss *Abstraction: the Key to CS1* and hypothesize that abstraction is the key to problem solving because abstraction (and symbol manipulation) are fundamental to computer science [96]. They also note abstraction's role in the modeling required in other disciplines like physics, chemistry and economics. In discussing how to teach abstraction through object-oriented programming, they point out the usefulness of abstraction in helping beginning computer science students to identify entities, characteristics of entities and information hiding [86].

Ventura took the model further in his 2003 [107] dissertation in which he reported on a study to confirm the hypothesis that general abstraction ability has a positive impact on student success in a CS1 course. The dissertation evaluated the identified predictors of success for an object-oriented first course and compared results with traditional procedural programming approaches. The predictors included prior programming experience, mathematical ability, academic and psychological variables, gender, and measures of student effort. Ventura's results showed that prior programming experience is not a predictor of success. Further, cognitive and academic factors such as SAT scores and critical thinking ability were weak indicators in comparison to the other predictors of success. Student effort and comfort level were found to be the strongest predictors of success. His study showed no correlation between stage of

cognitive development (abstraction ability) and final grade in CS1 (programming ability) [107, p. 122].

2.4 Kramer's Model and Question Patterns for Measuring Abstraction

This work shows that while Kramer's two-aspect model of abstraction is open to critique, it fits well with traditional conceptualizations, is particularly suited to the object oriented paradigm and aptly describes the application of abstraction in the field. Further, it is rooted firmly in Jean Piaget's description of cognitive development. Because Kramer's model appeared sound and eminently pertinent, it was adopted for this investigation. Besides being theoretically acceptable, Kramer's model underlies the empirical work by Hazzan and himself that yielded the "question patterns" (their term) for teaching and assessing abstraction. These helped to inform this research instrument and lend support to its validity.

Formal abstraction equates to "removing detail to simplify and focus attention." It is the term coined by the author for the scale comprised of problem-solving items requiring structure to be seen beneath irrelevancies. Descriptive abstraction equates to "generalization to identify the common core or essence." It is the term coined by the author for the scale comprised of items requiring interpretation and explication of most meaningful qualities.

As different as formal abstraction sounds from descriptive abstraction when they are defined, they are not mutually exclusive. Detail may be removed in order to generalize, or generalization may result from structural simplification. To restore a sense of their conceptual identity when they start to blur, Kramer

offers two graphic illustrations. Removing detail to simplify is illustrated by Harry Beck's map of the London Underground (the lower plate), but the geographic clarification in no way results in generalization. The map could not be used for getting around in the subways of New York City, Washington D.C., or Montreal.

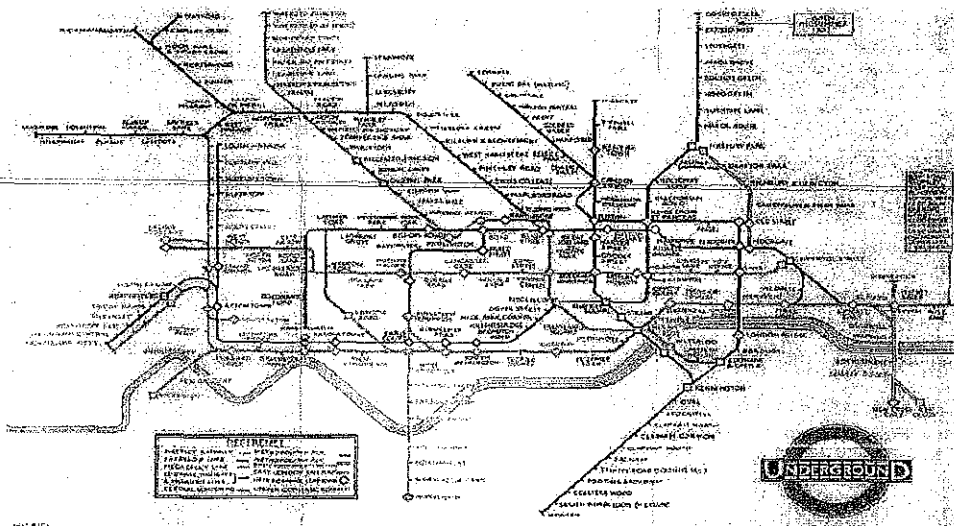
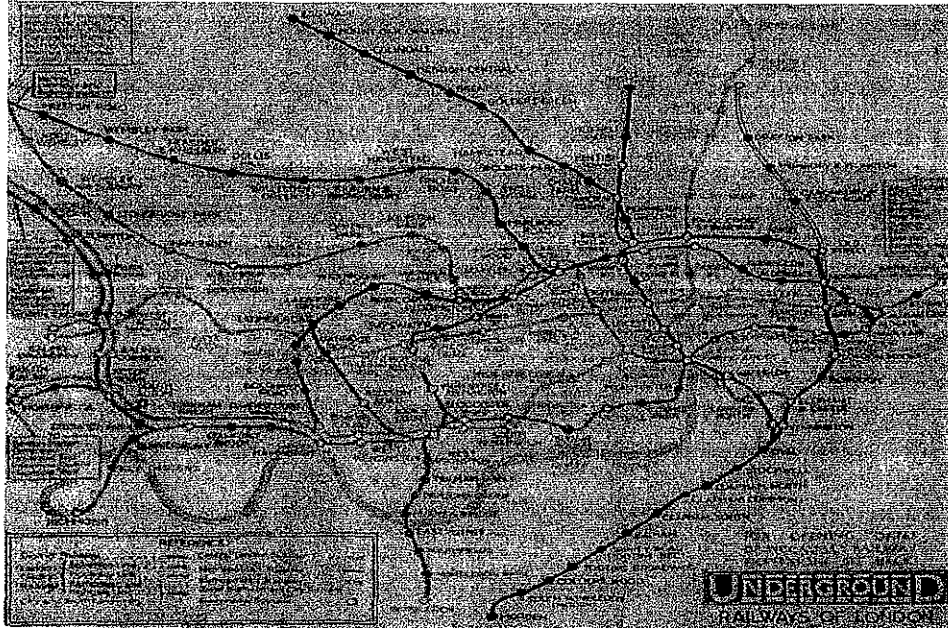


Figure 2.2. Comparison of London Underground Maps

A generalization that conveys a common essence is "Naked Blue IV" by Henri Matisse, shown below. By representing no woman in particular, it represents all women.



Figure 2.2 "Naked Blue IV" by Henri Matisse

The question patterns for abstraction offered by Hazzan and Kramer were created to be guides for teaching and evaluating abstraction throughout the computer science curriculum. Pattern 1, for example, is the following:'

Given two representations of a specific system, students are asked to explain which representation is more abstract and why. These representations could be program code, a UML diagram, a written description, a photo, and so on [46, p.5].

Pattern 4 offers another example:

Description of a specific system, with which the students are familiar, at different levels of abstraction. For example, describe your cell phone on at least four levels of abstraction [46, p. 6].

The examples make plain both their value and their limitation, at least as far as the present study is concerned. Their value lies in their content; the incorporated ideas make an excellent basis for developing questions. Their limitation lies in the fact that they are not readily adaptable to short-answer format or usable in disciplines outside computer science.

2.5 Conclusion

This chapter has examined the foundation work that has been done to describe abstraction as an important cognitive skill and an important aspect of human intellect that is manifested in the ability to understand, sort and manipulate information. It has shown that the ability to master complexity by creating and working with abstraction is central to computer science and the design of software. Many believe that this entails a thought process specific to the realm of computing, different even from the thinking and quantitative ability usually associated with mathematics.

Abstraction, as a human ability, is comprised of two complementary aspects, clearing away details to build simplifications and deriving generalizations that illuminate essentials. Agreement exists that this ability may be nurtured through instruction and experience, but that it rests upon a natural aptitude that is possessed by few. Agreement exists that this natural aptitude is

assessable, although no instrument yet exists for measuring it efficiently among prospective computer science majors who have not begun computer science coursework.

CHAPTER III

RESEARCH INSTRUMENT DEVELOPMENT AND DEPLOYMENT

3.1 Introduction

The main focus of this study was to determine if computer science majors have stronger abstraction skills than other majors. The goal was to develop a research instrument that could measure the relative abstraction skills of undergraduate students across disciplines to test hypothesis 1 from chapter 1: students who have declared computer science as their major will score higher than students in other majors on measurements of all three scales of abstraction.

There were challenges in developing such a test. The discussion in the computer science discipline about testing individuals for abstraction skill is an ongoing one and a standard for testing abstraction has not been agreed upon by the discipline. According to Kramer, "Unfortunately, we have been unable to find any existing appropriate tests. Tests for the formal operations stage focus mainly on logical reasoning and are not appropriate for testing abstraction skills nor distinguishing between the skills of students at college level [58, p. 41]."

Research indicates that abstraction is not a single, monolithic concept, but rather a constellation of different, yet somewhat similar mental processes. In order to deal with this complexity, a means was sought to test the three elements of abstraction identified in this chapter: conceptual abstraction, formal abstraction and descriptive abstraction. By separating the overall concept of abstraction into

these three cognitive areas, it becomes possible to develop measurements around the specific skill sets that are attributed to them. When the researcher says “testing abstraction,” the reference is to measuring aptitude for constructing abstractions.

The survey developed and implemented for this study was designed to test abilities in the three scales of abstraction (conceptual, formal and descriptive) among undergraduate students. Within that context, this chapter provides a detailed description of the study including the setting where the study took place, the population for the study, the theoretical underpinnings of the instrument, testing the scales of abstraction, development of the survey questions, data collection procedures, steps in instrument development, the statistical analysis of the study and additional questions in the study.

3.2 The Setting

The University where the study took place is a comprehensive, independent institution in New York with campuses in New York City and Westchester County with approximately 13,463 students. The gender mix at Pace is 39 percent male and 61 percent female. The University is divided by discipline into six schools: The Dyson College of Arts and Sciences, The Lubin School of Business, The School of Education, The Lienhard School of Nursing, The Seidenberg School of Computer Science and Information Systems and the School of Law. Each school, with the exception of the School of Law, offers

undergraduate programs. The institution awards associate, bachelors, masters, and doctoral degrees.

3.3 The Population for the Study

The survey for this study was made available to 560 undergraduate Pace University students in 32 class sections. These class sections can be formulated into three groupings: 1) a university core course in computing (CIS 101 - 210 students) taken by all students except computer science majors; 2) a computing course (CIS 102 - 292 students) with a prerequisite of CIS 101, that supports the university core but is not required; and 3) a required computing course (CS 121, CS 312, CS 389 - 48 students) or an upper level computing elective (CS 331 - 10 students) for computer science majors.

CIS 101, Introduction to Computing, the University core course in computing basics which is required of all Pace University undergraduate students (with the exception of computer science majors) focuses on building computing competency in three areas comprising data (Excel), Web development (HTML) and basic programming concepts (JavaScript). It is composed largely of freshmen and sophomores pursuing many different majors.

The CIS 102 courses included CIS 102Q - Problem Solving with LEGO Robotics, CIS 102 W - Web Design for Non-Profits, CIS 102T - Intergenerational Computing and one Honors course, CIS 396H - Problem Solving Using Technology and Human Endeavor in Urban Education. All of the CIS 102 courses focus on service learning. In these service learning courses, students

learn a series of computing-focused tasks at the beginning of the course and are then responsible for going out into the community to teach those skills to clients including school children or seniors, or to apply them in non-profit organizations. The student population ranged from sophomore through senior and students were drawn from many majors at the University, including computer science.

The third grouping is a selection of computer science courses from the undergraduate course cycle including CS 121 - Introduction to Programming, CS 312 - Research Methods in Computers and Society, CS 331 - Security in Computing and CS 389 - Software Engineering. The computer science courses were selected in order to provide a cross section of respondents who span the Pace University computer science majors.

A total of 227 surveys were returned from a total potential population of 560. Of those, nine were insufficiently complete for inclusion in the study and three were duplicates. This left a pool of 215 useable surveys returned from a population of 560 eligible students representing a return rate of 38.92 percent.

3.4 Theoretical Underpinnings of the Instrument

The instrument that was used (appendix A) was developed, in part, on the studies done by Kramer and Hazzan [46, 58] and to some extent, Bennedsen and Caspersen [12] and their work to develop an appropriate test of student abstraction skills.

In Hazzan and Kramer's 2006 work [46], they suggest a series of question patterns as a means of teaching abstraction in the computer science curriculum. They offer ten questions and recommend that an instructor adapt them to the context of the course that they are teaching. These questions were developed to present problem solving scenarios that require respondents to apply subject-specific abstraction skills in the manipulation and ordering of information. For example, the questions ask students to choose between two or more representations of a system and they ask them to choose the model that is more abstract, and then explain why it is more abstract. These questions are an excellent basis from which to develop questions to measure abstraction skills, particularly for the scale of descriptive abstraction. These questions were designed by the authors to be generic, adaptable and open-ended. However, they suggest caution in deploying these questions: "It is important to mention that such open questions require careful use so as to avoid a situation in which students give standard answers. We would not want our students, for example, to simply repeat considerations for using abstraction that they have heard in class [46, p.7]." These proposed questions were not 'short answer' and were not general in the sense of being usable for students in disciplines unrelated to computer science. As a result, these questions could not be used to test an interdisciplinary population of students for their potential as computer science students. In this light, it was incumbent to adapt these questions so that they would be suitable for a general audience (not just computer science majors) and that they be rendered in a format that is familiar to undergraduate students. In this context, it was deemed necessary to develop the survey questions in a multiple choice format

and this, as with all of the survey questions for each of the three abstraction scales, proved challenging. A set of questions were developed to test and measure abstraction levels in a cross section of respondents from all academic majors where they had to analyze and prioritize information and identify those elements that were most important within the given context.

3.4.1 Testing the Three Scales of Abstraction

The study looked at the tests run by Bennedsen and Caspersen [12] as well as the work done by Ventura [106, 107] to develop an assessment of abstraction abilities in the following core areas: conceptual abstraction, formal abstraction and descriptive abstraction. Conceptual abstraction refers to the individual's orientation in terms of being a generalist or a specialist. Formal and descriptive abstraction are different facets of a more general intellectual capability. While someone may score considerably better on one scale than the other, those of higher abstraction ability tend to score better on both than those of lower abstraction ability. Most descriptions and exemplifications of abstraction capture a bit of both "simplifying by removing detail" and "capturing the descriptive or structural essence in order to simplify [81, p.7]." Formal and descriptive abstraction offer a parsimonious means for conceptualizing abilities even though other descriptions of abstraction may tend to blur the distinction. The scale of formal abstraction identified for this study required the development of an abstraction either as an act of creativity or an act of deduction. The scale of descriptive abstraction identified for this study required that a familiar object,

like a cell phone, be described in the most elemental, abstract way. Each item presented a situation that required a simplifying determination to be drawn.

3.4.2 Developing the Questions

The challenge, in lieu of an established and accepted means of testing abstraction, was to develop a means of testing abstraction skills among undergraduate students that would render feedback on their ability to abstract information in both a problem solving context as well as an information manipulation one.

The initial instrument was developed after a careful review of the abstraction literature. After review, 23 questions were identified for the final instrument. Prior to dissemination of the survey, an independent professional designer of psychological measurements and surveys endorsed the survey's construct validity (appendix B). Construct validity pertains to a theoretical basis for a measurement's effectiveness, such as how the expansion of liquids offers the construct validity for the thermometer.

There are four sections on the survey including: background information, conceptual abstraction, formal abstraction and descriptive abstraction. The sections of the survey were organized in the following way: the background questions were designed to gather information regarding demographic characteristics of the student respondents. These included eight questions focusing, for each course taken, on which course they were enrolled in, current major, whether students had changed majors, if they had changed majors, what

their previous major(s) had been, gender, age, year in school, and program at Pace (i.e. Honors, CAP, Traditional).

The questions related to conceptual abstraction measure a self-reported inclination toward detail removal, simplification, and generalization. The three questions were designed to assess the respondents' self-reported orientation on a continuum ranging from the specific, on one end, to the general, on the other. Students were asked to respond to a five-point Likert scale (strongly disagree, disagree, neither agree nor disagree, somewhat agree, strongly agree) to indicate a relative place on the scale.

One question was designed to measure the respondents' interest in a variety of activities ranging from performing in public, to reading, to playing computer games. This question was included to gain some insight into a respondent's orientation in terms of actions, activities and interests. Some of these interests and activities have been anecdotally associated with students in certain majors (i.e., political science students like to read periodicals).

Six questions were developed to test a student's formal abstraction ability. These questions were problem solving questions that were oriented in logic and process. Several were slightly quantitative in nature. These questions challenged a student's ability to work with more intricate, quantitative issues.

Three questions were developed that asked students to evaluate an information object in order to identify a respondent's ability to focus on and feature the most important details in the object while ignoring or minimizing the

less important ones. These questions, therefore, were designed to test the respondent's descriptive abstraction abilities.

The survey instrument was developed, reviewed and then administered on paper to a section of CIS 102Q, Problem Solving with Lego Robotics. Twelve students took the survey and then were asked to discuss their reactions to the survey. Adjustments were made to the survey based on their feedback. The survey was then put into its initial electronic format and tested in a second service learning course, CIS 102W, Web Design for Non-Profits. The students were, again, debriefed on their reactions to the survey instrument. It was during this second pilot test that the phenomenon of 'abstraction anxiety' was observed, as reported by Ferguson, building on the Mathematics Anxiety Rating Scale developed by Richardson and Suinn [38], where students with mathematics phobias 'give up' when they, for example, encounter a problem where x's and y's replace 2's and 3's. The fact that certain questions, particularly those used to test formal abstraction, contained quantitative elements was troublesome to certain students and resulted in them dropping questions by not actually working through them to the level necessary to arrive at the correct answer.

3.5 Data Collection

The instrument was administered as a Web-based survey. There is a solid history of using Web-based surveys to collect data on students dispersed across a geographic area [48]. A link to the survey was e-mailed to each student

in more than 32 course sections offered by the Seidenberg School of Computer Science and Information Systems.

The survey was administered over 10 days beginning in the 12th week of a 14 week semester. This time was chosen because even the newest of students would have had a nearly complete semester's worth of experience in doing college level work. Logistically it was advantageous because it fell between final projects and final exams in many classes and would permit classroom instructors to give their students time to take the survey during class. Steps were taken to maximize the number of completed surveys. Email messages were sent to students and faculty in targeted courses were contacted directly and asked if they would make time in their courses for students to take the survey. In addition, there was an incentive for students to take the survey because they were told in the email that respondents completing the survey would be entered into a contest where 4 respondents selected at random would each win a \$25 American Express gift card.

3.6 Steps in Instrument Development

Table 1, below, shows the sequence of steps taken to develop the survey questions, test the survey instrument and then deploy the survey to the test population.

Table 1	
Steps in Instrument Development, Use and Deployment	
1.	Review the literature on abstraction

2. Formulate problem and plan the study
3. Develop the questions
4. Beta test the survey questions with test group on paper
5. Debrief survey respondents
6. Analyze results – determine validity of instrument
7. Incorporate changes in the survey instrument
8. Build the survey in the electronic surveying tool
9. Administer pilot run of survey in electronic format
10. Debrief Survey Respondents
11. Solicit input from the outside expert, Bob Birrer of Harris Interactive
12. Analyze results – determine validity of the instrument
13. Administer survey instrument to designated population
14. Periodically check and review data during the collection process
15. Close survey instrument
16. Analyze output data
17. Document and discuss results

3.7 Statistical Analysis

The data from the survey responses was collected using the Survey Monkey data collection tool. Collected data was disaggregated and run through a series of analytical tests to identify the relative abstraction skills of the respondents in the areas of conceptual abstraction, formal abstraction and descriptive abstraction.

Data collected by the survey instrument was imported into the Statistical Programming Software System (SPSS), version 15, a widely accepted and recognized software tool used for social science data analysis. The data studies included here were all rendered using the SPSS data analysis tool. Additionally, the SPSS tool provided a statistical means to compare data among respondents based on major and discipline.

The data was analyzed by major as well as by aggregation into related disciplines. The first tests run by the researcher were Chi-square analyses of the results of the scoring (discussed in detail in chapter IV) by major and discipline. According to Garson, "This statistic (Chi-square) is used to test the hypothesis of no association of columns and rows in tabular data. It can be used even with nominal data. A Chi-square probability of .05 or less is commonly interpreted by social scientists as justification for rejecting the null hypothesis that the row variable is unrelated (that is, only randomly related) to the column variable [43]."

Cross tabulation testing provided insights into comparisons and the inherent relationships between the scores on the three scales of abstraction. The results of Mean and Mode testing were also reviewed. Means testing establishes the adequacy of the sample size. Mode testing uncovers the highest frequency in a specific data set. Frequency testing helps to set the mean and variable for a statistical set. Cross tabulation allows for the joint distribution of two variables that provide a comparison between the two data points. According to the text *Basic Statistics* (2003): "Cross tabulation is a combination of two (or more) frequency tables arranged such that each cell in the resulting table

represents a unique combination of specific values of cross tabulated variables. Thus, cross tabulation allows us to examine frequencies of observations that belong to specific categories on more than one variable. By examining these frequencies, we can identify relations between cross tabulated variables [99]. “

3.8 Additional Questions Addressed in the Study

Questions for the survey were developed seeking answers to the hypotheses listed in chapter one. Answers to the following secondary questions are potentially interesting and had the potential to provide additional findings for the study.

Question 1: In examining academic disciplines, are there patterns evident in the abstraction scale scores?

Question 2: Is a student's academic program (Honors, Traditional, CAP) reflected in their abstraction skill level?

Question 3: Is abstraction aptitude related to gender?

3.9 Summary

Based upon the need discovered in the literature for a means to test abstraction abilities in students, a survey instrument was developed and implemented that would test three levels of abstraction among a cross section of undergraduate students at Pace University in New York. The literature concurs that abstraction ability is a key factor in success in the study and application of

computer science. This study defines three levels of abstraction; conceptual abstraction, formal abstraction and descriptive abstraction.

A survey was developed based on research which informed the questions which were developed, tested, refined and administered to a cross section of students at Pace University in New York, a doctoral-degree granting institution with a highly heterogeneous student body. The survey was administered over a ten day period in late spring, 2007, after which the data were imported into the SPSS data analysis tool in order to run analytical models. While an introductory discussion of these analysis techniques and models is included in this chapter, a detailed analysis of these results occurs in the next chapter, chapter IV.

Chapter IV

ANALYSIS OF THE DATA

4.1 Introduction

This chapter discusses the data from the survey instrument which was prepared for dissemination to specifically targeted populations at Pace University in New York. The chapter begins with a discussion of the results of the questions that comprised the survey instrument including a comparison of the demographic variables including age, gender, year in school and course enrolled. This is followed by an analysis of the responses that were given to the question groupings which were designed to test abstraction in the target population including the statistical models introduced in the prior chapter, which were used to identify patterns in the responses and to analyze the results. Analysis and comparison statistics were obtained using SPSS, version 15.

4.2 Comparison of Population with Respondents

A total of 227 surveys were returned from a total population of 560. Of those, nine were insufficiently complete for inclusion in the study and three were duplicates. This left a pool of 215 useable surveys returned from a population of 560 eligible students representing a return rate of 38.39 percent.

29.2 percent of the respondents had changed their major. This closely mirrors the US trend in changing majors [97].

The gender breakdown closely mirrored the Pace University Gender breakdown noted in chapter 1.

Table 1
Respondents by Gender

Gender Breakdown of Respondents by Percentage		Pace University
Women	60.8%	61%
Men	39.2%	39%

The Year in School breakdown was spread pretty evenly across the four levels of undergraduate students as shown in Table 2.

Table 2
Year in School

Year in School Breakdown by Respondent	
Freshman	26.7%
Sophomore	25.2%
Junior	19.9%
Senior	28.2%

A potentially important factor was the type of program in which the students were enrolled. Pace undergraduates fall into three program types: honors, for high level academic performers, traditional for the average range of students and the CAP (Challenge to Achieve) program for freshmen in need of stronger academic grounding in their first year. This breakdown reflects a higher percentage of honors students than is found in the Pace population. The survey results therefore come from a gender and year-in-school population that is very nearly equivalent to the Pace University population as a whole.

Table 3
Percentage Breakdown of Program Affiliation of Respondents

Program	Percentage	*Pace University (2006 – 2007)
Honors	24.4%	12.4%
Traditional	67.9%	76.4%
CAP	7.6%	11.2%

*Unofficial. Compiled from University sources.

4.3 The Survey Instrument and Survey Questions

The survey was constructed to have identifying questions including the course the student was enrolled in when asked to take the survey, current major, prior major if applicable and program affiliation. As an online survey taken on a computer, the survey questions were designed to be effective in the electronic format.

4.3.1 Section One – Identifiers – 4 questions.

Question 1 – What course is this?

Rationale: To determine in which course the student received the invitation to take the survey instrument.

Results: As shown in Table 4, a roughly equal, three part division between the three types of courses was achieved.

Table 4
Course Sections

<i>Course sections responding to the survey instrument</i>	
CIS 101 – Introduction to Computing	37.7%
CIS 102 – (Various) Service Learning	28.9%
CS - Computer Science Courses	33.3%

Question 2 – Your Major?

Rationale: To populate the major field in the database

Results: More than 31 individual majors from throughout the University's five undergraduate Schools were identified among the respondents

The breakdown of majors taking the survey spanned the gamut of undergraduate degree disciplines offered at the University and is reflective of the professional nature of Pace University's programs. Table 5 illustrates the distribution of majors among the respondents.

Table 5
Majors of Respondents

Major Pace University Undergraduate Major's Taking the Abstraction Survey Instrument	
Accounting	23
Art History	1
Biology	8
Business	10
Communications	9
Computer Science	30
Criminal Justice	1
Economics	2
Education	4
English	1
Environmental Science	1
Finance	9
Forensic Science	2
Hospitality management	1
Human Services	2
Information Systems	29
Management	7
Marketing	13
Mathematics	2
Musical Theater	1
Nursing	9
Physicians Assistant	4
Physical Therapy	1
Political Science	1
Professional Computer Studies	5
Psychology	4
Sociology	1
Technology Systems	10
Telecommunications	4
Theater	1
Undecided	4
Did Not Report	22
Total	210

These students were also joined by a small number of students holding a double major. These included:

Table 6
Listing of Double Majors

Pace University Undergraduate Student's Taking the Abstraction Survey Instrument Who Are Pursuing A Double Major	
Business/Psychology	1
Forensic Science/Chemistry	1
Philosophy/Environmental Studies	1
International Management/Finance	1
Political Science/International Studies	1

The surveys of the students indicating a double major were used based on the first undergraduate major identified.

Question 3 Have you ever changed your major?
and

Question 4 If so, what was your previous major(s)?

Rationale: To provide additional background information on students who may have developed abstraction skills in another major prior to acquiring their new major. Initially the study had intended to look at the results of students who had transferred into, and out of, a computer science major.

Results: 29.2 percent of the respondents had changed their majors during their undergraduate college careers. This corresponds to estimates by Michelle Releya of Long Island University in New York that approximately 33 percent of college students in the United States change their majors at least once [96, p.1].

4.3.2 Section Two – Conceptual Abstraction – four questions

The following section illustrates the responses to questions designed to measure an individual's self-reported identity in the scale of conceptual abstraction.

Scored on a five point Likert scale, respondents could score from 0 to 5.

Question 5 – I like to simplify and explain difficult questions to help others understand (Likert Scale 5 point answers). When combined, 73.4 percent of the respondents indicated a predilection toward conceptual abstraction.

Rationale: To determine the respondent's self-assessment in terms of abstraction and problem solving.

Results: As shown in Table 7, 34.1% of the respondents "strongly agreed".

However, "somewhat agreed" received the most answers at 39.4 percent.

Combined, 73.5% of respondents indicated a strong orientation toward conceptual abstraction.

Table 7
Responses to Question 5, "I like to simplify and explain difficult questions to help others understand."

	Response Percent
Strongly disagree	5.8%
Somewhat disagree	6.7%
Neither agree nor disagree	13.9%
Somewhat agree	39.4%
Strongly agree	34.1%

Question 6 – Specifics interest me more than generalizing.

Rationale: To gain an indication of the respondent's inclinations in terms of 'big picture vs. specifics' orientation.

Results: As shown in Table 8, 25% percent of the respondents "strongly agreed". "Somewhat agreed" was the most answered. When combined, 73.1 percent of the respondents indicated a predilection toward conceptual abstraction.

*Note: The wording of this item is reversed so that disagreement is indicative of higher abstraction than agreement.

Table 8
Responses to Question 6, "Specifics interest me more than generalizing." (scores transposed)

	Response Percent
Strongly disagree	6.3%
Somewhat disagree	6.7%
Neither agree or disagree	13.9%
Somewhat agree	47.6%
Strongly agree	25.5%

Question 7 – I tend to remember things by ideas rather than by details.

Rationale: To gather insight into each respondents orientation in identifying ideas as opposed to specific details. When combined, 65.8 percent of the respondents reported a predilection toward conceptual abstraction.

Results: As shown in Table 9, 29.3 percent of respondents "strongly agreed." "Somewhat agreed" was the most answered at 36.5%.

Table 9
Responses to Question 7, "I tend to remember things by ideas rather than details."

	Response Percent
Strongly disagree	6.3%
Somewhat disagree	12.5%
Neither agree nor disagree	15.4%
Somewhat agree	36.5%
Strongly agree	29.3%

Review of Results for Conceptual Abstraction Section, Questions 5 – 7

The Likert scale scores for the three conceptual abstraction questions reflect a self-reported 'concept orientation' toward basic attributes and traits associated with conceptual abstraction. These results may be indicative of a self-awareness of abstraction ability that is deserving of further exploration.

4.3.3 Section Three – Activities

A question was added to provide additional insight into the interests and activities of the respondents. Many of these have anecdotal associations with certain disciplines and majors.

Question 8 – Some activities that I enjoy include:

Rationale: To gain further insight into respondent's personal predilections in a series of activities including problem solving, art, gaming, etc.

Results: Further details were obtained to provide insight into the interests and habits of the respondents. Over half of the respondents enjoy problem solving (62.7%), using the computer (77.5%), reading magazines (67.5%) and playing computer games (50.2%). These results are indicative of activities that have been anecdotally linked to students in certain disciplines, i.e., communications majors like to read magazines, computer science majors like to play computer games.

Table 10
Responses to Question 8, "Some activities I enjoy include:"

	Response Percent
Problem solving	62.7%
Writing essays	27.3%
Making art	43.1%
Speaking in class	29.2%
Working in teams	39.2%
Computer games	50.2%
Using the computer	77.5%
Reading magazines	67.5%
Playing a musical instrument or singing	34.0%

4.3.4 Section Four – Formal Abstraction – five questions

The questions in section four were designed to test the respondents in problem solving-type areas that are associated with formal abstraction. These five multiple choice questions had a single correct answer and respondents scored one point for a correct answer and zero points for a wrong answer.

Question 9 – You are in a race and you overtake the second person. What position are you in?

Rationale: To ascertain the respondent’s ability to visualize the process of a physical event and to conceptualize the processes and sequence of events.

Results: As shown in Table 11, 46.3 percent of respondents selected the correct answer, ‘b’. This was also answer with the most responses, 46.3%.

Table 11
Responses to Question 9, “You are in a race...”

	Response Percent
Correct Answer = b – Percentage correct = 46.3%	
a) first	34.3%
b) second	46.3%
c) last	1.0%
d) insufficient data	18.4%

Question 10 – Suppose that a man from Mars can survive for exactly two weeks without food or sleep. What should he do at the end of the 14th day without having slept or eaten? Martians, like earthlings, cannot eat and sleep at the same time, but meals for Martians take several hours.

Rationale: To determine the respondents ability to process detail and abstract a sequence of events.

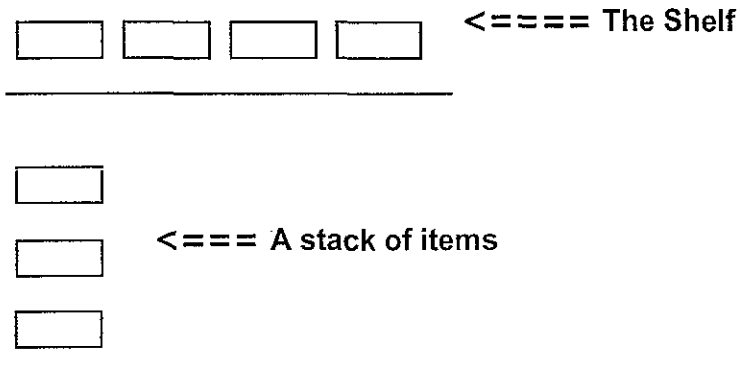
Results: As shown in Table 12, 22 percent of respondents selected the correct answer, “eat and sleep in the same order from two weeks ago.” The most answered was “eat first” with 34 percent of the responses.

Table 12
Responses to Question 10, “Suppose that a man on Mars...”

	Response Percent
Correct Answer = c – Percentage correct = 22.0%	
a) eat first	34.0%
b) sleep first	15.0%
c) eat and sleep in the same order from two weeks ago	22.0%
d) eat and sleep in the reverse order from two weeks ago	9.0%
e) insufficient data	20.0%

Question 11 – (includes visual) Suppose you have a robot that can remove items from a shelf, stack them up and replace them on the shelf. The robot, which can hold only one item at a time, is controlled by these instructions:

Table 13
Diagram from Question 11



1. Remove the item from the left-end of the shelf. This leaves the robot holding the item.
2. Remove the item from the top of the stack. This leaves the robot holding the item.
3. Place the item held by the robot onto the top of the stack.
4. Place the item held by the robot onto the right end of the line of items on the shelf.
5. Place the item held by the robot onto the left end of the line of items on the shelf

Four items are on the shelf. Give the sequence of instructions that get the robot to reverse their order. When the robot starts, the items look like this:



When the robot finishes, the items should look like this (although their exact placement on the shelf does not matter, only their order):



Rationale: As a question with a visual component, this question tests the respondent's ability to abstractly determine timing, placement and sequence of physical objects.

Results: As shown in Table 14, 45.1 percent of respondents correctly identified the right answer.

Table 14
Responses to Question 11, "Suppose you have a robot..."

	Response Percent
Correct Answer = b – Percentage correct = 45.1%	
a) 1 3 1 3 1 3 1 3 2 4 2 4 2 4 2 4	19.4%
b) 1 3 1 3 1 3 1 3 2 5 2 5 2 5 2 5	45.1%
c) 1 4 1 4 1 4 1 4	24.3%
d) 1 3 2 4 1 3 2 4 1 3 2 4 1 3 2 4	11.1%

Question 12 – (based on question 11) - There is more than one correct answer to the previous item, but all are similar in form, and all are verbose. What might be done to condense them?

Rationale: Tests the respondent's abstraction ability to condense information in a logical sequence.

Results: As shown in Table 15, 13.2 percent of respondents correctly identified the best answer.

Table 15
Responses to Question 12, "There is more than one correct answer..."

Correct Answer = f - Percentage Correct = 13.2%	Response Percent
a) New instructions could be added to the four that were given. For instance, a new instruction 6 could be "do instruction 1 then do 3."	23.6%
b) An instruction could be added that says "place the four items from the shelf onto the stack." It presumes nothing that the robot cannot already do.	17.4%
c) If it were not possible to extend the instruction set, we could at least adopt these expressive conventions among ourselves as a communication convenience.	18.1%
d) a and b	19.4%
e) a and c	8.3%
f) a, b, and c	13.2%

Question 13 – Did any of your elementary school teachers show you the trick for adding consecutive whole numbers between 1 and 10? First, you had to "fold the series of numbers in the middle and add them in pairs." $1+10$, $2+9$, $3+8$, $4+7$, $5+6$. Then you could see the shortcut: get the sum of the first pair, $1+10$, and multiply it by 5, which is the number of pairs. What can you say about this?

Rationale: To measure the respondent's ability to conceive of other logical answers to a word problem. This is a key feature of formal abstraction.

Results: As shown in Table 16, 20.2 percent of respondents selected the best answer.

Table 16
Responses to Question 13, "Did any of your elementary school teachers..."

Correct Answer = d - Percentage correct = 19.4%		Response Percent
a) It is an interesting anomaly.		32.6%
b) This technique can be applied to any consecutive sequence that starts at 1 and has an even number of entries.		24.3%
c) This technique can be applied to any consecutive sequence that starts at 1, whether		18.1%
d) This technique can be applied to any consecutive sequence regardless of where it starts and whether the number of entries is even or odd.		20.2%

Question 14 – (includes visual) In a Hi-Q game, a peg may jump over another peg into a vacant spot. When it does, the jumped peg is removed. The object of the puzzle is to discover a sequence of jumps that leaves only one peg left, in the center. The starting board is shown below, where each x represents a peg.

```

      x x x
      x x x
x x x x x x x
x x x   x x x
x x x x x x x
      x x x
      x x x

```

figure 4.1
graphic for Hi-Q question

Your friend gives you the solution. The first move is expressed as $5 \Rightarrow 17$.
But what does it mean?

Rationale: This question tests the respondent's ability to visualize a quantitative solution to a problem of time and space.

Results: As shown in Table 17, 18.1 percent of respondents selected the best answer.

Table 17
Responses to Question 14, "Hi-Q question"

	Response Percent
Correct Answer = d - Percentage correct = 18.1%	
a) It could mean so many things that, in and of itself, it defies usefulness.	28.3%
b) It means "jump the peg at point 5 into point 17." Point 17 has to be the blank space, and point 5 is the middle peg in the second row	37.7%
c) It means what is said by b, except that point 5 is actually the peg in the middle column	15.9%
d) It means what is said by b, except that point 5 could be any of four different points	18.1%

Review of Results for Formal Abstraction Section, Questions 9 – 14

Results for the six questions related to formal abstraction show mixed performance by the respondents. In no case did the majority of students get the correct answer; in only two of the six questions did respondents get the correct answer more often.

4.3.5 Section Five – Descriptive Abstraction

These questions were developed to test the respondent's skills in identifying the most important elements of a set of information (or an object). These questions were rendered in a multiple choice format with a single correct answer for each.

Question 15 – It is two-fifteen and the teacher asks the child to report the time.

She answers one seventy-five. What would you say about this?

Rationale: This question tests the respondent's ability to re-contextualize quantitative information.

Results: As shown in Table 18, 23.7 percent of respondents selected the best answer.

Table 18
Responses to Question 15, "It is two-fifteen..."

	Response Percent
Correct Answer = a - Percentage correct = 23.7%	
a) This shows an excellent understanding and was probably said in jest.	23.7%
b) The child needs some instruction in the conventions of reporting times of day.	25.2%
c) The child does not know how to tell time.	26.7%
d) The child should be tested on other times in order to make a fair assessment.	24.4%

Question 16 – What would be a good way of explaining what a Website is to someone who has never experienced the Internet?

Rationale: Replicating several of the questions posed by Hazzan (33) this question is designed to test the respondent's ability to develop a descriptive abstract for a physical object.

Results: As shown in Table 19, 50.8 percent of respondents selected the best answer.

Table 19
Responses to Question 16, "...good way of explaining a Web site..."

Correct Answer = b - Percentage correct = 50.8%

a) It is a domain-named location on a server, accessible with a browser.	27.3%
b) It is kind of like a telephone answering machine, but you "call it up" on a personal computer. What you get is a display on the screen.	50.8%
c) The Internet developed out of work funded by the Department of Defense in the late 1970s. Today, virtually all business and many individuals have Websites. You, personally, are better off with cable than dial-up.	18.9%
d) FrontPage is Microsoft's product for creating Webpages, but many people prefer Adobe's Dreamweaver. Knowledge of HTML (the hypertext markup language) is not necessary.	3.0%

Questions 17 - A description "from thirty thousand feet" is one that filters out details to focus on what is key. Which of the following would have made the most useful "thirty thousand feet" description of a cell phone in the old days, before people knew about them?

Rationale – This question tests the respondent's abilities to parse the key information about a physical object, and then select the best abstraction of that object.

Results: As shown in Table 20, 36.9 percent of respondents selected the best answer.

Table 20
Responses to Question 17, "...from thirty thousand feet..."

	Response Percent
Correct Answer = b - Percentage correct = 36.9%	
a) An electronic device used by individuals for communications that is light-weight, and portable. This is relatively new technology made possible by microprocessors.	14.6%
b) A telephone with an independent number that connects into the phone system by radiowaves instead of a telephone line. Calls can be made and received from anywhere.	36.9%
c) A highly multipurpose electronic device that may function as a telephone, a camera, an email reader, a music player, and even more (e.g. television). The features depend on the model, but the product ownership is exceedingly widespread.	13.6%
d) If you think of a walky-talky you have the idea: mobile communications. Service tends to be good with coverage, signal strength, and bandwidth constantly improving.	24.3%
e) Usage details vary from model to model, but first you power-up, then you enter the number, and then you press the send button. It works very much like a regular phone, and it is safe to say that no one has trouble learning to use one.	10.7%

Review of Results for Descriptive Abstraction Section, Questions 15 – 17

A review of the results for the three questions designed to measure descriptive abstraction both for the individual questions as well as the aggregate scores for this section as a whole, indicate that the respondents were more adept at the questions on descriptive abstraction. Questions 16 and 17, classic word problems, gave respondents the opportunity to test their ability to simplify in a

traditional multiple choice format. However, upon evaluation of the responses to question 15, it was decided to throw question 15 out as a 'bad' question.

4.3.6 Section Six – Individual Information

Question 18 – For confidential research purposes, may we look up your grades at the end of the semester?

Table 21
Answers to question 18

For confidential research purposes, may we look up your grades after the semester?	
	Response Percent
yes	49.2%
no	50.8%

Rationale: To obtain permission from the individual student to check their academic performance as an indicator of student success.

Results: Just under half of the respondents were willing to give authorization to have their academic performance in the classes. This was not an impediment to answering the questions for this survey.

Question 19 – If yes, what is your Pace student email?

We collected email addresses from 54.67 percent of the respondents

Rationale: To get identifying information to look up student grades.

Question 20 – Your age?

Ages ranged from 18 to 47 years with an average age of 21.84. The 47 year old, while a statistical outlier, is reflective of a population of returning adults in Pace University's undergraduate programs. Thus the ages of these respondents were factored into the average.

Question 21 - Your gender? (see Table 1)

Question 22 – Your year in school? (see Table 2)

Question 23 – Your program at Pace? (see Table 3)

Rationale: To get further indicators of the respondent's academic place at the University.

Results: Individual, identifying demographic information was gathered allowing for detailed evaluation of the responses, shown in Tables 1, 2 and 3.

4.4 Grouping of Majors

In order to develop statistical tests for the data gathered via the research instrument, the individual majors were grouped into 9 subsets by discipline. These groupings were based on traditional groupings of majors in the American higher education system and are consistent with the groupings used in studies by Ventura [107]. Individual majors were assigned to the discipline groups shown in Table 22.

Table 22
Grouping of Academic Majors by Discipline

	Percent of Whole
1. Computer Science	13.95%
2. Technology includes - Information Systems Technology Systems Professional Computer Studies Telecommunications	22.33%
3. Business Includes - Business Management Marketing	14.42%
4. Finance and Accounting	14.88%
5. Science and Math Includes - Biology Chemistry Forensic Science Mathematics Physicians Assistant Physical Therapy	8.88%
6. Social Science and Education Includes Criminal Justice Education Environmental Science Human Services Psychology Sociology	6.05%
7. Arts Includes - Communications Musical Theater Philosophy Political Science Theater	6.15%
8. Nursing	4.19%
9. Undecided	1.86%

The questions posed by the researcher looked to test whether students who have declared computer science as a major will score higher on all scales of abstraction than students who have declared other majors. Rigorous item analyses were performed in accordance with best practices. The item analysis, however, found that the responses to question 15 were so evenly distributed that the question and the scores were questionable from a quality standpoint. Therefore question 15 was eliminated as a 'bad' question.

4.5 Experimental Results in SPSS

This section reports on the data analysis that was performed on the results of the survey instrument in the SPSS statistical package. It was determined to test the results of the questions on conceptual abstraction, formal abstraction and descriptive abstraction based on the responses from the compiled discipline groups indicated in the previous section.

4.5.1 Comparing Discipline and Abstraction Abilities

The data was analyzed using cross-tabs statistics. Cross-tabs produces tables showing the joint distribution of two or more variables. In the first experiment, we cross tabulated questions focusing on conceptual abstraction with discipline. This exercise analyzed the results of the questions related to conceptual abstraction as rendered by the respondents within their nine assigned disciplinary groups in order to explore the effect of academic discipline on conceptual abstraction.

For scoring purposes on these Likert scale-answered questions, respondents could score from one to five points for each question depending on where their answer fell on the five point Likert scale from strongly disagree to strongly agree. Therefore, respondents scored anywhere from 3 to 15 which indicates the relative, self-reported perception of conceptual abstraction of the respondents by academic discipline. The results are shown in Table 23.

Table 23
Conceptual Abstraction and Discipline Cross-tabulation

		Conceptual											Total	
		3.00	4.00	6.00	7.00	8.00	9.00	10.00	11.00	12.00	13.00	14.00	15.00	
DisciplineArea	Computer Science	0	0	0	2	3	2	5	7	4	2	1	1	27
	Technology	0	0	0	2	2	8	9	13	10	1	2	1	48
	Business	0	0	3	2	0	7	7	7	5	0	0	0	31
	Finance & Accounting	0	0	1	2	6	2	10	6	5	0	0	0	32
	Science & Math	0	0	0	3	2	5	1	4	1	0	1	0	17
	Social Science & Education	0	0	1	3	3	2	3	4	0	0	0	0	16
	Arts & Humanities	0	0	0	0	1	3	6	4	0	1	0	0	16
	Nursing	0	0	1	0	1	5	1	1	0	0	0	0	9
	Undecided	2	1	0	0	0	0	0	1	0	0	0	0	4

The Chi-square test of association was performed and the result ($p \leq 0.00$) indicates that discipline is strongly associated with conceptual abstraction at the 0.05 level.

Table 24
Conceptual Abstraction and Discipline Cross-tabulation

Chi-Square Tests

	Value	df	Asymp. Sig. (2-sided)
Pearson Chi-Square	225.045 ^a	88	.000
Likelihood Ratio	111.415	88	.047
Linear-by-Linear Association	21.662	1	.000
N of Valid Cases	200		

a. 96 cells (88.9%) have expected count less than 5. The minimum expected count is .02.

In order to test the respondent's answers to questions based on formal abstraction, the responses to those questions were aggregated and cross tabulated by discipline. For scoring purposes, answers were scored on a basis of 0 through 1 (1 point for each correct answer, 0 points for incorrect answers) so formal scores range from 0 (no correct responses) to 6 (all correct responses) and are reflected on the following chart.

Table 25
Formal Abstraction and Discipline

		Formal						Total
		.00	1.00	2.00	3.00	4.00	5.00	
DisciplineArea	Computer Science	3	8	10	2	3	1	27
	Technology	16	17	11	2	1	0	47
	Business	8	8	7	4	1	0	28
	Finance & Accounting	5	9	10	3	0	0	27
	Science & Math	6	5	4	1	0	0	16
	Social Science & Education	4	4	4	3	0	0	15
	Arts & Humanities	2	3	7	4	0	0	16
	Nursing	2	6	0	1	0	0	9
	Undecided	2	0	2	0	0	0	4

No respondent answered all six questions correctly. However, 59 percent of computer science majors got three or more right, the highest of all disciplines. However, the result ($p = 0.347$) shown in the Chi-square analysis in Table 26 indicates that there is no association between discipline and formal abstraction.

Table 26
Formal Abstraction and Discipline

Chi-Square Tests

	Value	df	Asymp. Sig. (2-sided)
Pearson Chi-Square	42.922 ^a	40	.347
Likelihood Ratio	42.983	40	.345
Linear-by-Linear Association	.255	1	.614
N of Valid Cases	189		

a. 40 cells (74.1%) have expected count less than 5. The minimum expected count is .02.

The next analysis involved questions 16 and 17, focusing on descriptive abstraction. Discipline was cross tabulated against the results of these questions. Questions 16 and 17 were combined. For scoring purposes, answers were scored on a basis of 0 through 1 (1 point for each correct answer, 0 points for incorrect answers) so formal scores range from 0 (no correct responses) to 2 (both correct responses) and are reflected on the following chart.

Table 27
Descriptive Abstraction and Discipline

DisciplineArea	Descriptive			Total
	.00	1.00	2.00	
Computer Science	5	7	6	18
Technology	2	14	8	24
Business	12	4	3	19
Finance & Accounting	9	11	2	22
Science & Math	5	5	1	11
Social Science & Education	3	6	1	10
Arts & Humanities	6	8	0	14
Nursing	4	1	1	6
Undecided	2	1	0	3

Based on the results illustrated in Table 27, 33 percent of computer science majors answered both questions correctly. This compares with 12.5% of all other majors who answered both questions correctly.

The chi-squared results ($p = 0.029$) indicate that there is a significant relationship (confirming Ventura's experience) between discipline and descriptive abstraction.

Table 28
Discipline and Descriptive Abstraction

Chi-Square Tests

	Value	df	Asymp. Sig. (2-sided)
Pearson Chi-Square	28.255 ^a	16	.029
Likelihood Ratio	32.676	16	.008
Linear-by-Linear Association	9.730	1	.002
N of Valid Cases	127		

a. 17 cells (63.0%) have expected count less than 5. The minimum expected count is .52.

4.5.2 Comparing Abstraction Abilities

The next comparison explored the results of the conceptual abstraction questions with the results of the questions on formal abstraction. This test was run to determine what relationship, if any, exists between the scales of formal and conceptual abstraction. Table 29 shows how the 5 point possible scoring for formal abstraction cross-tabulates with the 15 point possible scoring for conceptual abstraction. The cross-tabulation indicates that there is a strong relationship between formal and conceptual abstraction.

Table 29
Conceptual Abstraction and Formal Abstraction

		Conceptual											Total	
		3.00	4.00	6.00	7.00	8.00	9.00	10.00	11.00	12.00	13.00	14.00	15.00	3.00
Formal	.00	2	0	1	6	6	7	7	13	5	1	0	0	48
	1.00	0	0	2	2	5	10	15	13	7	1	3	1	60
	2.00	0	1	2	3	4	9	11	16	7	1	1	0	55
	3.00	0	0	0	3	1	6	5	2	3	0	0	0	20
	4.00	0	0	0	0	0	1	0	1	2	0	0	1	5
	5.00	0	0	0	0	0	0	0	0	0	1	0	0	1
Total		2	1	5	14	16	33	38	45	24	4	4	2	189

The chi-square analysis below indicates that conceptual orientation scales is significantly related to formal abstraction ($p = 0.001$).

Table 30
Conceptual Abstraction and Formal Abstraction

Chi-Square Tests

	Value	df	Asymp. Sig. (2-sided)
Pearson Chi-Square	96.005 ^a	55	.001
Likelihood Ratio	48.001	55	.737
Linear-by-Linear Association	3.190	1	.074
N of Valid Cases	189		

a. 59 cells (81.9%) have expected count less than 5. The minimum expected count is .01.

Progressing through the results of the survey instrument, the conceptual abstraction data were cross tabulated with the data from descriptive abstraction. The results in Table 31 indicate that there is a strong relationship between conceptual and descriptive abstraction.

Table 31
Conceptual Abstraction and Descriptive Abstraction

		Descriptive			Total
		.00	1.00	2.00	
Conceptual	3.00	1	0	0	1
	4.00	0	1	0	1
	6.00	2	0	0	2
	7.00	3	6	0	9
	8.00	4	5	1	10
	9.00	9	10	3	22
	10.00	13	8	3	24
	11.00	10	19	3	32
	12.00	4	4	8	16
	13.00	0	3	1	4
	14.00	0	1	2	3
	15.00	1	0	1	2
	21.00	1	0	0	1

The results of the Chi-square analysis indicate that there is a statistically significant relationship between conceptual abstraction and descriptive abstraction ($p = 0.025$).

Table 32
Conceptual Abstraction and Descriptive Abstraction

Chi-Square Tests

	Value	df	Asymp. Sig. (2-sided)
Pearson Chi-Square	36.792 ^a	22	.025
Likelihood Ratio	37.846	22	.019
Linear-by-Linear Association	9.478	1	.002
N of Valid Cases	127		

a. 27 cells (75.0%) have expected count less than 5. The minimum expected count is .17.

Table 33 indicates the results of the cross-tabulation of the scores from the tests of formal and descriptive abstraction.

Table 33
Formal Abstraction and Descriptive Abstraction

		Descriptive			Total
		.00	1.00	2.00	
Formal	.00	8	12	1	21
	1.00	17	13	7	37
	2.00	18	20	8	46
	3.00	5	10	3	18
	4.00	0	2	3	5
	5.00	0	1	0	1

The chi-square analysis ($p = 0.195$) indicates that there is no relationship between formal abstraction and descriptive abstraction is weak.

Table 34
Formal Abstraction and Descriptive Abstraction

Chi-Square Tests

	Value	df	Asymp. Sig. (2-sided)
Pearson Chi-Square	13.538 ^a	10	.195
Likelihood Ratio	14.432	10	.154
Linear-by-Linear Association	4.357	1	.037
N of Valid Cases	128		

a. 8 cells (44.4%) have expected count less than 5. The minimum expected count is .17.

4.5.3 Comparing the Effect of Gender on Abstraction

The study progressed to examine the effect of gender on abstraction ability. In comparing the impact of gender on conceptual abstraction, the distribution of scores shows a strong relationship between them.

Table 35
Conceptual Abstraction and Gender

		Gender			Total
		Male	Female	Unreported	
Conceptual	3.00	1	0	0	1
	4.00	0	1	0	1
	6.00	1	1	0	2
	7.00	1	8	0	9
	8.00	6	3	0	9
	9.00	7	15	0	22
	10.00	6	18	0	24
	11.00	12	20	0	32
	12.00	9	7	0	16
	13.00	1	2	1	4
	14.00	1	2	0	3
	15.00	2	0	0	2

The Chi-square analysis ($p = 0.002$) indicates a strongly significant relationship between conceptual abstraction and gender.

Table 36
Conceptual Abstraction and Gender
Chi-Square Tests

	Value	df	Asymp. Sig. (2-sided)
Pearson Chi-Square	46.561(a)	22	.002
Likelihood Ratio	24.789	22	.307
Linear-by-Linear Association	.441	1	.507
N of Valid Cases	125		

a. 26 cells (72.2%) have expected count less than 5. The minimum expected count is .01.

The analysis continued to examine the effect of gender on formal abstraction. In comparing the impact of gender on formal abstraction, the distribution of scores shows them, again, to be unrelated.

Table 37
Formal Abstraction and Gender

		Gender			Total
		Male	Female	Unreported	
Formal	.00	8	10	1	20
	1.00	14	23	0	37
	2.00	17	28	0	45
	3.00	5	13	0	18
	4.00	2	3	0	5
	5.00	1	0	0	1

Table 38 shows the Chi-square analysis for formal abstraction and gender to have no relationship ($p = 0.561$).

Table 38
Formal Abstraction and Gender
Chi-Square Tests

	Value	df	Asymp. Sig. (2-sided)
Pearson Chi-Square	13.542(a)	15	.561
Likelihood Ratio	10.628	15	.779
Linear-by-Linear Association	2.137	1	.144
N of Valid Cases	126		

The analysis continued by looking at the impact of gender on the descriptive abstraction scale. The results indicate that gender is not related to descriptive abstraction ability.

Table 39
Descriptive Abstraction and Gender

		Gender			Total
		Male	Female	Unreported	
Descriptive	.00	16	31	0	47
	1.00	21	34	1	57
	2.00	10	12	0	22

The Chi-square analysis, likewise, confirms that descriptive abstraction and gender are clearly not related ($p = 0.770$).

Table 40
Descriptive Abstraction and Gender
Chi-Square Tests

	Value	df	Asymp. Sig. (2-sided)
Pearson Chi-Square	3.306(a)	6	.770
Likelihood Ratio	4.038	6	.672
Linear-by-Linear Association	.002	1	.964
N of Valid Cases	126		

a. 53 cells (88.3%) have expected count less than 5. The minimum expected count is .01.

4.5.4 Comparing the Effect of Program on Abstraction

The analysis in SPSS then continued by looking at the effect of program (Honors, traditional, CAP) on the three scales of abstraction. It could be estimated that an Honors student population might have stronger abstraction skills than other students. Table 41 examines the impact of program on conceptual abstraction and shows no relationship between the two.

Table 41
Conceptual Abstraction and Program

		Program				Total
		Honors	Traditional	CAP	Other	
Conceptual	3.00	0	1	0	0	1
	4.00	0	1	0	0	1
	6.00	0	1	0	0	1
	7.00	4	2	2	0	8
	8.00	1	5	1	1	8
	9.00	3	18	0	0	21
	10.00	6	16	1	0	23
	11.00	8	15	5	3	31
	12.00	7	5	1	0	13
	13.00	1	1	0	1	3
	14.00	1	2	0	0	3
	15.00	1	0	0	1	2

The Chi-square analysis indicates no relationship between program and conceptual abstraction ($p = 0.385$).

Table 42
Conceptual Abstraction and Program
Chi-Square Tests

	Value	df	Asymp. Sig. (2-sided)
Pearson Chi-Square	46.112(a)	44	.385
Likelihood Ratio	44.169	44	.464
Linear-by-Linear Association	.001	1	.971
N of Valid Cases	115		

a. 53 cells (88.3%) have expected count less than 5. The minimum expected count is .01.

Formal abstraction, likewise, is not impacted by program as shown in Table 43 where the table shows no relationship between the two.

Table 43
Formal Abstraction and Program

		Program				Total
		Honors	Traditional	CAP	Other	
Formal	.00	8	8	0	0	16
	1.00	10	19	4	1	34
	2.00	8	25	5	4	43
	3.00	5	11	1	0	17
	4.00	2	2	0	1	5
	5.00	0	1	0	0	1

The Chi-square analysis (Table 44) confirms that there is no relationship ($p = 0.733$) between formal abstraction ability and program.

Table 44
Formal Abstraction and Program
Chi-Square Tests

	Value	df	Asymp. Sig. (2-sided)
Pearson Chi-Square	15.736(a)	20	.733
Likelihood Ratio	18.440	20	.558
Linear-by-Linear Association	2.097	1	.148
N of Valid Cases	116		

a. 24 cells (80.0%) have expected count less than 5. The minimum expected count is .01.

The final analysis in SPSS compared the impact of program on descriptive abstraction ability. The clear result was that a respondent's program affiliation had no impact on formal abstraction skill.

Table 45
Descriptive Abstraction and Program

		Program					Total
		1	2	3	4	5	1
Descriptive	.00	7	32	4	1	0	44
	1.00	19	25	5	4	0	53
	2.00	7	9	1	1	1	19
Total		33	66	10	6	1	116

The Chi-square analysis indicates that formal abstraction and program are unrelated ($p = 0.089$).

Table 46
Descriptive Abstraction and Program
Chi-Square Tests

	Value	df	Asymp. Sig. (2-sided)
Pearson Chi-Square	13.744(a)	8	.089
Likelihood Ratio	12.678	8	.123
Linear-by-Linear Association	.079	1	.779
N of Valid Cases	116		

4.6 Summary

This chapter has reported on the results of the survey instrument developed to assess undergraduate students for three levels of abstraction ability: conceptual abstraction, formal abstraction and descriptive abstraction and compares these scores with the disciplines of the respondents. The results of the analytical tests run through SPSS are provided in Tables 1 – 34 and are discussed in this chapter. Cross-tabulation and Chi-squared tests between the discipline and abstraction scales showed the following predictive ability and relative statistical significance:

- Discipline and conceptual abstraction are significantly associated;
- Discipline and formal abstraction have no association;
- Discipline and descriptive abstraction are associated.

Cross-tabulation and Chi-squared tests between abstraction scales showed the following predictive ability and relative statistical significance:

- Conceptual and formal abstraction are strongly associated;
- Conceptual and descriptive abstraction are associated;
- Formal and descriptive abstraction are not associated.

Chapter V

FINDINGS, INTERPRETATIONS AND IMPLICATIONS OF THIS STUDY

5.1 Introduction

This chapter consists of the discussion of the findings of the survey instrument, the extrapolation of trends, patterns and key points from the data, the implications of the associations between the three scales of abstraction that were tested and an exploration of opportunities for future research. In addition, there are findings on the implications for the field of computer science on testing and measuring abstraction skill in individuals.

5.2 Review of the Study

The objective of this study was to explore abstraction as a measurable skill set in a broad population of undergraduate students with a goal of comparing the abstraction skills of computer science students against those of students in *other majors*.

A detailed exploration of the nature of abstraction in the computer science discipline was a necessary foundation for this work. This early stage research indicated that there was significant opportunity to extend the recent work by others in understanding and testing for abstraction skill in individuals. Chapter I described the nature of abstraction in the computer science discipline and defined abstraction in the literature. It then went on to define abstraction in the context of this study and described the three, specific scales of abstraction

central to this study: conceptual abstraction, formal abstraction and descriptive abstraction. It continued with an exploration of the research on abstraction, particularly the work that supported the hypothesis central to this study: that *computer science majors would score higher on a test of abstraction skills than students in other majors*. This, and the other hypotheses coming out of the research, guided this work. The conceptual framework for this study came from the foundation work of Kramer and Hazzan.

Chapter II discussed the literature review that informed this study, from the rich history of the study of abstraction as a part of human cognitive development to the significant role that abstraction has played in the evolution of the discipline of computer science. The fact that there has been consistent work in the field to enhance our understanding of abstraction in order to educate better computer science students provided a wealth of historical and current research.

Chapter III provided a description of the survey instrument and how it was developed, explored the population that was targeted for the study and, finally, discussed the methodology for data collection. Chapter III also posed three additional questions that were answered in the data analysis.

An in-depth analysis of the data collected by the survey instrument was the focus of chapter IV, and included detailed tables containing the responses to the questions on the survey instrument. The chapter also examined the results of the analysis done in SPSS. Chapter IV found that there were strong, statistically significant associations between the following:

- Discipline and conceptual abstraction

- Discipline and descriptive abstraction
- Conceptual and formal abstraction
- Conceptual and descriptive abstraction
- Gender and conceptual abstraction

It found that there were weak or no associations between the following:

- Discipline and formal abstraction
- Formal and descriptive abstraction
- Gender and formal abstraction
- Gender and descriptive abstraction
- Program and conceptual abstraction
- Program and formal abstraction
- Program and descriptive abstraction

Chapter V reviews the study in detail and continues on to discuss the findings of the study and the implications of the study for the computer science discipline and for future research.

5.2.1 Impact of One's Discipline on Abstraction Skills

Pace University has traditionally been a school for professional studies, and while that is changing and the University is growing its enrollment in the arts and sciences, many of the respondents to the survey (more than 75%) were enrolled in Pace's pre-professional programs.

. In theory, each major at the undergraduate level develops a process of critical thinking that informs the problem solving skills and processes appropriate

to that major. When majors are aggregated into disciplines, it is anticipated that the same or similar problem solving skills and processes are taught and shared. For example, a quantitative thought process is inculcated among the accounting and finance majors that is somewhat different from the quantitative skills taught to math majors or computer scientists. The students who have traditional quantitative course work have the opportunity to develop abilities that allow them to focus on problems of formal abstraction without experiencing the 'abstraction anxiety' that students lacking in-depth quantitative work often experience. Clearly students in every major develop skills that can be applied usefully in all three scales of abstraction. However, computer science majors seem to have an identifiable ability to function in all three scales of abstraction and to shift from a 'big picture' orientation to a 'small detail' orientation.

The literature strongly indicates that experience is important in the development of abstraction skills. 26.7% of the respondents were freshmen, students who had not yet been schooled in the abstraction skills relevant to their own discipline. This sparks the debate as to how much abstraction skill is inherent in the individual and how much it is a skill set learned by a combination of classroom work and experience.

5.2.2 Results of Studying the Three Scales of Abstraction

This study offered three scales of abstraction which research indicated may allow for a more specific and, perhaps, precise means of measuring and testing of abstraction skill: conceptual abstraction, formal abstraction and

descriptive abstraction. The central focus was on measuring whether computer science majors have a higher level of abstraction ability than other majors.

Conceptual Abstraction

The findings of this study indicate that conceptual abstraction is strongly related to the hypothesized abstraction scales: formal abstraction and descriptive abstraction. As shown in figure 1.1 in chapter I, conceptual abstraction is an umbrella skill that informs both formal and descriptive abstraction. Conceptual abstraction proved to be the only one of the three scales of abstraction that associates strongly with the two other scales, indicating that it serves as a foundation for abstraction ability. As in Kramer's observation [58, p.38] that his best computer science students have the best abstraction skills, it is indicated that conceptual abstraction associates strongly with discipline and with both formal and descriptive abstraction.

Formal Abstraction

Formal abstraction equates to removing detail to simplify and focus attention. It is the term representing the scale comprised of problem-solving items requiring structure to be seen beneath irrelevancies. The development of questions to test and measure formal abstraction abilities for this study was a challenging process and ultimately yielded the least definitive results. The attempt was a properly performed, first-ever try at building a brief, multiple-choice measure of an elusive cognitive ability. It was different in many ways from the

instruments suggested by Kramer and used by others. The respondents indicated that the formal abstraction questions were challenging and therefore an acceptable representation of the types of questions and problem statements that can be used to test for formal abstraction. Formal abstraction associated strongly with conceptual abstraction. However, analysis showed that formal abstraction was weakly associated with discipline and with descriptive abstraction.

Descriptive Abstraction

Descriptive abstraction equates to a process of generalization to identify the common core or essence. It is the term representing the scale comprised of items requiring interpretation and explication of most meaningful qualities. The research indicates that it is possible to measure an individual's ability to generalize or simplify in order to bring clarity to an item or object. Analysis shows that descriptive abstraction is strongly associated with discipline. It is also associated with conceptual abstraction. Discipline is not associated with formal abstraction.

The results of the study show a slightly different relationship between the three scales of abstraction. As shown in figure 4.1, below, while conceptual abstraction has been shown to be an 'umbrella scale' with which both formal and descriptive abstraction share association, formal and descriptive abstraction are unrelated except through the association with conceptual abstraction.

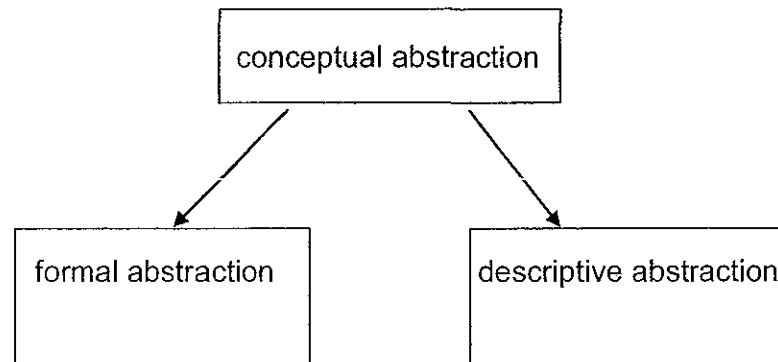


Figure 4.1. conceptual abstraction is associated with the other two scales, but formal and descriptive abstraction are not associated.

5.3 Testing the Hypotheses

The results of the analysis of the survey responses indicate potential for measuring abstraction ability in individuals. The responses from our population allowed us to answer the hypotheses that were formulated in chapter 1, though the outcomes were not always what was expected and were in some cases surprising.

Hypothesis 1: Students who have declared computer science as their major will score higher than students in other majors on measurements of all three scales of abstraction.

Finding: The performance of computer science majors on the survey instrument is the central focus of this study. Computer science students outperformed the other disciplines in answering questions designed to test each of the three scales of abstraction, but not as decidedly as the researcher would have predicted at the outset. The implications of this finding include support for Kramer's assertion

about his best computer science students being the most skilled at abstraction [58] and as well as a ratification of the way that abstraction is taught to computer science students.

Hypothesis 2: Conceptual abstraction is significantly related to formal abstraction.

Findings: Results show that conceptual abstraction is related to formal abstraction and serves as a foundation skill on which the abilities of formal abstraction, related as they are to formal problem solving, are able to thrive and function.

Hypothesis 3: Conceptual abstraction is significantly related to descriptive abstraction.

Findings: As with its relationship to formal abstraction, conceptual abstraction serves as the foundation skill set, or gateway, in which descriptive abstraction skills can operate most effectively.

Hypothesis 4: The relationship between formal abstraction and descriptive abstraction is weaker than the relationship between either of these and conceptual abstraction.

Findings: The research suggests that formal abstraction is only loosely associated with descriptive abstraction. Indications are that these two skills can operate efficiently and somewhat independently without supporting skills from the other. However, as noted in chapter II, as different as formal abstraction sounds from descriptive abstraction, when they are defined, they are not mutually exclusive. Detail may be removed in order to generalize, or generalization may

result from structural simplification. The implications for this finding are that formal and descriptive scales of abstraction are very different aspects of the cognitive ability that we call abstraction. While both share associations with the 'umbrella scale' of conceptual abstraction, it is possible for students in different disciplines to be strong in one, but not the other, while sharing a common grounding in conceptual abstraction.

Discussion of the Hypotheses: The research confirmed the four central hypotheses that are the foundation of this study. Computer science majors do have stronger abstraction skills than students in other majors. Conceptual abstraction is, indeed, a foundation skill that is indicative of an individual's abilities with both formal abstraction and descriptive abstraction. Formal abstraction and descriptive abstraction have shown to be abilities that are exclusive of one another and that one may be strong in one scale, but not in the other.

5.3.1 Additional Questions

Question 1: In examining academic disciplines, are there patterns evident in the abstraction scale scores?

Findings: Patterns became evident in the results indicating that certain majors show strength in certain scales. Technology majors placed very closely to computer science majors in conceptual and descriptive abstraction skills. However, they were substantially lower in formal abstraction skills. This may be because in Pace University's Seidenberg School of Computer Science and

Information Systems, technology majors (particularly information systems majors) share a few foundation courses with computer science majors including the CS1 and CS2 courses. Business majors performed respectably on the measurements of conceptual abstraction and moderately well in formal abstraction but poorly in descriptive abstraction. This may indicate that, as discussed in this chapter, that specific methodologies, like the case study method, develop specific abstraction skills. The performance of the business majors was particularly notable in the area of descriptive abstraction in that they performed poorly. Finance and accounting majors displayed a less than expected strength with both conceptual abstraction and formal abstraction but did moderately well with descriptive abstraction. In compared to the general business majors, they performed at the same level in conceptual abstraction, did a little better in formal abstraction, but were weaker in descriptive abstraction. Science and mathematics majors did surprisingly poorly (by comparison to business as well as finance and accounting majors) on all three scales of abstraction. Perhaps, for the science majors, this is attributable to the process of setting up experiments and interpreting results that aligns them to a focus on details. For the mathematics majors, perhaps this is due to the rigor of subscripts, superscripts, and the associated detail required by formal descriptions and proofs. This lends support to Knuth's assertion, discussed in chapter II, that there is indeed a difference in the thought processes of computer scientists and mathematicians. Social science and Education majors showed weakness with conceptual abstraction but performed on a par with business in

formal and descriptive abstraction. Based on their conceptual abstraction scores, Arts and Humanities majors do not see themselves as inclined toward abstraction. Yet they showed strength in formal abstraction at about the same level as computer science majors. Their score on descriptive abstraction, however was weak, below the level of Finance and Accounting majors. Nursing majors tested the weakest in all three scales of abstraction. Perhaps this is attributable to the practical, life and death, detailed nature of their discipline.

Question 2: Is a student's academic program (Honors, Traditional, CAP) reflected in their abstraction skill level?

Findings: Program affiliation had no impact on scores seen in this survey. That academic program is not significantly related to descriptive abstraction, although discipline (i.e. major) is, suggests that abstraction skill is something different from general academic ability, or perhaps a specialized component (like verbal, mathematical, and analytical or maybe even something more highly specialized such as musical ability). However, the relationship between academic program and descriptive abstraction is in the right direction, with honors students getting higher scores than traditional students, who received higher scores than CAP students. This is interpreted as a verification of the scale's validity.

Question 3: Is abstraction aptitude related to gender?

Findings: Queries run early in the testing process indicated clearly that there was a strong relationship between gender and conceptual abstraction. Yet, there is no relationship between gender and formal or descriptive abstraction abilities.

5.4 Key Findings of the Study

The study has shown that there is significance to the assertions of Kramer, Hazzan, Caspersen & Bennedsen and others that computer science majors have stronger abstraction skills and that the best computer scientists may be the best abstractionists. In addition, this study has shown the following:

- That it is possible to measure abstraction ability online, with short-answer questions. More work is necessary to determine how to do so with high level validity, particularly as relates to the scale of formal abstraction.
- Computer science majors are stronger than all other majors in the three scales of abstraction developed for this study, but not overwhelmingly so.
- Although unconfirmed by final grades (because this was not methodologically feasible), Kramer's assertion that abstraction may be the ability that separates the best students in computer science seems tenable. Discipline-specific abstraction skills may have a similar impact in other academic majors.
- Abstraction may be the ability that differentiates the mathematician from the computer scientist.
- If abstraction aptitude is key to success in computer science, there is absolutely no basis for presuming that one gender is better equipped for it than the other.

5.5 Implications of the Findings

Leading computer scientists including Kramer, Hazzan, Denning, Norman, Bergin, and others agree that the ability to develop abstraction skill is integral to

success in computer science and software development. Researchers in the discipline of computer science are striving to define abstraction as it pertains to computer science while also working to develop a means for testing it and developing it among students through course work. The results of this study confirm that there is a real potential to construct measures of abstraction that conform to accepted definitions and may be applied to fulfill the educational utilities that Kramer and Hazzan have seen for them.

It is also true that there is a troubling shortage of new talent pursuing the field of computer science. Identifying individuals, through testing, with strong abstraction ability is a tantalizing prospect. Potentially successful computer scientists could be identified among high school and college students.

Is there an ability that separates the skilled mathematician from the skilled computer scientist? The results of this research seem to indicate that, in fact, the level of abstraction skill may differentiate the two. Many in the field have relied on mathematics ability as an indicator of potential in computer science. However, Caspersen and Bennedsen [12] and Ventura [107] found that achievement in mathematics (such as SAT test scores) was not an indicator of success in CS 1 courses. Knuth [56] indicated that there was a difference between the skills that are honed by mathematicians from those honed by computer scientists.

These insights indicate that it is timely and necessary to develop a means to test individuals on abstraction skills. Test sets could be generic or discipline-specific and could be used to identify high school students with potential in

computer science, to test the development of undergraduates as they move through their courses, and to test the efficacy of mechanisms used in the curriculum to teach abstraction.

5.6 Recommendations for Future Research

It would be constructive to hold an 'abstraction summit' to gather the computer scientists, educators, programmers, cognitive development specialists and others who have done work in this area to arrive at definitions of what abstraction is as a theory as well as an applied skill. From this basis more effective testing and teaching could be developed.

The need for better ways to measure abstraction is, as noted, a driving force behind this study. There are clear benefits in recruitment and retention in developing tests for abstraction. The ability to identify potentially strong computer scientists exists through a continuation of this work. Another important opportunity lies in teaching abstraction to those whose test scores indicated weakness in their abstraction abilities. In light of the shortage of computer science students in American universities, it would be helpful to use such testing as a way to identify the relative abstraction skills of the individual, and then provide reinforcement for those whose abstraction skills are shown to be weaker. Dijkstra [29], Hazzan [46], Bergin [14] and others believe that abstraction can be cultivated. Providing a proper "apprenticeship" to those students who have the desire to be a successful computer scientist, but not yet the craftsman's ability, would have meaningful implications for our discipline. This indicates an

opportunity to strengthen the way that abstraction is taught in the classroom – not only in CS1 and CS2 - but throughout the computer science curriculum. Developing new and better ways to teach abstraction will be an important enhancement in the computer science discipline.

Another meaningful opportunity would be the use of the evidence of abstraction as an interdisciplinary skill to further the goals of those who, like Owen Astrachan [6], seek to find a place for computer science in the liberal arts pantheon. Abstraction is a core skill in computer science that has meaningful application across, perhaps, all academic disciplines. Further research may identify the abstraction skills necessary to be a competent historian, for example, and could fuel the work of informatics and, in turn, help to determine the skills of abstraction that computer scientists have in common with colleagues in other disciplines.

The 'keyness' of abstraction in computer science invites some speculations of a rather grand nature. Consider creativity. The research for this study gave some anecdotal indications that abstraction may play a role in creativity by bringing structure to the creative process and that abstraction may be, certainly in the descriptive abstraction scale, a manifestation of creativity. Another speculation that invites further research is the link between abstraction and time management. Effective time management can be seen as an application of formal abstraction and time management is certainly a key to academic success. It may even be that effective time management is a trait shared by the very good students of Kramer's who are skilled at abstraction. If,

in fact, further research indicates a link between time management and abstraction skill, this would inform the work going on in the area of teaching abstraction as well as the area of remedial abstraction because time management is a discipline that is widely taught with some definable success.

5.7 Opportunities for the Computer Science Discipline

Opportunities for the discipline of computer science are significant if the potential research opportunities identified herein are pursued. Developing a further understanding of abstraction can yield important insights into the learning process in general and the field of computer science specifically. Abstraction research has the potential to create opportunities and processes that will have beneficial impacts in secondary education, higher education and the professions.

It is likely that abstraction testing may be useful as a recruiting tool with abstraction tests being administered to promising students on the high school level. Using abstraction tests, it could be possible to include additional abstraction work in the high school computer science course work that would reinforce the high school computing curricula, including, perhaps, AP computer science programs. Identifying abstraction as an interdisciplinary skill that is learned in computer science is an way to give it's courses an interdisciplinary appeal. This is an opportunity for growth for the field of computer science and a means by which to extend the influence of computer science into other disciplines.

5.9 Conclusions

Developing an abstraction curriculum and teaching protocol that can be exported across disciplines has the potential to be one of the computer science field's great contributions to academic discourse and scientific discovery. Abstraction is a context dependant, yet widely accepted aspect of human cognition that is vitally important for success in the study of computer science, computer programming and software development. In practice, abstraction makes the intangible and complex understandable and malleable in ways that fuel the applied development process. This study is a needed addition to the literature, undertaken at a time when the potential for harnessing the concept and application of abstraction have captured the interest of many academics and practitioners in the field of computer science. In undertaking this study, the researcher has sought to develop a closer understanding of abstraction as it exists in the literature, as it is applied in the discipline of computer science and to test the efficacy of measuring it among a cross section of undergraduate students. The three scales developed within this study, conceptual abstraction, formal abstraction and descriptive abstraction offer meaningful constructs in terms of refining our understanding of the ways that abstraction can be used as a problem solving tool.

The practical application of measuring abstraction skills are significant and this study was both motivated and informed by this potential. It revealed the feasibility of developing discipline-specific abstraction tests. Such instruments can be valuable in better understanding the abilities and needs of computer

science students and professionals as well as in identifying students with the abstraction skills to be effective computer science practitioners. Therefore, identifying abstraction ability in individuals can be beneficial to the discipline of computer science. The present work has shown that it is possible to measure and compare abstraction skills between students in computer science and students in other disciplines.

1. Take Our Abstraction Survey - Win an American Express Gift Card!

Welcome to our survey! It is designed to measure abstraction abilities among college students. It should take you about 10 minutes to answer these questions though a few of them require some thought. By taking this survey you will be contributing to a research project. All respondents will be entered into a drawing to win one of four \$25 American Express gift cards. To be entered in the drawing, you must complete the entire survey and include your email address when prompted at the end of the survey. Your answers and personal information will all be kept confidential. Thank You!!!

1. What course is this?

2. Your Major?

3. Have you ever changed your major?

- yes
 no

4. If yes, what was your previous major?

5. I like to simplify and explain difficult concepts to help others understand

- Strongly disagree Somewhat disagree Neither agree nor disagree Somewhat agree Strongly agree

6. Specifics interest me more than generalization

- Strongly disagree Somewhat disagree Neither agree or disagree Somewhat agree Strongly agree

7. I tend to remember ideas better than details

- Strongly disagree Somewhat disagree Neither agree nor disagree Somewhat agree Strongly agree

8. Some activities that I enjoy include (choose all that apply)

- Problem solving
 Writing essays
 Making art
 Speaking in class
 Working in teams
 Computer games
 Using the computer
 Reading magazines
 Playing a musical instrument or singing

3.

9. You are in a race and you overtake the second person. What position are you in?

- a) first
- b) second
- c) last
- d) insufficient data

10. Suppose that a man from Mars can survive for exactly two weeks without food or sleep. What should he do at the end of the 14th day without having slept or eaten? Martians, like earthlings, cannot eat and sleep at the same time, but meals for Martians take several hours.

- a) eat first
- b) sleep first
- c) eat and sleep in the same order from two weeks ago
- d) eat and sleep in the reverse order from two weeks ago
- e) insufficient data

4.

11.

Suppose you have a robot that can remove items from a shelf, stack them up, and replace them on the shelf. The robot, which can hold only one item at a time, is controlled by these instructions:

<==== The Shelf

<==== A stack of items

1. Remove the item from the left-end of the shelf. This leaves the robot holding the item.
2. Remove the item from the top of the stack. This leaves the robot holding the item.
3. Place the item held by the robot onto the top of the stack.
4. Place the item held by the robot onto the right end of the line of items on the shelf.
5. Place the item held by the robot onto the left end of the line of items on the shelf

Four items are on the shelf. Give the sequence of instructions that get the robot to reverse their order. When the robot starts, the items look like this:



When the robot finishes, the items should look like this (although their exact placement on the shelf does not matter, only their order):



- a) 1 3 1 3 1 3 1 3 2 4 2 4 2 4 2 4
- b) 1 3 1 3 1 3 1 3 2 5 2 5 2 5 2 5
- c) 1 4 1 4 1 4 1 4
- d) 1 3 2 4 1 3 2 4 1 3 2 4 1 3 2 4

12. There is more than one correct answer to the previous item, but all are similar in form, and all are verbose. What might be done to condense them?

- a) New instructions could be added to the four that were given. For instance, a new instruction 6 could be "do instruction 1 then do 3."
- b) An instruction could be added that says "place the four items from the shelf onto the stack." It presumes nothing that the robot cannot already do.
- c) If it were not possible to extend the instruction set, we could at least adopt these expressive conventions among ourselves as a communication convenience.
- d) a and b
- e) a and c
- f) a, b, and c

13. Did any of your elementary school teachers show you the trick for adding the consecutive whole number between 1 and 10? First, you had to "fold the series of numbers in the middle and add them in pairs."* $1+10, 2+9, 3+8, 4+7, 5+6$. Then you could see the shortcut: get the sum of the first pair $1+10$ and multiply it by 5, which is the number of pairs. What can you say about this?

- a) It is an interesting anomaly.
- b) This technique can be applied to any consecutive sequence that starts at 1 and has an even number of entries.
- c) This technique can be applied to any consecutive sequence that starts at 1, whether it has an even or an odd number of entries.
- d) This technique can be applied to any consecutive sequence regardless of where it starts and whether the number of entries is even or odd.

5.

14. In Hi-Q a peg may jump over another peg into a vacant spot. When it does, the jumped peg is removed. The object of the puzzle is to discover a sequence of jumps that leaves only one peg left, in the center. The starting board is shown below, where each x represents a peg:

```

      x x x
      x x x
x x x x x x x
x x x   x x x
x x x x x x x
      x x x
      x x x
  
```

Your friend gives you the solution. The first move is expressed as $5 \Rightarrow 17$. But what does it mean?

- a) It could mean so many things that, in and of itself, it defies usefulness.
- b) It means "jump the peg at point 5 into point 17." Point 17 has to be the blank space, and point 5 is the middle peg in the second row.
- c) It means what is said by b, except that point 5 is actually the peg in the

- middle of the second column.
- d) It means what is said by b, except that point 5 could be any of four different points.

6.

15. It is two fifteen and the teacher asks the child to report the time. She answers one seventy-five. What would you say about this?

- a) This shows an excellent understanding and was probably said in jest.
- b) The child needs some instruction in the conventions of reporting times of day.
- c) The child does not know how to tell time.
- d) The child should be tested on other times in order to make a fair assessment.

16. What would be a good way of explaining what a Website is to someone who has never experienced the Internet?

- a) It is a domain-named location on a server, accessible with a browser.
- b) It is kind of like a telephone answering machine, but you "call it up" on a personal computer. What you get is a display on the screen.
- c) The Internet developed out of work funded by the Department of Defense in the late 1970s. Today, virtually all business and many individuals have Websites. You, personally, are better off with cable than dial-up.
- d) FrontPage is Microsoft's product for creating Webpages, but many people prefer Adobe's Dreamweaver. Knowledge of HTML (the hypertext markup language) is not necessary.

17. A description "from thirty thousand feet" is one that filters out details to focus on what is key. Which of the following would have made the most useful "thirty thousand feet" description of a cell phone in the old days, before people knew about them?

- a) An electronic device used by individuals for communications that is light-weight, and portable. This is relatively new technology made possible by microprocessors.
- b) A telephone with an independent number that connects into the phone system by radiowaves instead of a telephone line. Calls can be made and received from anywhere.
- c) A highly multipurpose electronic device that may function as a telephone, a camera, an email reader, a music player, and even more (e.g. television). The features depend on the model, but the product ownership is exceedingly widespread.
- d) If you think of a walky-talky you have the idea: mobile communications. Service tends to be good with coverage, signal strength, and bandwidth constantly improving.
- e) Usage details vary from model to model, but first you power-up, then you enter the number, and then you press the send button. It works very much like a regular phone, and it is safe to say that no one has trouble learning to use one.

7. Individual Information

This is a confidential survey for research purposes. Five respondents will be chosen at random for prizes. You must include your email address when prompted at the conclusion of this survey to win.

18. Your Age

19. Your Year in School

- A) Freshperson
- B) sophomore
- C) Junior
- D) Senior

20. Your Program at Pace

- Honors
- Traditional
- CAP
- Other (please specify)

21. Your Gender

- A) Male
- B) Female

8. Conclusion

22. For confidential research purposes, may we look up your grades after the semester?

yes

no

23. If yes, what is your Pace student email address? (you must enter your email address to be eligible for the prize drawing)

Please be assured that your responses will be kept confidential and that the information you provide will be reported only as an aggregate of all data collected. If you would like to learn more about the results of this survey, please contact Jonathan Hill at jhill@pace.edu

To: Jonathan Hill

From: Robert Birrer

Date: 7/27/07

Hi Jonathan:

I wanted to provide you feedback on your on-line survey.

Your survey was designed very well. Respondents experienced a questionnaire was no more than 10 minutes in length with the potential for cash incentive, which is well within the standards for executing a web based survey.

The survey flowed well, invoking a variety of question types, (single mention, multiple response, single mention with other specifics) within screens that were not too busy looking or too lengthy (where a respondent would have to scroll). This setup and the relatively short time it takes to complete, assures keeping the respondents attention and yielding a high response rate. I especially liked the functionality of being able to mouseover any part of the response text and click, thereby selecting the radio button.

Your use of a fully anchored 5 point rating scale is a sound decision. Research has shown that there are no significant differences with the deployment of a fully anchored versus an end anchored 5 point scale. Also, the choice of 5 response categories versus 2 creates the greatest gain in reliability. It is also known that the use of an odd numbered scale is easier to complete and will yield more accurate information. You also laid out the scale direction with the least desirable (negative) end of the scale first, which is the preferred approach.

From everything that I have seen, this is a sound data collection instrument. It's clear, concise and efficient.

Regards:

Bob Birrer

Manager of Production Programming

Harris Interactive

(609) 919-2535

References

- [1]. Adey, P., and Shayer, M. (1994) *Really Raising Standards: Cognitive Intervention and Academic Achievement*, (Routledge: London, England).
- [2]. Aiken, R.M., Hughes, C. E. , Moshell, , J. M., (2002) "Computer science curriculum for high school students"
- [3]. Alexander, S., Amillo, J., Boyle, R., Clark, M., Daniels, M., Laxer, C., Loose, K., and Shinnars-Kennedy, D. (2003) "Case Studies in Admissions to and Early Performance in Computer Science Degrees", *ACM SIGCSE Bulletin*, vol. 35, no. 4, 137-147.
- [4]. Almstrum, V.L., Barker, L.J., Owens, B.B., Adams, E., Aspray, W., Dale, N.B., Dann, W., Lawrence, A., and Schwartzman, L. (2005) "Buidiling a Sense of History: Narratives and Pathways of Women Computing Educators", *ACM SIGCSE Bulletin*, vol. 37, no. 4, 173-189.
- [5]. Alphonse, C., and Ventura, P. (2002) "Object Orientation in CS1-CS2 by Design", *Proceedings of the 7th Annual Conference on Innovation and Technology in Computer Science Education*, 70-74.
- [6]. Alt, C., Astrachan, O., Forbes, J., Lucic, R., and Rodger, S. (2006) "Social Networks Generate Interest in Computer Science", *Technical Symposium on Computer Science Education, Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education*, 438-442.
- [7]. Astrachan, O.L., (1997), *A Computer Science Tapestry*. McGraw-Hill
- [8]. Beaubouef, T., and Mason, J. (2005) "Why the High Attrition Rate for Computer Science Students: Some Thoughts and Observations", *ACM SIGCSE Bulletin*, vol. 37, no. 2, 103-106.
- [9]. Beck, R.E., Cassel, L.N., and Austing, R.H. (1989) "Computer Science: A Core Discipline of Liberal Arts and Sciences", *Technical Symposium on Computer Science Education, Proceedings of the Twentieth SIGCSE Technical Symposium on Computer Science Education*, 56-60.
- [10]. Bennedsen, J., and Caspersen, M.E. (2004) "Programming in Context – A Model-First Approach to CS1", *Proceedings of the thirty-fifth SIGCSE Technical Symposium on Computer Science Education*, 477-481.

- [11]. Bennedsen, J., and Caspersen, M.E. (2005) "An Investigation of Potential Success Factors for an Introductory Model-Driven Programming Course", *International Computing Education Research Workshop, Proceedings of the 2005 International Workshop on Computing Education Research*, 155-163.
- [12]. Bennedsen, J., and Caspersen, M.E. (2006) "Abstraction Ability as an Indicator of Success for Learning Object-Oriented Programming?", *ACM SIGCSE Bulletin*, vol. 38, no. 2, 39-43.
- [13]. Bergin, J., Koffman, E., Proulx, V.K., Rasala, R., and Wolz, U. (1999) "Objects: When, why, and how?", *Journal of Computing in Small Colleges*, vol. 14, no. 4.
- [14]. Bergin, J., and Reilly, R. (2005) "Programming: Factors That Influence Success", *Proceedings of the 36th SIGSCE Technical Symposium on Computer Science Education*, 411-415.
- [15]. Berztiss, A. (1987) "A Mathematically Focused Curriculum for Computer Science", *Communications of the ACM*, vol. 30, no. 5, 356-365.
- [16]. Brooks, F (1975) [Frederick P. Brooks, Jr. The Mythical Man-Month, Addison-Wesley, 1975, page 7].
- [17]. C. Bohm and G. Jacopini, "Flow Diagrams, Turing Machines and Languages with Only Two Formation Rules" *Communications of the ACM*, Volume 9, Number 5, May 1966, pages 366-371
- [18]. Bond, T.G. (2004) "Piaget and the Pendulum", *Science and Education*, vol. 13, no. 4-5, 389-399.
- [19]. Bucci, P., Long, T.J., Weide., (2001) "Do We Really Teach Abstraction?" , *Proceedings of SIGCSE 2001, ACM*, 26 - 30
- [20]. Burg, J., and Lüttringhaus, K. (2006) "Entertaining with Science, Educating with Dance", *ACM Computers in Entertainment*, vol. 4, no. 2, 1-15.
- [21]. Capretz, L. F. (2003), "A Brief History of the Object Oriented Approach", *ACM SIGSOFT Software Engineering Notes, March*, vol. 28, no. 2
- [22]. Cimino, J. (2005) "In Defense of the Desiderata", *Journal of Biomedical Informatics*, Volume 39, Issue 3, June 2006, Pages 299-306

- [23]. Cohn, S.P. (2003), Transcript of the Meeting of Department of Health and Human Services, National Committee on Vital and Health Statistics, Subcommittee on Standards and Security, Found as of this writing at <http://www.ncvhs.hhs.gov/030325tr.htm>
- [24]. Connelly, R., Hadimioglu, H., Herscovici, D., Ivanov, L., and Hoffman, M. (2005) "Course Continuity in the Computer Science Curriculum", *Journal of Computing Sciences in Colleges*, vol. 21, no. 2, 172-176.
- [25]. Computer Technology Industry Association – Electronics Industry Data Exchange Group, (2007), *Glossary of Terms*
http://eidx.comptia.org/reference/glossary/gloss_a.aspx
- [26]. Cuny, J., and Aspray, W. (2002) "Recruitment and Retention of Women Graduate Students in Computer Science and Engineering", *ACM SIGCSE Bulletin*, vol. 34, no. 2, 168-174.
- [27]. Devlin, K. (2003) "Why Universities Require Computer Science Students to Take Math". *Communications of the ACM*, vol. 46, no. 9
- [28]. Denning, Peter J. (2004), "The Field of Programmers Myth.", *Communications of the ACM*, vol. 47, no. 7, page 20.
- [29]. Dijkstra, ("Algorithms in Modern Mathematics and Computer Science" in Lecture Notes in Computer Science, 1981, volume 122, pages 82-99; also in the anthology Selected Papers in Computer Science, Cambridge University Press, 1996, Chapter 4, pages 87-114)
- [30]. Dobbs, D. (2007) "The Gregarious Brain", *The New York Times Magazine*, July 8, 2007 p. 44 - 49
- [31]. D'Onofrio, N. (2005) Transcript of Remarks to the New York State Education Summit, Albany, NY, 11/2/2005, from IBM, p. 5
- [32]. Dodds, Z., and Karp, L. (2006) "The Evolution of a Computational Outreach Program to Secondary Students", *Technical Symposium on Computer Science Education Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education*, vol. 38, no. 1, 448-452.
- [33]. Dunnigan, J.F. (1992) "*The Complete War Games Handbook*", (Morrow, New York)

- [34]. Eckerdal, A., McCartney, R., Mostrom, J.E., Ratcliffe, M., Sanders, K., and Zander, C. (2006) "Putting Threshold Concepts into Context in Computer Science Education", *Annual Joint Conference Integrating Technology into Computer Science Education, Proceedings of the 11th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education*, 103-107.
- [35]. El-Nasr, M.S., and Smith, B.K. (2006) "Learning Through Game Modding", *ACM Computers in Entertainment*, vol. 4, no. 1, 1-20.
- [36]. Ennis, R. (1985) *Cornell Critical Thinking Tests Level X & Level Z Manual*, 3rd ed., (Critical Thinking Books & Software: Seaside, CA).
- [37]. Evans, G.E., and Simkin, M.G. (1989) "What best predicts computer proficiency?", *Communications of the ACM*, vol. 32, no. 11, 1322-1327.
- [38]. Ferguson, R. (1986) "Abstraction Anxiety: A Factor of Mathematics Anxiety", *Journal for Research in Mathematics Education*, vol. 17, no. 2, 145-150.
- [39]. Fitch, F.B. (1974) *Elements of Combinatory Logic*, (Yale University Press: New Haven, Connecticut).
- [40]. Fox, R., Newell, G., and Frank, C. (2004) "Experiences Hosting High School Summer Camps to Promote University Outreach", *Proceedings of the 2nd Annual Conference on Mid-South College Computing*, 166-172.
- [41]. Freidman, Thomas, (2007) "Laughing and Crying", *The New York Times Digital Edition*, May 23, 2007
- [42]. Galpin, V.C., Sanders, I.D., Chen, P. (2007) "Learning Styles and Personality Types of Computer Science Students at a South African University", *Annual Joint Conference Integrating Technology into Computer Science Education, Proceedings of the 12th Annual SIGCSE conference on Innovation and Technology in Computer Science Education*, 201-205.
- [43]. Garson, G.D. (2007) *Quantitative Research in Public Administration*, found as of this writing at <http://www2.chass.ncsu.edu/garson/pa765/chisq.htm>
- [44]. Gersting, Judith L. (1987) *Mathematical Structure for Computer Science*, 2nd Ed. (W.H. Freeman:New York).
- [45]. Hazzan, O. (2002) "Reducing Abstraction Level When Learning Computability Theory Concepts", *Annual Joint Conference Integrating Technology into Computer Science Education, Proceedings of the 7th annual conference on Innovation and Technology in Computer Science Education*, 156-160.

- [46]. Hazzan, O., and Kramer, J. (2006) "Abstraction in Computer Science and Software Engineering: A Pedagogical Perspective", *System Design Frontier Journal*, vol. 3, no. 12, 1-9.
- [47]. Helps, C. Richard , Jackson, Robert B. , Romney, Marshall B. (2005) "Student expectations of computing majors", Proceedings of the 6th conference on Information technology education (SIGITE '05)
- [48]. Houle, B.J. (2004) *Adult Student Persistence in Web-Based Education*, Ph.D Dissertation. New York University, (UMI: Ann Arbor, MI).
- [49]. Howell, D.C. (1999) *Fundamental Statistics for the Behavioral Sciences*, 4th ed., (Brooks/Cole: Boston, MA).
- [50]. IEEE, (2004), Software Engineering 2004: Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering, "The Joint Task Force on Computing Curricula", ACM/IEEE,
- [51]. Inhelder, B., and Piaget, J. (1958) *The Growth of Logical Thinking*, (Routledge and Kegan Paul: London, England).
- [52]. Jacobson, I., Christerson, M., Jonsson, P., and Overgaard, G. (1992) *Object-Oriented Software Engineering: A Use Case Driven Approach*, (Addison-Wesley: England).
- [53]. The Joint Task Force on Computing Curricula. (2001) "Computing Curricula 2001 Computer Science", *Journal on Educational Resources in Computing*, vol. 1, no. 3es, 1-240.
- [54]. Karam, M., Keirouz, W., Hage, R., "An Abstract Model for Testing MVC and Workflow Based Web Applications, *Advanced International Conference on Telecommunications and International Conference on Internet and Web Applications and Services (AICT-ICIW'06)*
- [55]. Knuth, D.E. (1974) "Computer Programming as An Art" The ACM Turing Award Lecture, found at <http://fresh.homeunix.net/~luke/misc/knuth-turingaward.pdf>
- [56]. Knuth, et al, (1979) "Algorithms in Modern Mathematics and Computer Science" Proceedings of the ("Algorithms in Modern Mathematics and Computer Science" in Lecture Notes in Computer Science, 1981, volume 122, pages 82-99; also in the anthology Selected Papers in Computer Science, Cambridge University Press, 1996, Chapter 4, pages 87-114)
- [57]. Knuth, D., (1996) "Algorithmic Themes," Selected Papers in Computer Science, Cambridge University Press, 1996, (Chapter 5, pages 115-122)

- [58]. Kramer, J. (2007) "Is Abstraction The Key to Computing?" *Communications of the ACM*, vol. 50, no. 4, 36-42.
- [59]. Kumar, A.N., Beidler, J., Bhagyavati, Farian, H., Haas, M., Kushleyeva, Y., Lee, F., and Russell, I. (2005) "Innovation in Undergraduate Computer Science Education", *Journal of Computing Science in Colleges*, vol. 21, no. 2, 138-142.
- [60]. Kurtz, B.L. (1980) "Investigating the Relationship Between the Development of Abstract Reasoning and Performance in an Introductory Programming Class", The Papers of the Eleventh SIGCSE Technical Symposium on Computer Science Education, SIGCSE Bulletin, 110-117.
- [61]. LaVoie, D., "Prefatory Note: The Origins of "The Agorics Project"", found as of this writing at <http://www.philsalin.com/hth/hth.html>
- [62]. Lester, C.Y., and Brown, M. (2004) "Creating Gender Parity: An Instruction Aide's Influence", *ACM Journal of Educational Resources in Computing*, vol. 4, no. 1, 1-14.
- [63]. Lewandowski, G., Johnson, E., and Goldweber, M. (2005) "Fostering a Creative Interest in Computer Science", *Technical Symposium on Computer Science Education, Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education*, vol. 37, no.1, 535-539.
- [64]. Liu, Y.A., Stoller, S., Gorbovitski, M., Rothamel, T., Liu, Y.E., (2005) "Incrementalization Across Object Abstraction", *ACM SIGPLAN Notices, Proceedings of the 20th annual ACM SIGPLAN conference on Object oriented programming, systems, languages, and applications OOPSLA '05*
- [65]. Maj, S.P., Veal, D., Duley, R., (2001) "A Proposed New High Level Abstraction for Computer Science", *Proceedings of the thirty-second SIGCSE technical symposium on Computer Science Education, July, 2001, p. 199 - 203*
- [66]. Maldonado, H., Lee, J.E.R., Brave, S., Nass, C., Nakajima, H., Yamada, R., Iwamura, K., and Morishima, Y. (2005) "We Learn Better Together: Enhancing eLearning with Emotional Characters", *Proceedings of the 2005 Conference on Computer Support for Collaborative Learning*, 408-417.
- [67]. Markoff, John, (2005) "Three Technology Companies Join to Finance Research", *The New York Times Digital Edition, December 15, 2005*
- [68]. Mazlock, L.J. (1989) "Identifying potential to acquire programming skill", *Communications of the ACM*, vol. 23, no. 1, 14-17.

- [69]. Moorman, P., and Johnson, E. (2003) "Still a Stranger Here: Attitudes Among Secondary School Students Towards Computer Science", *Annual Joint Conference Integrating Technology into Computer Science Education, Proceedings of the 8th Annual Conference on Innovation and Technology in Computer Science Education*, vol. 35, no. 3, 193-197.
- [70]. Mosley, P.H. (2002) *The Cognitive Complexities Confronting Developers Using Object Technology*, DPS Dissertation, Pace University, (UMI: Ann Arbor, MI).
- [71]. Means, H.W. (1991) "Using Literature in a Computer Science Service Course: Improving Abstract/Critical Thinking Skills", *Journal of Computing Sciences in Colleges*, vol. 6, no. 5, 30-34.
- [72]. Merriram-Webster Online Dictionary found as of this writing at <http://www.m-w.com/dictionary/abstraction>
<http://www.m-w.com/dictionary/abstracting>
- [73]. Lachman, R., Lachman, J., Butterfield, E.C., "Cognitive Psychology and Information Processing: An Introduction", (Lawrence Ehrbach Associates, New York)
- [74]. Mulmuley, K. (1987) *Full Abstraction and Semantic Equivalence*, (MIT Press: Cambridge, MA).
- [75]. Norman, D.A. (1988) *The Psychology of Everyday Things*, (Basic Books, Inc.: New York, New York).
- [76]. Norman, D.A. (1993) *Things That Make Us Smart: Defending Human Attributes In The Age of The Machine*, (Addison-Wesley: Reading, MA).
- [77]. Norman, D.A. (2004) *Emotional Design: Why We Love (or Hate) Everyday Things*, (Basic Books: New York, New York).
- [78]. Norman, D.A. (2007) Author's Web Site <http://www.jnd.org/>
- [79]. Nowaczyk, R.H. (1983) "Cognitive skills needed in computer programming", *Paper presented at the Annual Meeting of the Southeastern Psychological Association*, Atlanta, Georgia.
- [80]. Olivieri, L.M. (2005) "High School Environments and Girls' Interest in Computer Science", *ACM SIGCSE Bulletin*, vol. 37, no. 2, 85-88.
- [81]. Or-Bach, R., Lav, I., (2002) "Cognitiive Activities of Abstraction in Object Orientation: an Empirical Study", *The SIGCSE Bulletin*, Vol. 36, No. 2, June.

- [82]. Parnas, David Lorge, (2007) Forum, *Communications of the ACM*, vol. 50, no. 6, June
- [83]. Perrenet, J., Groote, J.F., Kaasenbrood, E. (2005) "Exploring Students' Understanding of the Concept of Algorithm: Levels of Abstraction", *Annual Joint Conference Integrating Technology into Computer Science Education, Proceedings of the 10th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education*, 64-68.
- [84]. Perrenet, J., and Kaasenbrood, E. (2006) "Levels of Abstraction in Students' Understanding of the Concept of Algorithm: The Qualitative Perspective", *Annual Joint Conference Integrating Technology into Computer Science Education, Proceedings of the 11th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education*, 270-274.
- [85]. Pollack, L., McCoy, K., Carberry, S., Hundigopal, N., and You, X. (2004) "Increasing High School Girls' Self Confidence and Awareness in CS through a Positive Summer Experience", *Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education*, vol. 36, no. 1, 185-189.
- [86]. ProgressiveArt.com (2007) Web Glossary found as of this writing at http://www.progressiveart.com/art_terms.htm
- [87]. Ralston, A., and Shaw, M. (1980) "Is Computer Science Really That Unmathematical?", *Communications of the ACM*, vol. 23, no. 2, 67-70.
- [88]. Rauchas, S., Rosman, B., Konidaris, G., and Sanders, I. (2006) "Language Performance at High School and Success In First Year Computer Science", *ACM SIGCSE Bulletin, Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education*, vol. 38, no. 1, 398-402.
- [89]. Recker, M., Dorward, J., Dawson, D., Halioris, S., Liu, Y., Mao, X., Palmer, B., and Park, J. (2005) "You Can Lead a Horse to Water: Teacher Development and Use of Digital Library Resources", *International Conference on Digital Libraries, Proceedings of the 5th ACM/IEEE-CS Joint Conference on Digital Libraries*, 1-8.
- [90]. Roedkelein, J.(2004) *"Imagery in Psychology, A Reference Guide"*, (Praeger/Greenwood, Westport, CT)
- [91]. Schacter, J., (1999) "The Impact of Education Technology on Student Achievement: What the Most Current Research Has to Say", *Milken Exchange on Education Technology*, 1-12.

- [92]. Scragg, G., and Smith, J. (1998) "A Study of Barriers to Women in Undergraduate Computer Science", *Technical Symposium on Computer Science Education, Proceedings of the Twenty-Ninth SIGCSE Technical Symposium on Computer Science Education*, vol. 30, no.1, 82-86.
- [93]. Sevo, R. (2005) "The Policy Push for Women in Science and Engineering", *ACM International Conference Proceeding Series, Proceedings of the International Symposium on Women and ICT: Creating Global Transformation*, vol. 126, no. 3, 1-6.
- [94]. Shaw, Z., (2006) A Dyslogic for the Industry (blog) found as of this writing at http://www.zedshaw.com/rants/indirection_is_not_abstraction.html
- [95]. Sorkin, S., Tupper, D., and Harmeyer, K. (2005) "Instructional Multimedia Institutes For Mathematics, Science, and Technology Educators" *Journal of Computing Science in Colleges*, vol. 20, no. 3, 28-37.
- [96]. Sprague, P., Schahczenski, C., (2001) "Abstraction the Key to CS1" , *Journal of the Consortium for Computing in Small Colleges*, vol. 17, no. 3 (February, 2002)
- [97]. Stanek, S., (2005) "Want to Change Your Major? You're Not Alone", *Chicago Tribune*, July 31, 2005 (Chicago, IL, Final Edition)
- [98]. Stanley, J.C., Keating, D.P., and Fox, L.H. (1974) *Mathematical Talent: Discovery, Description, and Development*, (John Hopkins University Press: Baltimore, Maryland).
- [99]. STATSOFT (2003), *Introduction to Statistics*, Online Textbook, found as of this writing at <http://www.statsoft.com/textbook/stbasic.html>
- [100]. Steele Hanson, A.M. (1986) *Critical Thinking Ability of Novice and Expert Computer Programmers*, University of Idaho.
- [101]. Stockard, R., Klassen, M., and Akbari, A. (2005) "Computer Science Higher Education Pipeline", *Journal of Computing Science in College*, vol. 20, no. 3, 102-113.
- [102]. Suppes, Patrick, Bing Han, Julie Epelboim, and Zhong-Lin Lu, (1999) "Invariance of brain-wave representations of simple visual images and their names", *Publications of the National Academy of Sciences, Psychology*, 96. 25
- [103]. The Tate Museum (2007) Online Glossary found as of this writing at <http://www.tate.org.uk/collections/glossary/definition.jsp?entryId=8>

- [104]. Taylor, H.G., and Mounfield, L.C. (1989) "The Effect of High School Computer Science, Gender, and Work on Success in College Computer Science", *ACM SIGCSE Bulletin, Proceedings of the Twentieth SIGCSE Technical Symposium on Computer Science Education*, vol. 21, no. 1, 195-198.
- [105]. Tulloch, W., Miller, M., (2002) "Institutions as Abstract Boundaries" *e-rights Talks* found as of this writing at <http://www.erights.org/talks/categories/categories.html>
- [106]. Ventura, P.R. (2003) *On the Origins of Programmers: Identifying Predictors of Success for an Objects First CS1*, Diss. The State University of New York at Buffalo, (UMI: Ann Arbor, MI).
- [107]. Ventura, P.R., and Ramamurthy, B. (2004) "Wanted: CS1 Students. No Experience Required", *ACM SIGCSE Bulletin, Proceedings of the 35th SIGCSE technical symposium on Computer Science Education*, vol. 36, no. 2, 240-244.
- [108]. Verelst. J., (2004), "The Influence of The Level of Abstraction on The Evolvability of Conceptual Models of Information Systems", *Proceedings of the 2004 International Symposium on Empirical Software Engineering*.
- [109]. Von Glaserfeld, E.(1995) *"Radical Constructivism: A Way of Knowing and Learning"*, (Routledge, London, UK)
- [110]. Wadsworth, A., Quoted in <http://evolution.massey.ac.nz/assign2/MH/webpage.htm>
- [111]. Walker, H., and Schneider, G.M. (1996) "A Revised Model Curriculum for a Liberal Arts Degree in Computer Science", *Communications of the ACM*, vol. 39, no. 12, 85-95.
- [112]. Wilkens, L., (1999) "What Concepts of Computer Science are Essential for Students Entering the Field?" *Proceedings of The Consortium for Computing in Small Colleges Fourth Annual NORTHEASTERN CONFERENCE*
- [113]. Wilson, B.C. (2002) "A Study of Factors Promoting Success in Computer Science Including Gender Differences", *Journal of Computer Science Education*, vol. 12, no.1-2, 141-164.
- [114]. Wilson, B.C., and Shrock, S. (2001) "Contributing to Success in an Introductory computer Science Course: A Study of Twelve Factors", *ACM SIGCSE Bulletin, Proceedings of the Thirty Second SIGCSE Technical Symposium on Computer Science Education*, vol. 33, no. 1, 184-188.