SOFTWARE PATTERN CONSTRUCTS

Ross A. Gagliano and G. Scott Owen
Department of Mathematics and Computer Science
Georgia State University
Atlanta, Georgia 30303-3083

Abstract: A software pattern construct is a syntactical structure in a higher order language which provides examples of correct format for common programming expressions. Typical applications include: input and output phrasing and the use of prompts; control structures; conditional and logical constructs; and secondary data structures as arrays, records, and files. This paper attempts to describe software patterns, and to provide motivation and examples for their use with several languages in an evolving on-line reference system.

INTRODUCTION

Novice programmers, occasional programmers, and programmers who use a variety of languages frequently have difficulty in remembering the precise syntax of a particular software construct. An example of this, which is shown later, is the multiway branch statement; e.g., CASE in Pascal and Ada, and SWITCH in C. These constructs are all quite similar, but do have slight variations among them.

Another example might be the use of string versus character delimiters. Ada and C use " for strings and ' for characters while Pascal uses ' for both strings and characters. A simple example of the particular software construct is usually sufficient, and it has occurred to us that the computer system itself could provide such assistance for the more common expressions.

The notion of a software pattern construct is modeled after the so-called Cloze Procedure (Cook, Bregar, and Foote, 1982; Entin, 1986). This procedure utilizes a skeleton of a particular fragment of code to test the skill of the programmer. This use of the Cloze Procedure with programming languages was derived from its original application, that of measuring the comprehension of a group of readers by requiring them to correctly insert words that were deleted from a passage of prose.

Another approach to the problem was to provide what is called a "Conversational Programming System" (Sebesta, 1985). These systems consist of: syntax directed editors, graphics oriented editors, incremental compilers, and other software tools. Several variations of this approach have been attempted, and work is continuing in this area (Meyer, 1986; Harrison,
Rosenfeld, Wang, and Westin, 1986). However, these systems are difficult to implement, and, also, cumbersome to modify for new languages.

A much simpler approach, it seems to us, would be to have a set of readily available sample software pattern constructs. This would be both more easily implementable and directly modifiable to incorporate new languages. Thus, our work on the development of a set of software pattern constructs has begun, and a viable means to implement their use has been explored in an on-line system at the Georgia State University.

ORLS SYSTEM

A major goal of this research is to establish an Online Reference for Language Syntax (ORLS) System. The ORLS system would be menu driven, and would include several programming languages. The objective of the ORLS system is to provide examples of many of the necessary Software Pattern Constructs in order that programmers could quickly resolve their own questions regarding language syntax. This paper presents some preliminaries of the design and implementation of the ORLS system. Even though this system is primarily aimed at undergraduate and graduate students, its principles are generally applicable to other programming environments.

Before discussing the design of the ORLS system let us define the constraints and provide a description of our local programming environment. At Georgia State University (GSU), much of the Computer Science (CSc) program is based on IBM PC compatible microcomputers. The students are exposed to a variety of programming languages, all of which run on these machines. These languages include Pascal, Ada, C, FORTRAN, and PROLOG.

The students use Pascal (and some FORTRAN) for their first two courses, and then Ada is used for the third course (Data Structures). They are introduced to the languages C and PROLOG in a course on Programming Languages. In many courses, e.g., Computer Graphics and Numerical Analysis, they can use any language of their choice. More languages will be made available in the near future.

Until recently, the PC's available to the students were stand alone floppy disk based systems. These systems are all hardwired to the GSU mainframe system, thus used primarily as terminals by both the students and many others outside of the CSc program. With this configuration, any kind of general on-line help facility was not feasible. However, the GSU Computer Center now has begun to network all of the PC's together with several hard disk based file servers. With these commonly accessible disks, it is now feasible to implement the ORLS system.

The ORLS system will actually be a part of an overall help system which the Computer Center is in the process of installing. The primary focus of the overall system will not be beginning programmers, but, users who are mainly running canned programs on the mainframe.

This major system is designed in two parts. A small memory resident program will always be available via a set of keystrokes. Having this program resident in memory allows the user to invoke it in the middle of editing text or other mainframe sessions, without exiting. When invoked, this program will display a menu of possible choices. After the user has made a menu selection, the program will then download from the hard disk system the appropriate submenus or help screens.

The ORLS system will be made a part of the main menu. When this choice is made, a second menu will be displayed. This menu will list the available