Using the Kolbe A™ Conative Index to Study the Retention of Computer Science Students

Robert Lingard, Brenda Timmerman, and Elizabeth Berry
California State University, Northridge, CA 91330
rlingard@csun.edu, eberry@csun.edu, btimmer@csun.edu

Abstract - **Recent studies at California State University, Northridge using the Kolbe A™ index have shown differences in conative profiles between computer science students near graduation and those just entering the program. Since conation, or a person's inherent talent or natural way of doing things, relates to how a student approaches learning, it may provide clues as to why some students quit the program prior to graduation. The Kolbe instrument used in this study is a tool for measuring conative talents. It has been widely used in industry to aid in management activities and assist in the formation of effective teams. Its use in education has been limited, but it has potential for providing insight regarding differences in student learning and in understanding issues related to student retention. For example, one recent study showed a significantly higher *implementor* conative instinct among entering students than existed among graduating students. This suggests that some students with strong *implementor* talents may have dropped out of the program, possibly because they were not given sufficient opportunity to engage in "hands on" activities. It may be that some students are discouraged from continuing in the major because they find the learning environment incompatible with the natural ways they approach learning.**

Index Terms - Active Learning, Conation, Student retention

INTRODUCTION

The overall graduation rate (defined as the percentage of incoming freshmen that graduate within six years) at California State University is disappointingly low and within the department of Computer Science the situation is even worse. Of the more than 200 students who decide to major in computer science each year, fewer than 40 typically complete the program. Although many of these students eventually graduate in other fields, the high number leaving the computer science program is of concern.

The high dropout rate among computer science students is not unique to CSUN. Even among universities with excellent overall graduation success, computer science retention rates are, by comparison, poor. For example, the University of Waterloo reports an average graduation rate of 73.7%, but the computer science rate is only 58.1%, the lowest of all reported programs [1]. In a recent study at the University of Minnesota on the reasons for student dropouts [2], it was noted that "students with interests in physical science and computer science were more likely to leave." The *Conseil de la Science et de la Technologie*, in Quebec reported that in "computer science technology at the collegial level, more than 70% of the young people who register abandon the program along the way." [3] The same report indicates that the overall dropout rates in the scientific and technical disciplines as a whole is only about 50%. It seems that the retention of computer science students is more difficult than for other majors. In order to improve retention a greater understanding of the reasons so many students leave the program is needed.

One possible explanation is that many students decide to major in computer science without really knowing much about the field and whether it is a good match for their interests and abilities. There is, however, significant evidence suggesting that other factors also contribute to the high dropout rate. The wide diversity of preparedness among students entering the program has been cited as a problem. [4] In many cases early courses in the program are directed at the well-prepared students, ignoring the needs of those with little prior background in the field. Many of these less prepared students are women and economically disadvantaged students whose pre university experience did not include significant computer use. It might be possible to improve retention through changes in the way the subject is taught. At Loughborough University, a report documented a correlation between improvements in teaching and a reduction in the dropout rate. [5]

In particular, active learning techniques have been given credit for improving learning in many studies. [6,7,8] In this regard, it is interesting to note that women seem to have a much greater preference for active learning than do men. [9] Additionally, some interesting results from studies at CSUN using the Kolbe A™ index [10] have suggested a relationship between students' natural ways of doing things (i.e., their conative talents), like problem solving, and success in the computer science program. These studies further indicate that changes in instructional techniques could have positive effects with respect to both student achievement and retention.

THE KOLBE A™ INDEX

The Kolbe A™ index [11] is an instrument that measures conation or a person's inherent talent or natural way of doing things and predicts what a person will or will not do, given the freedom to act. Where intelligence tests measure cognitive abilities and personality tests measure values and preferences, the Kolbe A™ index measures the conative or the way people act while trying to achieve goals.

Conation is related to learning styles. The term "learning styles" refers to the ways individuals interact with, take in, and process new stimuli or information. Preferred learning styles are influenced by intelligence, personality, psychomotor abilities, and motivation, and conation relates strongly to motivation. Conative talents are different from but similar to learning style preferences. It isn't that understanding conation is better than understanding learning styles, but both are important to help students learn, and until recently, the concept of conation has been neglected. [12]

The Kolbe A™ index identifies four modes or striving instincts – *fact finder*, *follow thru*, *quick start*, and *implementor* – each prompting people to act in a certain way. The *fact finder* probes, asks questions, weighs pros and cons, and uses experience. This person collects data and establishes priorities before making a decision. The *follow thru* individual seeks structure and makes schedules. This person needs a sense of order and plans ahead. The *quick start* individual innovates, takes risks, improvises, and plays hunches. When asked to give a presentation, the *quick start* comfortably ad-libs. The *implementor* uses space and materials, builds, constructs, and uses hands-on equipment with ease. This person creates handcrafted models and insists on quality materials.

Everyone has each of these abilities to some degree and can operate in any of these modes. However, people are most productive and comfortable when they are able to utilize their strongest conative talents. Figure 1 is a typical example that graphically depicts the degree to which each of these abilities is present. The four striving instincts are expressed through three possible operating zones, indicating how the individual will make use these talents. A score of 7 to 10 in a given mode places the individual in the insistence zone. This indicates *how the person will act*. A score of 4 to 6 indicates the response or accommodating zone or *how the person is willing to act*, and a score of 1 to 3 represents the prevention or resistance zone or *how the person won't act*. It doesn't mean people can't act in all of these ways; it just means that some ways won't come naturally.

According to a report published by the Kolbe Corporation [13], the Kolbe A™ index has undergone an extensive statistical analysis of more that 24,000 individual results. The analysis has shown that the index is highly reliable in terms of repeated testing. That is, there was no significant difference in individual scores when the test was repeated over both short and long periods of time. Furthermore, the Kolbe instrument was shown to be unbiased with regard to age, gender, race, or national origin.

MOTIVATION

Only a few years ago there were so many students wanting to study computer science that many schools took steps to limit the number of majors. At CSUN the computer science program was declared "impacted" which allowed the department to restrict admissions to only the most qualified students. No one was concerned with the dropout rate when there were too many students to begin with. In fact, some argued that the high dropout rate among beginning computer science students was a good thing, that the students who left the program probably didn't belong in computer science in the first place.

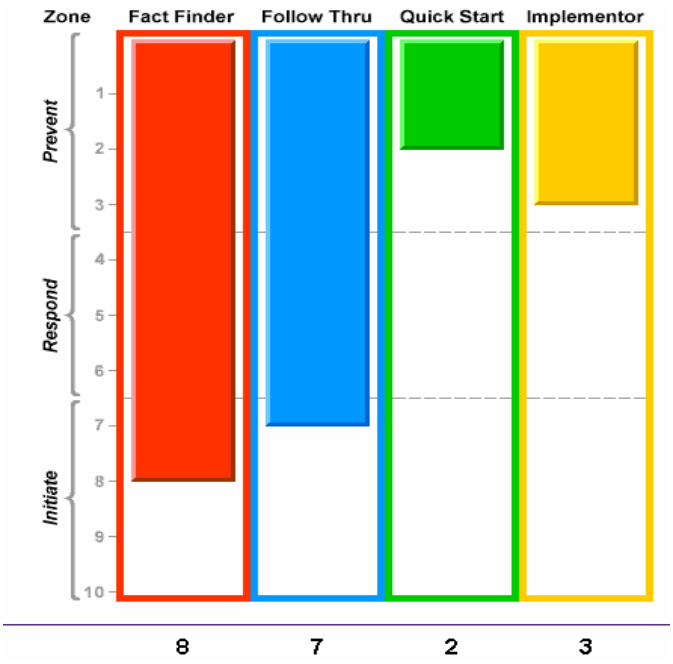

FIGURE 1
SAMPLE KOLBE A™ INDEX RESULTS

Often courses were designed to eliminate students from the program who did not seem to have the appropriate abilities and skills for the major. According to Robert Balmer, dean of engineering at Union College, "[at many schools] freshman and sophomore courses . . . were viewed as a way to weed out weak students. If you couldn't hack it, nobody cared. They just wanted the best students."[14] No one was concerned that the way these classes were taught might also cause some well-qualified students to leave the program, nor did they worry that they were likely weeding out some students who could eventually become good contributors to the field. While it is a good thing when a student discovers early on that the field of study they have begun is not appropriate for their interests or abilities, it is extremely unfortunate when the decision to leave the major is based on a school's failure to provide a learning environment that supports the student's needs or natural modes of learning.

Today, in most colleges and universities enrollments in computer science programs are declining, and there is a recognized need to try to retain more students. Certainly, there is a valid reason for trying to understand why students leave the program and for making changes in the program to improve retention. However, there are more important reasons for being concerned with retention besides the probably temporary condition of declining enrollments. When students who do not fit the computer science mold are discouraged from continuing, the field may be losing valuable contributions from individuals with diverse points of view.

When women and minorities are discouraged, the field might be less inclined to develop computer based systems to satisfy their needs or interests, and their diverse views may not be considered in the systems that are developed. Similar arguments can be made for students who may have different ways of learning or alternate approaches to problem solving.

An additional consideration relates specifically to the field of software engineering. In software engineering most projects are developed as team efforts, and it has been claimed that, in general, project success is enhanced by team synergy or a balance of skills and talents. Industrial users of the Kolbe index for forming teams attest to its success. This approach defines team synergy in terms of an optimal distribution of conative talents. [15] These claims were supported by a study at CSUN of software engineering student teams over a two-year period. It was shown that there was a statistically significant correlation between team synergy as defined by Kolbe and success on class projects. [16] A similar study conducted at the University of Arizona within the Department of Systems and Industrial Engineering produced comparable results. [17]

The CSUN study also revealed the interesting fact that it was difficult to form synergistic teams because of a lack of certain conative talents among the population of software engineering students. Another reason for being concerned about retention, therefore, might be to try to retain students in the program with those conative talents that seem to be scarce among upper division computer science students.

In summary, there are likely many reasons students leave the computer science program. It is not only because they lack interest in the field. Many of the students who leave the program may do so because of the way beginning computer science courses are taught. There are many reasons to be concerned about retention of students in computer science, but in order to improve the situation, a better understanding of the factors causing students to drop out is needed. The current study uses the Kolbe A™ index in an attempt to gather some information regarding the students leaving the program and to use that information to propose changes in the learning environment to retain more of the qualified students who enter.

RESULTS FROM PREVIOUS STUDIES

Over a two-year period all computer science students taking a required upper division course in software engineering were given the Kolbe A™ index. The results from this study are shown in Table I below.

Specifically, of the 181 students studied over this period, 87 (48%) had *fact finder* as their primary mode of operation (i.e., the *fact finder* score was greater than or equal to the scores of any other mode), and 116 students (64%) had *fact finder* as their first or second predominate mode. Nearly half of the students (90 of the 181 students) had *follow thru* as their primary or secondary mode of operation. Also, and shown in this table, 54% of these students were insistent *fact finders* (meaning that their *fact finder* scores were seven or above) and 33% were insistent in the *follow thru* mode. Over sixty percent of the students were resistant to a *quick start* approach (meaning that their quick start scores were three or below) and thirty percent would resist an i*mplementor's* approach.

TABLE I
DISTRIBUTION OF CONATIVE TALENTS FOR UPPER DIVISION COMPUTER SCIENCE STUDENTS

	Fact Finder	Follow Thru	Quick Start	Imple-mentor
Prevent or Resist	2%	5%	62%	30%
Respond or Accommodate	44%	62%	33%	64%
Initiate or Insist	54%	33%	5%	6%

By way of comparison, we can see a great difference in the distribution of conative talents when individuals from another field are analyzed. Table II below shows the results for marketing managers based on a study done by the Kolbe Corporation. [18] Notice especially the difference in the *quick start* and *follow thru* columns. For the general population the expected value for both the *prevent/resist* and *initiate/insist* rows is 20% for each mode, and it is 60% for the *respond/accommodate* row. [13]

TABLE II
DISTRIBUTION OF CONATIVE TALENTS FOR MARKETING MANAGERS

	Fact Finder	Follow Thru	Quick Start	Imple-mentor
Prevent or Resist	14%	27%	11%	56%
Respond or Accommodate	55%	64%	44%	44%
Initiate or Insist	31%	9%	45%	0%

Not too surprisingly, most computer science instructors have Kolbe profiles that are almost identical to those of the typically successful students. A study of computer science instructors conducted at CSUN shows a conative profile that is not significantly different from that of the students they teach. This comparison of average scores for each mode is shown in Table III.

TABLE III
COMPARISON OF CONATIVE TALENTS BETWEEN INSTRUCTORS AND STUDENTS

	Fact Finder	Follow Thru	Quick Start	Imple-mentor
Instructors	6.8	6.2	3.4	3.4
Students	6.5	5.8	3.6	4.5

These instructors probably teach in similar ways to how they were taught and how they approach learning, and, therefore, tend to attract and retain students like themselves, a cycle that needs to be broken. This probably explains why very little active learning is done in computer science classes, and may indicate why some students (those with profiles

different from those of typical computer science students) may not feel that computer science is the right field for them. It may not be the material taught that discourages them, but rather the manner in which it is taught.

In another study that examined teamwork in software engineering classes [19], it was found that the most productive student teams were those with synergy, or a balance of conative talents. When a team is composed of like Kolbe characteristics, inertia develops. If a team is composed of too many *fact finders* and *follow thru* individuals, "analysis paralysis" will set in, as members spend endless hours probing, questioning, researching, and organizing. However, a shortage of *quick start* and *implementor* talents among computer science students makes forming synergistic teams difficult. This is a problem not only in the classroom but within the field of engineering and technology as well. [18] It could be a great benefit to the field if universities could produce computer science graduates with these other conative talents.

OBJECTIVES OF THE CURRENT RESEARCH

The goal of this research is to encourage a more diverse set of students to enter and continue in the field of computer science. In particular, this effort focuses on retention efforts for students with non-typical conative talents by engaging them to a greater extent in the types of problem solving activities in which they thrive. Although some might argue that the high dropout rate among computer science majors is not a problem, that only students who don't really belong in the program are being eliminated, the position here is that many students who could benefit from the program and who could in turn benefit the field are being discouraged from continuing. As stated in the article by Sorensen on women and computer science [20], there are losses to society "when the scientific and technological talents and experiences of women are not utilized." The same might be said for many other students who leave the program because it fails to accommodate their natural ways of learning.

CURRENT RESEARCH RESULTS

Past studies using the Kolbe A™ index focused on upper division students in the computer science program. One question that arose was whether the Kolbe profiles of students entering the program matched those of those nearer graduation. If the conative instincts of beginning students were significantly different, it would suggest that there might be a relationship between conative talents and students dropping out of the program.

The current research, therefore, used the Kolbe A™ index to determine the conative talents of students just entering the computer science program. We gave the Kolbe A™ index to students taking the first course in the computer science program last fall. A total of 58 entering students participated in the study. The distribution of their conative talents is shown in Table IV below.

It was expected that the results might show a great difference between the two groups. At first glance the results seem quite similar. The higher than expected degree of similarity between the groups probably means that students with significantly different conative profiles have already chosen a different field before entering college. However, if the two tables are compared using a chi-squared test, a statistically significant difference is observed ($p < 0.015$). The greatest difference seems to be in the *implementor* mode. Recall that an *implementor* is one who uses space and materials, and builds, constructs, and uses hands on equipment with ease.

TABLE IV
DISTRIBUTION OF CONATIVE TALENTS FOR FRESHMEN

	Fact Finder	Follow Thru	Quick Start	Imple-mentor
Prevent or Resist	7%	3%	71%	14%
Respond or Accommodate	53%	57%	22%	74%
Initiate or Insist	40%	40%	7%	12%

A chi-squared test on the *implementor* column of the tables shows a somewhat greater significant difference ($p < 0.01$). Conducting a t-test analysis of the raw *implementor* scores (i.e., comparing the average scores of the two groups) also shows a statistically significant difference ($p < 0.015$). Since the sample size is small and the probabilities are not extremely low, more study is required before reaching any definite conclusions. The fact that this experiment has been done only once at a single institution also argues for more investigation and study. The results do, however, raise some interesting questions for both student learning and retention that are worthy of consideration.

Comparing the results for the two groups of students, we see twice as many beginning computer science students as advanced students initiate in the *implementor* mode. That is, it seems that many more beginning students instinctively operate as *implementors*. Additionally, fewer than half as many are resistant to the *implementor* approach compared to the upper division students. This implies there would be typically less resistance to the ideas of *implementors* on a team of beginning students than there would be on an upper division team.

Although the results are not statistically significant, the beginning students seem to have much less *fact finder* instinct as well. The implication of this is that teams formed of such students would be less likely to be dominated by the ideas of the *fact finders*. Overall, it would be easier to form synergistic teams among these beginning students.

For this set of beginning students their performance in the first course in computer science was analyzed. In particular, the grades received by the *implementors* were compared with those of the other students. Surprisingly, there was no statistically significant difference in average grade, failure rate, or withdrawal rate as shown in Table V. (For the purpose of continuing in the computer science program, a C or better is required in the course. Both Ds and Fs are considered to be non-passing grades.)

TABLE V
COMPARISON OF IMPLEMENTORS WITH OTHER STUDENTS IN FIRST COMPUTER SCIENCE COURSE

	Implementors	Other Students	All Students
Average Grade	2.40	2.26	2.30
Lower Than Passing Grade	28.6%	24.4%	25%
Withdrawals from Course	14.3%	15.6%	15.4%

The conclusion one is tempted to draw is that if students with strong *implementor* instincts are leaving the program, it is not because they are having more difficulty with the courses than other students. It strengthens the idea that they may be changing their majors because they are not comfortable with the learning environment in the computer science program and that their natural ways of doing things are inconsistent with the ways in which they are required to operate. It is possible that by giving more opportunities for *implementors* to work in a "hands on" mode or to work on more concrete and less abstract tasks, they may find the computer science major more appealing. This is only conjecture at this point since there could be other reasons explaining why this study shows that fewer upper division are *implementors* than entering students.

POTENTIAL USES OF THE KOLBE A™ INDEX

The Kolbe A™ index can be used to help understand why some students don't do well or don't feel comfortable in the computer science program, as well as, to provide some guidance for improving the learning environment, and as an instrument to devise improved teaching strategies.

It can be used to point out potential conative deficiencies in the field that a particular student's talents might address. While individuals who are required to work in ways that are inconsistent with their conative talents can feel stressed, the field can benefit from individuals with diverse conative approaches. It would be a mistake to discourage students from majoring in computer science just because they don't have the "right" Kolbe profile. Also, a Kolbe analysis might help students who are interested in computing to pick among the various majors that might be available: computer science, computer engineering, software engineering, information technology, and information systems. They could choose the one that was the closest match to their conative talents.

In addition, the Kolbe results can be used to devise learning strategies that improve the learning experience and appeal to a wider variety of students. Previous studies [21] have shown how the knowledge of students' conative talents can be used to improve teaching techniques. For example, it was determined that typical computer science students are high in *fact finder* and *follow thru* instincts. These students tend to be very comfortable with the traditional lecture mode of instruction where they are able to gather information and have time to analyze it before acting on it. However, in many active learning approaches used today, students are asked to interact with others in the class or the instructor in an ad hoc manner, trying to find solutions to problems without a prior analysis of pertinent information. This experimental, open-ended approach to learning and problem solving is much more consistent with the *quick start* mode of operation and, therefore, might have its greatest appeal among students who don't typically major in computer science. While the *fact finder* and *follow thru* individuals might appreciate the empirical nature of certain active learning exercises, they tend to be uncomfortable if they haven't been allowed to prepare in advance, which is often the way such activities are conducted. The challenge is to provide a learning environment that is flexible enough to allow students to operate in a mode that is comfortable for them.

POSSIBLE ALTERNATIVE LEARNING STRATEGIES

Two questions arise. First, is it possible to devise teaching and learning strategies that are well suited to students with strong *implementor* instincts, and second, would it be a good idea to do so? We believe that the answer to both questions is "yes." While previous studies have concluded that adapting teaching strategies to the conative instincts of typical students in the major can enhance learning, it might be a mistake to focus solely on the typical students and ignore the others. Providing a variety of approaches and offering students some flexibility in the way they are allowed to learn might be the best solution.

For example, in order to address the natural learning approaches of the *implementor* student, courses would need to contain more "hands on" activities. It is likely that these students find the introductory computer science course to be too abstract, that there is too much emphasis on "planning" and not enough on "doing." The recommendation here would not be to eliminate the important design activities associated with learning from program, but to provide more opportunities for students to engage in more concrete activities.

A good example of this is the approach taken in teaching an introductory computer science course at the United States Military Academy at West Point. [22] Because of the school's mission, it is necessary for all the students at the Academy to be successful in an introductory computing course, regardless of their majors. And, the instructors do not have the option to "weed out" students from the IT program. As a continuous process of improving the course, the Electrical Engineering and Computer Science Department added the use robots as part of the active-learning environment they used to teach the course. The robots were used to help students learn fundamental computer programming concepts, and the instructors feel the short-term impact of the revised course was positive and substantial. They noticed that some students grasp the concepts more easily this way. Although there is no conative data on these students, it is likely that this approach would have been helpful to the *implementor* students. They stressed the fact that all students seemed to find this approach exciting.

The above is a good example of adapting the learning strategy to accommodate a wider range of students. In order to

appeal to the *implementor*, learning approaches must be developed which allow students an opportunity to build something concrete. In software development, many activities tend to be abstract and conceptual rather than real and tangible. While this is, of course, a necessity in software design, any opportunities given the *implementor* to construct or assemble tangible items might provide elements of comfort in the learning process. Creating diagrams, building prototypes, and utilizing tools are all activities that approach the *implementor's* needs with respect to their natural way of doing things. More work is needed to develop specific activities to this end.

CONCLUSIONS AND RECOMMENDATIONS

The current study has presented evidence that students entering the computer science program at CSUN possess a somewhat different distribution of conative talents from those who are nearer to graduation. In particular, it seems that students with strong *implementor* talents, as measured by the Kolbe A™ index, are the ones most likely to leave the program. The disadvantage of this is not only a loss of students in the program, but that the field is losing a significant source of conative talent. That talent is important when attempting to create effective and synergistic software development teams.

Although no specific recommendations can be made at the present time, it is suggested that computer science programs consider an approach in which students are given a greater opportunity for "hands on" activities, especially in early courses in the program. Many such approaches, such as the one described above in use at West Point, are consistent with efforts to provide "active learning" experiences. Efforts in this direction would appear to be advantageous from multiple points of view.

Future plans for this research effort at CSUN include three activities. First, there is an ongoing plan to use the Kolbe A™ index in order to increase the existing collection of data and to measure any changes in the conative profiles of incoming students over time. Second, the progress of incoming students will be tracked as they move through the program in an effort to analyze the relationship between their conative profiles and if or when they leave the program. Finally, an effort will be undertaken to develop, apply, and assess specific approaches to add "hands on" activities to the introductory computer science courses in the program.

One limitation of this study, of course, is that it has been done only at CSUN. It is difficult to know whether the results will generalize to other institutions. It is hoped that this effort will encourage others to conduct similar studies in order to add to the body of knowledge regarding the importance of conation to the issue of student retention in computer science.

REFERENCES

[1] Institutional Analysis & Planning, "Degree Completion Rates of the 1992 Year One Cohort," University of Waterloo, Ontario, Canada, http://www.analysis.uwaterloo.ca/HTML/kpiGrad2001.htm, 2001.

[2] Wambach, C., Mayer, A., Hatfield, J. and Franko, J., "General College Persisters and Leavers: A Comparative Study," University of Minnesota, 2003.

[3] "Towards a Quebec Innovation Policy," Conseil de la Science et de la Technologie, 1998.

[4] Boyle R., Carter J., and Clark M., "What Makes Them Succeed? Entry, progression and graduation in Computer Science*," Journal of Further and Higher Education*, vol. 26, no. 1, pp. 3-18, February 1, 2002.

[5] McCall, Alastair, "Improvements in Teaching Make a Sharp Difference," Times Newspapers, Ltd., London, 2003.

[6] McConnell, J. J., "Active Learning and Its Use in Computer Science," *SIGSCE Bulletin*, Special Issue, vol. 28, pp. 52-54, 1996.

[7] Bonwell, C. C., and Eison, J. A., "Active Learning: Creating Excitement in the Classroom," ASHE ERIC Higher Education Report No. 1, The George Washington University, 1991.

[8] Sridharan, Bhavani and Kinshuk, "Reusable active learning system for improving the knowledge retention and better knowledge management," *IEEE International Conference on Advanced Learning Technologies*, 2003.

[9] Timmerman, Brenda and Lingard, Robert, "Assessment of Active Learning with Upper Division Computer Science Students," *Proc. ASEE/IEEE Frontiers in Education Conference*, 2003.

[10] Lingard, R. and Berry, E., Improving Team Performance in Software Engineering, *Selected Papers from the 11th International Conference on College Teaching and Learning*, Chambers, C. (ed.), Florida Community College at Jacksonville, 2000.

[11] Kolbe, K., *Pure Instinct*, Times Books, New York, 1993.

[12] Reitan, R. M., and Wolfson, D., "Conation: A Neglected Aspect of Neuropsychological Functioning," *Archives of Clinical Neuropsychology*, vol. 15, pp. 443-453, 2000.

[13] *Kolbe Statistical Handbook*, Kolbe Corp., Phoenix, 2003.

[14] Loftus, Margaret, "Lending a Hand," *Prism*, ASEE, Vol. 14, No. 5, January 2005.

[15] Kolbe, Kathy, *The Conative Connection,* Addison-Wesley, Reading, MA, 1990.

[16] Lingard, Robert and Berry, Elizabeth, "Teaching Teamwork Skills in Software Engineering Based on an Understanding of Factors Affecting Group Performance," *Proc. ASEE/IEEE Frontiers in Education Conference*, 2002.

[17] Fitzpatrick, E., Askin, R., and Goldberg, J., "Using Student Conative Behaviors and Technical Skills to Form Effective Product Teams," *Proc. ASEE/IEEE Frontiers in Education Conference*, 2001.

[18] "Statistical Analysis of Kolbe Indexes," Kolbe Corp., Phoenix, 2000.

[19] Berry, Elizabeth and Lingard, Robert, "Teaching Communication and Teamwork in Engineering and Computer Science," *Proc. ASEE Annual Conference & Exposition*, 2001.

[20] Sorensen, Knut, "Gender and Inclusion Policies for the Information Society," Strategies of Inclusion: Gender and the Information Society, University of Edinburgh, Scotland, http://www.rcss.ed.ac.uk/sigis, 2004.

[21] Timmerman, Brenda, Lingard, Robert and Barnes, G. Michael, "Active Learning with Upper Division Computer Science Students," *Proc. ASEE/IEEE Frontiers in Education Conference*, 2001.

[22] Schumacher, Jerry, Welch, Don and Raymond, David, "Teaching Introductory Programming, Problem Solving and Information Technology with Robots at West Point," *Proc. ASEE/IEEE Frontiers in Education Conference*, 2001.