Toward a Framework of Programming Pedagogy

Wilfred W. F. Lau
The University of Hong Kong, Hong Kong

Allan H. K. Yuen
The University of Hong Kong, Hong Kong

INTRODUCTION

As a major topic in information technology education, computer programming has been taught to both major and non-major students in universities. While there has been ongoing debate on whether students should be taught programming (Soloway, 1993), the literature shows that learning to program poses a lot of difficulties to novices (Bonar & Soloway, 1989). Dijkstra (1989) describes programming as "radical novelty" in which our usual strategy of metaphors and analogies simply does not apply. Pea (1986) identifies three types of conceptual bugs which are rooted in a superbug where "there is a hidden mind somewhere in the programming language that has intelligent interpretive powers".

Why is learning to program so difficult? One difficulty is that learning to program needs the acquisition of a multitude of inter-related skills. Jenkins (2002) argues that programming is a complicated task, which requires the mastery of a number of skills such as problem solving, abstraction, mathematical logic and testing, debugging and so forth. A novice programmer simply lacks these skills. More importantly, success in learning to program demands knowledge of computer itself. Ben-Ari (1998) points out that students lack a viable mental model to learn programming. On the other hand, undue emphasis is placed on the learning of programming syntax (Deek, 1999). In this article, we will focus on approaches of teaching computer programing. Winslow (1996) introduced the term "programming pedagogy" in his paper. Although programming pedagogy is not explicitly defined in the paper, the term here refers to any instructional methods and strategies which are used to teach students introductory programming. Due to these reasons, programming pedagogy calls for special attention.

BACKGROUND

Over the years, pedagogical innovations have been proposed to cope with these difficulties. These include a variety of programming tools (Smith & Webb, 2000). These tools help novice programmers to develop programs through program visualisation and algorithm animation. Deek and McHugh (1998) evaluate programming tools used to teach programming. One common problem among these tools is that they fail to integrate into the curriculum. This suggests that there is a need to investigate what programming pedagogy should be adopted together with tools to bring about innovations in programming instruction. This article intends to review on programming pedagogy reported in the literature. A theoretical framework on programming pedagogy grounded on literature review is proposed which attempts to conceptualise pedagogy in terms of the cognitive and technological dimensions. For researchers, this review not only provides a summary of pedagogy adopted to date, but also theoretical underpinning for future research on programming pedagogy. For practitioners, the proposed framework can help them to evaluate and reflect on their own pedagogy with an aim to improve the quality of teaching and learning computer programming.

APPROACHES OF TEACHING PROGRAMMING

We identified seven pedagogical approaches of teaching computer programming arising from the review of appropriate literature. The following sections provide a brief description of each of these approaches.

Structured Programming Approach

In the 1970s, one catchword in programming is structured programming. It is an approach, which intends to "support the production of correct, understandable programs which are easy to modify and maintain" (Freiburghouse & Liskov, 1973). The approach allows control structures of sequence, selection, and repetition only. The GOTO statement is considered detrimental to structured programming (Dijkstra, 1968). To facilitate the development of a structured program, a top-down design, which decomposes a large program into a manageable smaller program, is used. Programs are improved successively through stepwise refinement. In this

manner, it is hoped that quality program can be produced. Although it was advocated in the 1970s, it is still one of the prevalent programming approaches today.

Problem Solving Approach

Barnes, Fincher, and Thompson (1997) describe a programming methodology consisting of four steps, namely, Understanding, Designing, Writing and Reviewing. Following the same line of thinking, Thompson (1997) proposes a problem solving approach in teaching functional programming. He claims that using the approach, "a novice can make substantial progress in completing a programming task before beginning to write any program code." Gries (1974) emphasizes the importance of problem solving in programming. He argues that usual assumption that students should have learned programming after giving tools and examples is not pedagogically sound. To address this problem, he suggests the four-phase process of problem solving by Polya (1957).

Software Development Approach

It is equally important that students should know how to translate algorithms into syntactically and semantically correct solution of the problem that form the program. In this regard, Deek (1999) develops a methodology which incorporates both the problem solving skills and the programming skills into a single process that provides a framework for beginning students. As noted by Deek (1999), there are three kinds of difficulties faced by students when learning to program: (1) deficiencies in problem solving strategies and tactical knowledge; (2) ineffective pedagogy of programming instruction; and (3) misconceptions about syntax, semantics, and pragmatics. The new approach, which incorporates both the problem solving skills and the programming skills, can help address all the three kinds of difficulties.

Small Programming Approach

If programming creates a large cognitive load on novices, it is reasonable to reduce such load by programming in a "small" scale. In this sense, we distinguish between the terms Programming-in-the-small and Programming Language-in-the-small.

Glaser, Hartel, and Garratt (2000) introduce the idea Programming by Number in teaching ML and Java. Programming by Number is intended to get students started in writing program by providing a step-by-step guidance to students while allowing flexibility in the design of the solution to a problem. When writing functions, they suggest the following steps: (1) name the function; (2) write down its type; (3) enumerate all cases; (4) deal with any simple case(s); (5) list the ingredients in preparation for the complex case(s); (6) deal with the complex case(s), where some inspiration is required; and (7) think about the result.

Brusilovsky, Kouchnirenko, Miller, and Tomek (1994) review on three approaches of teaching introductory programming, namely, the incremental approach, the mini-language approach, and the sub-language approach. In the incremental approach, new language subsets, which introduce new programming language constructs while retaining all the constructs of preceding subsets, are introduced successively to novices. In the mini-language approach, a small and simple language is used to support the first steps in learning to program. In most cases, a student learns how to program by controlling an actor, which can be a turtle, a robot, or any other active entity, in a microworld. The sub-language approach uses a special starting subset of the full language which contains easily visualizable operations to introduce programming to novices. In short, these three approaches provide a simple and small language subsets and a visually appealing metaphor embedded in a context-rich environment to help novices start programming.

Language Teaching Approach

Robertson and Lee (1995) give a research manifesto for the application of a second natural language acquisition pedagogy to the teaching of programming languages. They argue that programming has traditionally taught with little reference to natural language pedagogy. To conclude, they provide some areas for further research such as the value of reading programs before writing, the use of authentic programs, the study of the cultural milieu of programs, and so forth. Baldwin and Macredie (1999) argue that research in the learner strategies in second language pedagogy may provide insight into programming pegadogy. Based on the call for a more learner-centred environment, they believe that learner strategies are one of the issues in teaching programming that can help address difficulties in learning to program. Deek and Friedman (2001) describe their ideas of how programming and writing are learned in parallel. They argue that the common element that exists in both domains, problem solving and program development, provides "new ways for students to transfer skills between domains".

Learning Theory Approach

Lister and Leaney (2003) argue that traditional norm-referencing approach to grading tends to target at average students. As a result, weaker students cannot program well and stronger students are not challenged. They suggest a criterion-referencing approach to grading so that explicit and clear criteria are set for each grade. In deciding the criteria, reference is made to the Bloom's Taxonomy of Educational Objectives (Bloom, 1956). Macfarlane and Mynatt (1988) examine the effectiveness of advance organizer in teaching