4

Automating the Planning and Construction of Programming Assignments for Teaching Introductory Computer Programming

Jeroen J. G. van Merriënboer, Hein P. M. Krammer, and Rudolf M. Maaswinkel

Department of Instructional Technology, University of Twente,
P.O. Box 217, 7500 AE Enschede, The Netherlands

Abstract: This chapter describes CASCO, an automated system for the planning and construction of programming tasks for introductory computer programming. The generated tasks have the form of completion assignments, which consist of an incomplete example program with (a) instructions to complete, extend or change the program so that it meets certain specifications, (b) explanations on new features that are illustrated by parts of the incomplete program, and (c) questions on the working and the structure of the program. The planning and construction of the completion assignments is based on a model in terms of programming plans, student profile, and problem database.

Keywords: intelligent task generation, computer programming, instructional strategies, intelligent tutoring systems (ITS), student modelling

4.1 Introduction

The goal of this chapter is to describe a system for intelligent *task generation* in the domain of teaching computer programming to novice programmers. Task generation refers to the dynamic construction of student exercises, cases, problems or assignments in such a way that they are tailored to the particular needs of individual students. Research on intelligent task generation in

R. D. Tennyson (ed.), *Automating Instructional Design, Development, and Delivery*

computer-based instructional systems began as early as the end of the 60's (e.g., [9]) and has, amongst others, been conducted in the fields of elementary arithmetic (IDEBUGGY, [3]; LMS, [8]), teaching economics (SMITHTOWN, [7]), and trouble shooting of electronic circuits (MHO, [6]). With regard to the field of introductory computer programming, the authors are not familiar with other research pertaining to the dynamic construction of programming assignments. However, there has been done important work on the selection of programming assignments from an existing database of problems, such as in the Stanford BIP project (Basic Instructional Program, [2, 13]).

The tasks that will be constructed by our system are "completion assignments." These kind of assignments serve a central role in the *Completion Strategy*: An instructional strategy for teaching introductory computer programming which focuses on the completion of increasingly larger parts of well-designed, well-readable but incomplete computer programs [12]. In several experiments [10, 11], the Completion Strategy yielded higher learning outcomes than more traditional strategies that were focusing on the students' unconstrained generation of new computer programs. In this strategy, students are offered a series of programming assignments which may consist of the following five elements:

1. A programming problem as described in a problem text - this is the only element that is required for *all* completion assignments.
2. An example program - usually the student is offered an incomplete example program that yields a partial solution to the posed programming problem; however, the student may also be offered a complete example program or no example program at all.
3. Task instructions - either to solve the programming problem (if no example program is provided), to complete the incomplete example program, or to extend or change a complete example program so that it meets a new problem specification.
4. Explanations on new features of the programming language that are illustrated by--parts of--the example program.
5. Questions on the working and the structure of the example program.