PROBLEM SOLVING AND COGNITIVE FOUNDATIONS FOR PROGRAM DEVELOPMENT: AN INTEGRATED MODEL

Fadi P. Deek, James A. McHugh

ABSTRACT
This paper addresses the interdependence among problem solving, cognition, and program development. The goal is to present a coherent and complete theoretical model which can serve as a basis for program development environments. To determine the type of support such environments should provide we first identify the actual difficulties involved in learning programming. We then synthesize a common model for problem solving based on existing methodologies. We next examine the problem solving tasks specific to program development, identifying how to adapt or enhance the general common model to the area of program development. Finally, we determine the cognitive science and learning theory relevant to problem solving identifying for each task of the common model the appropriate cognitive techniques required, thus defining a *Dual Common Model* which integrates the problem solving methodology and program development tasks with the cognitive knowledge and skills needed at each stage of the process.

KEYWORDS
Problem solving, program development, cognitive model, cognition, human information processing, human learning

PROBLEM SOLVING AND PROGRAM DEVELOPMENT

It is now widely agreed that the ability to write programs, and the difficulties encountered therein, extends far beyond learning the syntax of a specific programming language. There are three kinds of challenges students face when learning the tasks of program development: deficiencies in problem solving strategies and tactical knowledge; misconceptions about syntax, semantics, and pragmatics of language constructs; and ineffective pedagogy of programming instruction. While not minimizing the importance of syntactical issues [Shneiderman 1980; Rogalski and Samurcay 1990, 1993], research clearly indicates that the most fundamental obstacles to learning programming are related to its problem solving character [Mayer 1981; Perkins and Martin 1986; Perkins, Hancock, Hobbs, Martin, and Simmons 1986; Spohrer and Soloway 1986; Rist 1986; Johnson 1990; Lee and Pennington 1993; Ebrahimi 1994; Ennis 1994; Weidenbeck and Scholtz 1996.

Deek (1997) reviewed twelve different models of problem solving developed this century. Two of the earliest methods for problem solving were given by Dewey (1910) and Wallas (1926), and represent opposite approaches. Dewey's approach essentially articulates the scientific method for problem solving, while Wallas' approach represents the non-systematic, creative view of problem solving. Subsequent models combined elements of both the scientific and the creative approaches. Principal among these is Polya's famous work on problem solving. The Polya model (1945 and 1962) elaborately specifies a problem solving method supported with examples and documented in a series of books. Independently, Johnson's model (1955) refers to Wallas, while Kingsley and Garry's model (1957) elaborated on Dewey. A separate, but similar, model was presented by Osborn (1953) and Parnes (1967). Neither Johnson nor Kingsley and Garry introduced significant development over their

predecessors. Despite the independence of these three methods, they are basically consistent in their approach, an important indication of the stability of the methodology over time. A different approach was introduced by Simon (1960) who viewed the process as a collection of four cognitive abilities: intelligence, design, choices and implementation. More recent methods were developed to provide mathematics, science and engineering students with a method for problem solving. Generally, these models divided the problem solving process into a more finely specified process than the earlier methods. Notable among these models is the work of Rubinstein (1975), who introduces an element of reservation. One such reservation is at the problem understanding stage where he looks at possible solutions before finalizing the problem statement; there is a similar withholding of commitment at the final problem solution. Otherwise his method represents the standard view. Other popular methods are Stepien, Gallagher, and Workman (1993), Etter's (1995), Meier, Hovde and Meier (1996), and Hartman's (1996) who presented models that basically follow the Polya model without any radical changes.

One can identify a common integrated model for problem solving based on the models just reviewed. Although the general form of the methodology is clear from the review, it will be beneficial to carefully synthesize these methods into a common model for problem solving. The goal is to capture the essential features of these problem-solving approaches, and to provide an established, recognized framework which can serve as the basis for a problem solving method that we will later adapt to the area of program development. It is clear that an integrated view of problem solving includes the following: understanding and defining the problem, developing a plan for solving the problem, designing and implementing the plan to produce a solution, and verifying and presenting the results. A synthetic view of the tasks involved by these objectives follows. Later on, a more comprehensive model that explicitly addresses the cognitive and program development aspects of the process will be defined.

Programmers must develop skills which include: learning the language, composing new programs, comprehending, reusing and integrating existing programs, debugging, testing, modifying, and documenting the programs they write. All of these skills are essential to carry out the tasks of program development. These are cognitive tasks related to language and require knowledge of the syntax and semantics of the programming language [Shneiderman 1980; Rogalski and Samurcay 1990, 1993]. Other cognitive tasks, related to problem solving, such as problem understanding, analysis, and design of the solution, require domain, strategic and tactical knowledge, as well as practical knowledge of the programming language [Wirth 1971; Pennington and Grabowski 1990].

A DUAL COMMON MODEL FOR PROBLEM SOLVING AND PROGRAM DEVELOPMENT

In this section, our common model of problem solving and the tasks of program development will be joined with the work of Bloom (1956) on cognition, Sternberg (1985) on human information-processing, and Gagne (1985) on human learning to create a Dual Common Model for Problem Solving and Program Development supported by the necessary knowledge and skills that must be developed and the expected tasks that must be performed at each stage of the process. This dual model (called dual because it explicitly brings the problem solving method/program development tasks and cognition into one model) can form the basis for specifying environments for problem solving and program development. The remaining parts of this section describe the six stages of the model and *Figure* 1 illustrates the cognitive system of the problem solving and program development.

Formulating the Problem

We identify three activities for this stage: *preliminary problem description*, *preliminary mental model*, and *structured problem representation*. Domain knowledge, problem modeling and communication skills are required to carry out these activities. Identification of knowledge through information gathering methods and representation of this knowledge are primary requirements of the problem solving process. From the viewpoint of the cognitive model, the combination of this information with

other knowledge such as domain knowledge, leads to comprehension of the problem question, a major objective of this stage [Bloom 1956]. In terms of cognitive structures, knowledge acquisition processes are used to acquire, recall, and integrate the information and knowledge needed to devise and implement a solution [Sternberg 1985]. In terms of cognitive outcomes, verbal information that confirms problem understanding and identifies facts is an important result of this stage [Gagne 1985.

Planning the Solution
We identify three activities for this stage: *strategy discovery*, *goal decomposition*, and *data modeling*. Domain, problem, and strategic knowledge are required to carry out these activities. From a process viewpoint, the major cognitive activities at this stage are the application of knowledge, and problem analysis and decomposition. Understanding of knowledge is demonstrated by the appropriate application of that knowledge. The use of knowledge, facts, and the application of concepts, theories or principles to plan a solution are in turn demonstrated by outlining the steps necessary to reach a solution by solving simpler, related problem, or by drawing charts and graphs which visually depict a solution. The cognitive processes of analysis and decomposition, which involve breaking the problem into component parts, entail identifying and establishing a hierarchy which organizes the problem into its parts and sub-parts [Bloom 1956]. The most relevant cognitive structure is the performance component which directs the solution planning and the problem decomposition process [Sternberg 1985]. The important cognitive outcomes of this stage include intellectual skills, which demonstrate the ability to apply knowledge and outline a detailed plan for a solution [Gagne 1985].

Designing the Solution
We identify three activities for this stage: *Organization and refinement*, *data/function specification*, and *module logic specification*. The same cognitive knowledge and skills are required for design as for planning. From a process viewpoint, the major cognitive activity at this stage is synthesis, which is concerned with the reintegration of interrelated components into a coherent whole, rearranging when necessary, establishing relationships, and producing a new and well-organized whole as a viable solution to the problem [Bloom 1956]. The most relevant cognitive structure is the performance component which in addition to directing the decomposition process is concerned with the identification and selection of tasks; and the organization, sequencing and execution of these tasks [Sternberg 1985]. The important cognitive outcomes of this stage include cognitive strategies which demonstrate the ability to carry out the transformation of previously developed plan for a solution into an actual solution [Gagne 1985].

Translation
We identify three activities for this stage: *Implementation*, *integration*, *and diagnosis* (of errors). The cognitive knowledge and skills required for translation include those for design, but supplemented by organizational, syntactical, semantic, and pragmatic skills. From a process viewpoint, the major cognitive activities at this stage are the application of knowledge, synthesis, and organization [Bloom 1956]. Application in this stage refers to the pragmatic ability to use the general knowledge of language syntax and semantics to implement a coded solution. Synthesis enters in two ways: first, with respect to the integration of existing software components into the solution, and secondly, with respect to the piecemeal integration of the modules under development. The relevant cognitive structures are the knowledge acquisition and the performance components [Sternberg 1985]. The knowledge acquisition component is concerned primarily with determining relevant language features and integrating previously identified partial solutions. As usual, the performance component involves the organization and execution of these tasks. The important cognitive outcomes of this stage include intellectual skills demonstrated by the ability to apply knowledge the diagnostic analysis of errors [Gagne 1985].

Testing
We identify three activities for this stage: *Critical analysis*, *evaluation*, and *revision*. The knowledge and skills for this stage are the same as for the previous translation stage, with the important exception that organization skills that predominate there are now replaced by metacognitive skills. From a

process viewpoint, the major cognitive activities at this stage are analysis, evaluation, and metacognition [Bloom 1956]. The most relevant cognitive structure is the metacognitive component, concerned with monitoring the thinking process and evaluating the solution [Sternberg 1985]. The important cognitive outcome is a self-critical attitude [Gagne 1985], which demonstrates the ability to critically assess one's own thought processes as well as ones' own intellectual creations.

Delivery

We identify three activities for this stage: *documentation*, *presentation*, and *dissemination* of the different solution parts in an organized and comprehensible form. From a process viewpoint, the major cognitive activity at this stage is synthesis which requires the ability to produce a well-organized whole [Bloom 1956]. The most relevant cognitive structure is the performance component directing task organization [Sternberg 1985]. The important cognitive outcome of this stage is verbal information as exhibited by the ability to formulate and organize a complete and coherent report [Gagne 1985].

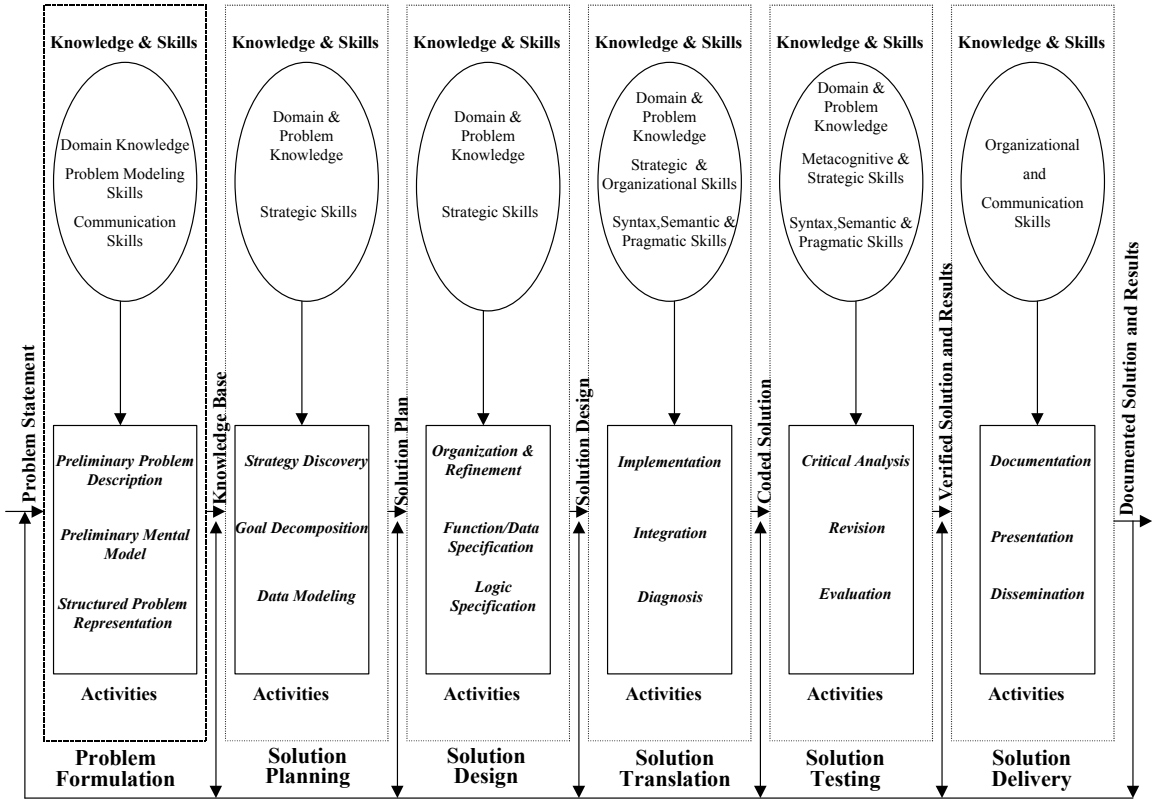

Figure 1. The cognitive activities, knowledge, and skills in the dual common model

CONCLUSION

Research into teaching and learning programming, as well as the development of computing systems which assist novice programmers, underscores the need for a comprehensive framework for programming which includes not only programming language skills but also problem solving, software engineering concepts, and cognitive skills. The tasks of problem solving and program development form an interdependent process, each stage requiring specific knowledge and cognitive skills. Problem solving skills are essential to understanding the fundamentals of computing, and should be learned while studying programming. Problem formulation, planning and design are essential prerequisite tasks to coding and testing because any difficulties or errors at these earlier stages lead to errors in the final stages. Despite this, research and development on the teaching and

learning of programming have devoted disproportionate attention to language-related activities, with less attention given to the earlier tasks of problem definition, requirement, and specification.

REFERENCES

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

Deek, F.P., An Integrated Environment for Problem Solving and Program Development, Unpublished Ph.D. Dissertation, New Jersey Institute of Technology, 1997.

Dewey, J., How We Think, Boston, Massachusetts: Heath, 1910.

Ebrahimi, A., "Novice programmer error: language constructs and plan composition" International Journal of Human-Computer Studies, 41, pp. 457-480, 1994.

Ennis, D., "Combining problem solving and programming instruction to increase the problem solving abilities in high school students", Journal of Research on Computing in Education, 26 (4), pp. 488-496, 1994.

Etter, D.M., "Engineering Problem Solving with ANSI C: Fundamental Concepts", Englewood Cliffs, New Jersey: Prentice Hall, 1995.

Gagne, R.M., The Conditions of Learning, Fourth edition, New York: Holt, Rinehart and Winston, 1985.

Hartman, H., Intelligent Tutoring, preliminary edition, Clearwater, Florida: H&H Publishing Company, 1996.

Johnson, W.L., "Understanding and debugging novice programs", Artificial Intelligence, 42, pp. 51-97, 1990.

Johnson, D.M., The Psychology of Thought and Judgment, New York, New York: Harper, 1955.

Kingsley, H.L., and R. Garry, The Nature and Conditions of Learning, Englewood Cliffs, New Jersey: Prentice Hall, 1957.

Lee, A.Y. and N. Pennington, "Learning Computer Programming: A Route to General Reasoning Skills?" in C.R. Cook, J.C. Sholtz and J.C. Spohrer (Eds.) Empirical Studies of Programmers: Fifth Workshop, pp. 113-136, Norwood, New Jersey, Ablex, 1993.

Mayer, R.E, "The psychology of how novices learn computer programming", ACM Computing Surveys, 3 (1), pp. 121-141, March 1981.

Meier, S.L., R.L. Hovde, and R.L. Meier, "Problem solving: Teachers' perception, content area models, and interdisciplinary connections", Journal of School Science and Mathematics, 96 (5), pp. 230-237, 1996.

Osborn, A., Applied Imagination, New York: Scribner's Sons, 1953.

Parnes, S.J., Creative Behavior Guidebook, New York: Scribner's Sons, 1967.

Perkins, D.N. and F. Martin, "Fragile Knowledge and Neglected Strategies in Novice Programmers", in E.Soloway and S. Iyengar (Eds.) Empirical Studies of Programmers: First Workshop, pp. 213-229, Norwood, New Jersey, Ablex, 1986.

Perkins, D.N., C. Hancock, R. Hobbs, F. Martin, and R. Simmons, "Conditions of learning in novice programmers", Journal of Educational Computing Research, 2 (1), pp. 37-56, 1986.

Polya, G., How to Solve It, Princeton, New Jersey: Princeton University Press, 1945.

Polya, G., Mathematical Discovery: On Understanding, Learning and Teaching problem Solving, New York: Wiley, 1962.

Rist, R.S., "Plans in Programming: Definition, Demonstration and Development", in E.Soloway and S. Iyengar (Eds.) Empirical Studies of Programmers: First Workshop, pp. 28-47, Norwood, New Jersey, Ablex, 1986.

Rogalski, J., and R. Samurcay, "Acquisition of programming knowledge and skills", Psychology of Programming, in J.-M. Hoc, T.R.G. Green, R. Samurcay, D. Gilmore (Eds.), pp. 157-174, London: Academic Press, 1990.

Rogalski, J., and R. Samurcay, "Task analysis and cognitive model as a framework to analyze environments for learning programming", in E. Lemut, B. du Boulay, G. Dettori (Eds.), Cognitive Models and Intelligent environments for Learning programming. pp. 6-19, Berlin: Springer-Verlag, 1993.

Rubinstein, M., Patterns of Problem Solving, Englewood Cliffs: New Jersey, Prentice Hall, 1975.
Shneiderman, B., Software Psychology: Human Factors in Computer and Information Systems, Boston, Massachusetts: Little, Brown and Company, 1980.

Simon, H.A., The New Science of Management, New York, New York: Harper and Row, 1960.
Spohrer, J., and E. Soloway, "Novice mistakes: are the folk wisdom correct?", Communications of the ACM, 29 (7), pp. 624-632, 1986.

Sternberg, R.J., Beyond IQ: A Triarchic Theory of Human Intelligence, Cambridge, Massachusetts: Cambridge University Press, 1985.

Wallas, G., The Art of Thought, New York: Harcourt Brace Jovanovich, 1926.

Weidenbeck, S., V. Fix, and J. Scholtz, "Characteristics of the mental representations of novice and expert programmers: an empirical study", International Journal of Man-Machine Studies, 39, pp. 793-812, 1993.

Wirth, N., "Program development by stepwise refinement, Communications of the ACM, 14 (4), pp. 221-227, 1971.

Fadi P. Deek, James A. McHugh
College of Computing Sciences
New Jersey Institute of Technology
Newark, NJ 07102
973.596.2997 (O)
973.596.5777 (FAX)
Email: fadi.deek@njit.edu