The Computing and Composition as an Integrated Subject in the Secondary School Curriculum

Fadi P. Deek, Robert S. Friedman, and Howard Kimmel

College of Computing Sciences
New Jersey Institute of Technology
Newark, NJ USA
973.596.2997 (O)
973.596.5777 (Fax)
Email: fadi.deek@njit.edu

Abstract

Many students enter college lacking basic problem solving and communication skills. The situation is even more exacerbated for students from urban high schools. We have disseminated a "computing and composition" approach, originally implemented in first-year college programming and English composition courses, to a high school curriculum in an urban setting. The Computing and Composition Project served students attending four Newark, New Jersey high schools, each serving populations with different profiles of academic performance. The instructional program included the development of a series of case studies based on the state high school science and mathematics content standards and the specific curriculum in place at the schools. These learning modules were designed to emphasize the skills required to solve problems and learn the syntax of the C++ programming language. Teamwork was an important part of this project.

An important component of the instructional method included a problem solving and program development process to assist beginning computer science students fulfill the complex, multiple tasks of programming. After working through the process, students will have produced a carefully designed and fully documented solution to a problem. In the composition component, the writing process was the basic procedure through which students gain skills in and an understanding of what is necessary to produce effective expository prose written to respond to specific problems. Measurements of success and problem areas in the implementation of this program will be presented and methods for overcoming obstacles in such programs suggested.

I. Introduction

A problem solving and program development process (Deek, 1997) offers computing instructors as well as instructors in other disciplines a comprehensive model that assists in synthesizing their teaching and learning objectives. This methodology,

adapted in various disciplines, is tailored to foster students' and instructors' taking into consideration the necessary tasks to be performed, skills that must be developed, and the expected learning outcomes. This has yielded enhanced interdepartmental cooperation (Deek, Deek, & Friedman, 1999; Friedman, Deek, & Deek, 2000). Examples of this cooperation have been taking place between introductory computing and English composition courses at the college level (Deek & Friedman, 2001) and have now been disseminated at the high school level, as we describe in this paper. As will be discussed below, each stage of this problem solving and program development process (Deek, 1997) is significantly analogous to the writing process, as described by leading compositionists of a cognitive bent, such as Flower and Hayes (1977 & 1980).

Our pedagogy is based in a pragmatic view of the shared (NJIT) and specific contexts (a Computer and Information Science instructor and an English instructor) in which we have been operating. Our intention is to exploit the commonalties in our disciplinary approaches while remaining cognizant of ideologies and pedagogies our academic contexts would allow. We decided to use the cognitive process analogies we could tease out of our domains while being aware of the competing influences of early and current approaches to writing and programming instruction. Specifically, while we fully understand that the term "writing process" must be complicated beyond a facile, linear concept, the plethora of English handbooks proffered by the dominant publishers hasn't displaced the cognitivist view of composition.

II. Computing and Composition

Problem solving and programming activities are closely intertwined in the introductory course on computing taken by all first year students at NJIT. In this course, each class is goal directed, as it is designed around a problem-solving experience that takes into consideration the programming material covered in that session. Problem-solving heuristics and program development tasks are integrated and introduced as series of activities requiring specific knowledge and skills that must be acquired and mastered by the students.

The problem solving and program development process, consisting of formulating the problem, planning the solution, designing the solution, translating the solution, testing the solution, and delivering the solution, begins with the students, and the instructor acting as facilitator, formulating the problem. This requires the students' understanding the question as well as identifying the problem's facts. Planning the solution proceeds with the development of an appropriate strategy based on various solution alternatives, and simplifying the problem into smaller sub-problems that can be more easily addressed. The solution design then begins to take shape by organizing, refining and specifying the various components of the solution, and defining its algorithmic specification. Each of the tasks in problem formulation, planning, and design are problem solving activities that require specific knowledge and skills that students need to develop, as students will be using them continuously throughout the problem. The process continues with implementing the algorithmic solution into programming language code. This is done by translating the details of design into language syntax that is tested and then executed on the computer to produce the result. The product of these activities is a deliverable that

documents the work performed at each stage of the process. The activities of this process are all carried out by the students, starting with defining the problem through the delivery of the complete solution. The instructor's role is primarily focused on guiding the students as they progress from one stage into the other.

In the composition course, the cognitive approach to the "writing process" is one perspective that writing teachers introduce in an effort for students to gain skills in and an understanding of what is necessary to produce effective expository prose written to respond to both specific problems and more generalizable issues. Faigley (1986) explains that writing has been viewed primarily from an expressive, cognitive, or social theoretical perspective. We have chosen to work from a cognitive, goal-directed and pragmatic point of view for the reasons stated above. Therefore, following Flowers and Hayes, but not abandoning opportunities to benefit from the reciprocity and recursive nature of writing, we encourage students to engage a process of composition that is analogous to their efforts in their computing class, by the planning, organization, drafting and revising of multiple versions of an essay and other writing assignments based on themes or ideas generated by their work in the computing class, as well as readings and discussion concerning computing in particular and technology as it affects their socio-cultural worlds.

Like the problem solving and program development process, each of these four categories contains processes and sub-goals. While planning any writing assignment, students engage in exercises familiar to this audience that result in the narrowing down of incipient ideas and amorphous possibilities (Flower & Hayes, 1980). Brainstorming and free writing are popular techniques used in the planning stage, but they are followed by the breaking down of ideas into components in order to test possible arrangements of information into sections of drafts as an organizational strategy. The analytic processes of peer review give each member of the class access to a potentially cohesive discourse community comprised of both computing and composition disciplines. Each member of a problem-solving team reads other students' essays in order to locate theses and supporting evidence, the presence of an organizational strategy and the coherence of ideas. Students revise drafts based on their peers' comments and submit a final draft for a grade. In the terminology of the problem solving and program development process, students work toward understanding the writing problem they've been issued (Problem Formulation). They then generate ideas (Solution Planning), organize those ideas (Solution Design), draft those ideas in sentence and paragraph form (Solution Translation), test the efficacy of those ideas with others (Solution Testing), and finalize those ideas in the form of a coherent essay (Solution Delivery). We explicitly engage students in discussions that use this terminology as a way to bridge the concepts of the process that results in an executable program in the computing class with a critical investigation of that process and its applicability to pragmatic problem solving in other domains. As will be described shortly, one combined writing/composition assignment on the development of a citation translation program demonstrates the analogies of process, design and application between the two domains. Before the discussion of particular examples, however, we describe the problem-solving methodology in greater detail.

III. Process Integration

Problem formulation requires knowledge about the domain or subject, problem modeling skills, and communications skills. The identification of knowledge through information gathering techniques and the representation of this knowledge are the primary requirements of this first stage in the problem solving and program development process. Problem formulation, combined with domain knowledge, leads students to comprehend the problem at hand, acquire new, related knowledge, and develop important cognitive processes (Bloom, 1956). The underlying cognitive structures are the knowledge acquisition processes that are used to acquire and integrate knowledge (Sternberg, 1985) for subsequent planning, design, and implementation activities. The cognitive outcome takes the form of verbal information that confirms problem understanding and identification of facts essential at this stage (Gagne 1985).

In both freshman composition and introductory computing, instructors stress the importance of students verbalizing what they have come to know, confirming to themselves and others that they understand the problem at hand and can identify its essential components. This process, obviously applicable to the computing classroom, is analogous to the brainstorming and clustering activities that begin our composition process. Here, students consider not only the writing problem that has been set out, but also as many contextual and tangentially related items of information as possible. Our goal is to situate the computing problem in a discursive environment, one larger and more complex than simply following the steps necessary to arrive at an executable program. The products of these cognitive activities are sets of words that are investigated for potential inclusion as either thematic cores or informational subsets (Lundsford, 1985).

Solution planning requires domain knowledge, problem-specific knowledge, and strategic skills. Planning requires general problem solving strategies to discover and assess solution alternatives and produce a plan for the problem. The application of knowledge, which involves problem analysis and decomposition or breaking the problem into component parts (Bloom, 1956), is the key requirement of this stage. The most relevant cognitive structure is the performance component that directs the solution planning, alternative evaluation and the problem decomposition process (Sternberg, 1985). An important cognitive outcome of this stage are the intellectual skills that demonstrate the ability to apply knowledge (Gagne, 1985), which confirms its understanding.

This stage again supports the goals of the composition course by accentuating the value of verbal and written expression of problem analysis. Without such discussion, student writer/programmers most readily avoid the complex inter-relationships within the contexts of computer science and programming languages, information systems, and human-computer interaction, particularly as it pertains to usability. Moreover, the cognitive skills necessary for the problem decomposition process are analogous to those that support paragraph formation and development, as the complexities of decomposition reappear as writers select, rearrange and reorder the sentences that ultimately comprise a particular paragraph (Flower & Hayes, 1980).

Solution design requires the same knowledge and skills as the planning stage. The major cognitive activity at the solution design stage is synthesis, or the reintegration of interrelated components into a coherent whole. It refines and rearranges components when necessary, establishing a hierarchy that organizes the problem into its parts and sub-parts, and producing a new and well-organized whole as a viable solution to the problem (Bloom, 1956). The most relevant cognitive structure is still the performance component, which in addition to directing the decomposition process, is concerned with the identification, selection, organization, and sequencing of these tasks (Sternberg, 1985). An important cognitive outcome of this stage is strategic ability demonstrated by the ability to specify detailed solution design (Gagne, 1985).

Again, this outlining and reorganization of information is practiced in both classrooms, as composition concerns of coherent organization and development of ideas are realized in a logically determined working outline. Realizing that outlining has lost its luster in current composition theory, we continue to employ this organizational technique for several specific reasons. Foremost, the act of outlining reinforces the concept of cohesiveness as it offers writers another opportunity to consider reorganizing, revising and deleting the basic ideas the writer intends to communicate. Whereas in the computing classroom sub-goals are translated into algorithms, in the composition classroom we point out that topic sentences in paragraphs serve to guide an audience toward the specific mapping of organizational strategies that lead readers through increasingly credible steps toward a conclusion. Following that train of thought, we try to impress upon our student writers the point that the more sophisticated the organizational structure and strategy, the more important transitions bridging ideas between paragraphs become. Effective transitions result from a clear understanding of the explicit and implicit relationships of related ideas and components. In the computing course, where a detailed solution design is the outcome sought for this stage, sentences that demonstrate the design framework are the goals in the writing classroom.

Solution translation requires, in addition to the knowledge and skills of the previous stage, additional organizational, syntactical, semantic and pragmatic skills. There are two major cognitive activities at the solution translation: synthesis and organization. This requires the ability to maintain and access previously generated and organized knowledge (Bloom, 1956). The relevant cognitive structures are the performance component, which is here concerned with the execution of a planned design and the memory component (Sternberg, 1985), essential for syntactical and semantic information. An important cognitive outcome is the intellectual skill required for logical or deductive ability (Gagne, 1985), demonstrated by the diagnostic analysis of programming errors.

The deductive and inductive critical thinking problems that students work through in the composition process are parallel to the activities of solution translation in computing. In composition, the cohesion of paragraphs that contain discrete units of information is refined to indicate the logical connection of ideas. This is demonstrated in rhetorical schema based in chronology, spatial orientation or hierarchies of importance that help advance the reader through the essay by stressing the tacit and explicit

transitional devices writers employ (Vatz, 1973). But these critical thinking problems, two of which are described below, carry more currency in the interdisciplinary context in which we work. Fortune and Kalmbach's (1999) introductory essay in *Computers and Composition* finds that object-oriented programming brings with it an opportunity to begin to shift the focus from the logic and syntax of the code to a consideration of the larger structural relationships where function and rhetoric are articulated. We believe that the critical thinking processes necessary to follow the "strict logic" of structured programming language become far more complicated when situated in an environment where the context of the coding work is continuously infused into discursive activity that challenges linearity, even as it applies to code. The rationale and outcomes of the infusion of HTML coding into the composition classroom has also been described: If "students create web pages in their writing classes," claim Fortune and Kalmbach, "the rhetoric of those pages and the rhetoric of their code interact in a way that positions students to appreciate not just each rhetoric at the same time but the relationship between rhetorics as well" (322). This is essentially the purpose of our work, but brought into a programming environment that demands the accretion, accrual and reassessment of each step of the programming process we employ.

Solution testing requires similar cognitive skills to the previous stage, but with the addition of metacognitive skills. The major cognitive activities at the solution testing stage are critical analysis, retrospection, and evaluation (Bloom, 1956). The relevant cognitive structure is the metacognitive component, concerned with monitoring the thinking process and evaluating the solution (Sternberg, 1985). An important cognitive outcome is a self-critical attitude (Gagne, 1985), the ability to critically assess one's own thought processes as well as one's own intellectual creations.

Through classroom discussions regarding programming activity and peer reviews of descriptive essays concerning or analogous to the activities underway in the computing course, students are never far away from the explicit connections between the two classes. In a composition environment, peer review accomplishes many of the same goals of solution testing. The solution being "tested" in peer review is the viability and credibility of the conclusion -- whether the thesis has been supported and to what extent. To arrive at this point, however, students must converse with one another so that reader/reviewers can articulate not only whether the writer has been generally successful, but more importantly, explain to the writer exactly how they understand the essay to cohere, how it is organized, how effective are the developmental sentences and transitional markers and how well the thesis of the essay has been supported.

Solution delivery also requires organizational and communication skills. The problem solving and program development cognitive activities have been completed by this stage so no further problem transformations are involved. However, the information produced during the course of previous stages has to be organized, presented and possibly disseminated. The documentation of the solution strategy, code, and test results is the first task. In the case of large or group projects, the results may have to be orally presented. Moreover, for some projects it may be appropriate to disseminate the information to an appropriate community of interest, a task made both more feasible and

more important by the availability of Internet technology. And it is here where group presentations of computing projects come into play in the composition course. In our sections of introductory computing and composition, students combine their computer science work with their English class readings through oral presentations that reinforce the communication components of a beginning programming class but also explain the analogies uncovered between communications and computing, even at this fundamental level.

IV. Implementation and Evaluation of the Computing and Composition Project

During the 2000-2001 academic year, the authors received funding from the Lucent Technologies Foundation to implement a version of the computing and composition project in four Newark, New Jersey high schools. 37 students attending Barringer, Science, Technology and University High Schools participated in the project, which was designed to meet the following goals and objectives:

1. Provide a year-long instructional and mentoring program in problem solving and programming in C++ language.
2. Introduce Newark High School students to freshman level computing and composition work.
3. Demonstrate the similarities and connections between problem solving in a programming and a writing environment.
4. Develop video instructional materials to be piloted at the four participating high schools (with a goal of disseminating to other Newark high schools in future academic years.)
5. Disseminate a series of case studies to which can be applied the problem solving methodology used in C++ instruction.

The project provided a yearlong instructional and mentoring program in problem solving, programming and composition to 37 students in grades nine through twelve. High school teachers from each of the four schools were selected to act as on-going coordinators for the project at their respective schools. A training program for the high school teachers in problem solving and programming was provided to prepare them for the implementation of the project, as well as for its continuation beyond the period of funding. Two NJIT instructors were involved in the delivery of computing and composition content for students at the four schools. To assist the program instructors and the high school teachers, NJIT graduate students were selected to work as teaching assistants at each of the schools. Instruction took place at the respective high schools, with periodic sessions provided by NJIT via teleconferencing.

Assignments

Students engaged in several programming assignments, all of which had composition assignments that were derived from the pedagogical objectives of the programming work. With the overall theme of problem-solving as a process of subdivision of large problems into smaller, more manageable ones, discussion of the central role of algorithms in programming was made more concrete by introducing the concept of algorithms as processes as commonplace as traveling to school each day. Students were asked to write an essay that relates precisely, and in chronological order,

the actions and steps necessary, from the moment they wake up until they walk through the school door, for them to arrive at school on time and prepared to work.

A subsequent programming assignment created a scenario in which the students had hired a construction company to build two approximately perpendicular roads (within 10 degrees plus or minus) through the center of the town. The roads were to be a minimum of one mile in length and not greater than a mile and a quarter, and they would divide the town into four distinct quadrants. Planning should be done for each quadrant as to the location of shops, schools, housing, police, hospitals, etc. As the construction company would need to create these two roads, surveyors would have to take readings of height elevations every 100 feet. Students, as surveyors, were instructed to take these readings by starting at the outskirts of the town and giving the initial reading a value of zero. All successive readings were to be taken as a plus or minus value in relation to the initial value (integer values only). Both roads were to start exactly 2,700 feet from the precise center of town. One road should extend to the western and eastern sides, and the other road to the northern and southern sides of town.

Students were asked to write a program that would allow the engineer to enter the elevation reading that was taken at each foot marker. A sentinel would be used to indicate when all of the readings have been entered. The student would determine the following for both roads:

- The length
- The highest and lowest points of elevation
- The total number of marker points that are higher than the starting point
- The total number of marker points that are lower than the starting point
- The total number of marker points that are at the same level as the starting point

Additionally, the engineer would need to know whether the ends of each segment of each of the roads are at the same level, and if not, what end (N, S, E, or W), is higher than another. The program students were to create were to have the necessary data structures to handle all of the information. The data structures should be initialized to some known values. Students were instructed to do error checking wherever appropriate. All prompts to the user should be unambiguous. There should be no global variables. Proper documentation should precede the entire program and each of the modules used. Side comments should be used where necessary. All modules should perform one function well.

The companion writing assignment began by students demonstrating that they completely understood what they were to program, continued by having them explain what the configuration of their town is and how they arrived at that configuration, and concluded with a brief proposal memo to the City Manager for the construction of the town to their specifications. After defining such terms as “perpendicular,” “integer,” and global variable, students were instructed to write one paragraph that described the problem they were trying to solve (problem description), then the steps it would take to solve it (a verbal algorithm). Once they demonstrated that they understood the

programming assignment, their next writing assignment was for them to describe the content of each of the quadrants and offer a rationale as to why different buildings, parks and homes were placed into each of their town's four quadrants.

Evaluation of the program

Evaluation instruments were developed for use during the course of the project, and at its conclusion. In addition, a project evaluator began conducting initial interviews with project teachers and teaching assistants. Soon after the project was launched, written surveys of high school teachers and teaching assistants were administered to ascertain the progress of the project. Both groups reported satisfaction with project activities but pointed out some difficulties that required attention. One such issue was the differences in the level of mathematics being studied by the students. The project team determined that adjustments were needed in the programming assignments to create varying levels of difficulty in the mathematics required for them. This allowed for the varied backgrounds of the students to be served accordingly.

Three evaluation instruments were developed for use at the end of the project, two for the teachers and one for the students. The teachers responded to questions concerning their perceived outcomes for the students as a result of their participation in the program, and a self-reporting instrument on their perceived preparedness for teaching problem solving and programming in the near future. The students completed a post-course survey to ascertain their perceptions of what they have learned and gained from the course material, as well as their interests in the content of the course. It should be noted that one high school, Science High School, had 13 students participating in the course and responding to the survey. The other three high schools had fewer students (between 7 and 12) in each school participating in the project, and not all the students responded to all the questions.

Based on the overall analysis of responses to the surveys, we believe the program was successful in meeting most of the project objectives outlined for students and teachers. These surveys indicated that the students were developing their problem solving and cognitive skills, learning programming skills, teamwork skills, and, for the most part, improving their perceptions and attitudes towards careers in science and engineering. However, while we were successful in achieving most of our objectives, problems were encountered that required modifications in our plan of operation. The logistics of the implementation had to be modified to fit within the operations of the Newark Public Schools. In the original plan, there was to be a "lecture" by one of the NJIT instructors to students, simultaneously, at all four high schools through video conferencing, following by "hands-on" laboratory instruction by the high school teachers, supported by NJIT teaching assistants. Unfortunately, two things occurred which made this mode of instruction infeasible:

1- The equipment at the four schools was not functioning and could not be repaired in time for the project start date; and
2- A common schedule for the students at all four schools could not be developed.

The mode of instruction varied among the schools, depending upon the high school teacher's comfort level with the course content. This was expected to have an

impact on evaluation results among the schools, as was indeed shown in the responses to the end of course surveys and informal interviews of high school teachers and teaching assistants. For example, an interview with the Newark District's Language Arts specialist indicated that the English teachers in one high school (Science High School) did not participate in the project, and the students were found to be struggling with the writing component. The Language Arts specialist did not become involved in the project until March, at which time students began to receive some guidance with the composition component other than that provided by the NJIT instructor and teaching assistants. But the high school English teachers remained disconnected from the project and the students dealt with the writing component as best as they could without their direct formal guidance. This required additional effort of the NJIT English instructor to provide framework for the students' writing about the solving of problems, and framing the questions that should be asked by the teachers for each module.

There was more participation by English teachers in the other high schools, and the computer science teachers from two of the schools reported positive impacts on student writing during the course. For example, to the question, "How has the Computers and Composition Course impacted upon your students academically?" one teacher wrote: "Although I am not a Language Arts teacher, it seems as if the student's writing is more fluid. … "Their work seems more organized." Another teacher wrote: "They have learned about the 'real world' requirements to document the project they are working on!" and "I believe the students done more writing and lived with the expectation that writing is part of the computer programming environment."

For the question: "Regarding the writing strategies introduced to your students through the course, which ones do you believe have been most effective?" teachers' responses included: "The first where they had to list what they did and were forced to write in a logical sequential manner." And, "I believe each writing assignment had its place versus one being better than the other."

The student survey included three questions in the course outcome section related to the impact of the course on their writing and communication skills. Two statements on the survey indicated that the course improved students' writing skills and motivation to write better. A third statement indicated that the course had no impact on the students' writing skills. Students were asked whether they agree or disagree with these statements. The results for the three schools with less than ten students each were inconclusive as the data were scattered. However, for Science High School, with 13 students, it was clear that students in general did not believe the course had any impact on their writing skills. This was consistent with the comments of the District Language Arts specialist who found that the English teachers were not well integrated into the project by working with the students on their writing skills within the context of the course. The students were essentially left on their own. It is apparent that future activities will require more initial contact with and greater involvement of the High School English teachers, as well as the High School Computer Science teachers. This must include professional development activities for both components of the project. The idea is to have the English teachers act as a critical part of the course and thus participate in project activities.

Other areas of the student survey that provided useful input are interest, difficulty, understanding, and learning. Three questions focused on student interest in the course, in programming, and in problem solving. The results were inconclusive as the data was scattered for each school between "agree" and "disagree." The statements concerned with the difficulty of the course were more informative. It was found that students enrolled in the higher-level math courses, such as calculus, were less likely to find the course and the problems difficult, whereas students taking the lower level math courses, such as algebra and trigonometry, were more likely to find the course difficult. Since the problem sets focused on mathematical problems, this should not be surprising. Early in the project, it was determined that some students were having difficulties with the mathematics in the case studies, and as a result, the problems were modified so that several "layers" were created for each problem, providing the students with a scaffolding approach to learning. The students were then better able to pace their problem understanding and solution development.

A similar pattern was found with the two statements related to understanding the concepts and course content, and the four statements related to student perception of their learning. The student responses to the statements related to understanding indicated that overall for each school students believed that as a result of participating in the course, they

- Gained a good understanding of the concepts covered; and
- Gained additional understanding of problem solving and programming by doing the assignments.
-

Four statements were selected that reflected the students' perceptions of learning as a result of participation in the course. Overall, students reported that they learned a great deal from the assignments and from the problem solving/programming case studies. Students were unanimous in their opinions that solving the problems was a good learning experience and the class was a good learning experience.

The high school computer science teachers also believed that the course was an excellent learning experience for the students and would like to have it continued. The teachers were especially enthusiastic about the case studies/course materials. One teacher thought they were excellent materials, while another teacher wrote, "The material has been presented in a very understandable and thorough manner."

In summary, the project was successful in meeting its objectives of

- Developing a set of case studies for teaching problem solving and programming.
- Training teachers in the methodology of problem solving within the context of the fundamental course on programming.
- Developing students' problem solving and cognitive skills; while exposing them to programming, computing applications, and information technology.

V. Conclusion

We have presented a model for interdisciplinary instruction and cooperation in computing and composition based on identified commonalties in the processes of problem solving, programming, and composition. This model can be extrapolated to other learning environments where a linear, yet recursive approach to teaching and learning is applicable. However, there are basic issues for consideration given the approach presented. From an instruction viewpoint and based on our experience, this model has led to significant restructuring of course content, requirements, and curricula. For one, we have found the traditional semester to be restricting in terms of the time available to cover all the content that the computing course traditionally offered. With a new emphasis on documenting the steps of the problem-solving methodology, particularly the final stage where delivery of the solution occurs in both the form of an executable program and a descriptive text, the domain of composition blurs into that of computing. The obverse applies as well. Writing, Speaking, Thinking was originally designed so that student writing was derivative of the scientifically-oriented readings appearing in anthologies with a historical and social science bent. With a new focus on computing science, specifically the pragmatic work of problem solving with and through programming, the "English" instructor's purview increases to where one must understand the basic properties of programming and systems design.

Acknowledgements

This work is supported, in part, by a grant from the Lucent Technologies Foundation "Preparing Newark's Youth for Global Skills" Grant Program.

References

Bloom, Benjamin. (Ed.) (1956). *Taxonomy of Educational Objectives, Handbook I: Cognitive Domain*, New York: McKay, 1956.

Deek, Fadi & Friedman, Robert. (2001). Computing and Composition: Common Skills, Common Process, *Journal of Computer Science Education* - ISTE SIGCS, *15* (2).

Deek, Fadi, Deek, Maura, & Friedman, Robert. (1999). The virtual classroom experience: Viewpoints from computing and humanities, *Journal of Interactive Learning Environments, 7* (2 & 3), pp. 113-136.

Deek, Fadi. (1997). *An Integrated Environment for Problem Solving and Program Development*, Unpublished Ph.D. Dissertation, New Jersey Institute of Technology.

Faigley, Lester. (1986). Competing theories of process: A critique and a proposal. *College English, 48* (6), 527-542.

Flower, Linda & Hayes, John. (1980). Identifying the organization of writing processes. In Gregg, Lee & Steinberg, Esther. (Eds.), *Cognitive Processes in Writing*. Hillsdale, NJ: Lawrence Erlbaum, pp 3-30.

Flower, Linda & Hayes, John. (1977). Problem-solving strategies and the writing process. *College English*, 39 (4), 449-461.

Fortune, Ron and Kalmbach, James. (1999). Letter from the guest editors. *Computers and Composition 16* (3), 319-324.

Friedman, Robert; Deek, Fadi; & Deek, Maura. (2000). Bridging technology and pedagogy: Interdisciplinary computing and composition. *WebNet Journal: Internet Technologies, Applications and Issues, 2* (1), pp. 60-67.

Gagne, Robert. (1985). *The Conditions of Learning*, Fourth edition, New York: Holt, Rinehart and Winston.

Sternberg, Robert. (1985). *Beyond IQ: A Triarchic Theory of Human Intelligence*, Cambridge, Massachusetts: Cambridge University Press.

Vatz, Richard. (1973). The myth of the rhetorical situation. *Philosophy and Rhetoric, 6*, pp. 154-161.

FADI P. DEEK
Fadi P. Deek is Associate Dean of the College of Computing Sciences, Director of the Information Technology Program and Associate Professor of Information Systems at NJIT. His current research includes learning technologies, software engineering, programming environments, problem solving/cognition/learning theory, and computer science education. He is currently Project Director for the $2.5 million N. J. Information-Technology Opportunities for the Workforce, Education and Research Project.

ROBERT S. FRIEDMAN
Robert S. Friedman is a Research Professor in the Information Technology Program. His current research includes computer mediated communication systems, asynchronous learning systems design, multimedia technologies, and communication theory. He is currently Investigator and Project Manager for the $2.5 million N. J. Information-Technology Opportunities for the Workforce, Education and Research Project.

HOWARD KIMMEL
Howard Kimmel, Professor of Chemistry and Assistant Vice President for Academic Affairs at New Jersey Institute of Technology, has spent the past twenty-five years designing and implementing professional development programs and curricula for K-12 teachers in science and technology. At the college level, he collaborates on projects exploring teaching methodologies and assessment strategies in first year college courses in the sciences, engineering, and computer science.