Characteristics for the Basic Syntax of an Object-Oriented Design Language (OODL)

Nancy K. Gautier
Southeastern Louisiana University

D.L. Carver
Louisiana State University

Abstract

Formal specification languages were originally developed to satisfy a need to complement natural language descriptions of requirements specifications. However, formal specification languages have permeated all stages of system development. The common basis for formal specification languages is a firm mathematical foundation for a formal method. It is difficult to accommodate the diverse technology and complex systems of today with one formal method. As a result, various areas have been addressed by formal methods and Wing[1] has developed a taxonomy of formal methods.

This work focuses on the design phase of software development. Object-Oriented techniques are touted as a more natural way of developing today's data-oriented, interactive, on-line systems. Top-down design methods are not conducive to tapping the potential of Object-Oriented Programming languages of today, namely inheritance, which promotes reusability of code.[2] To meet this demand, numerous Object-Oriented Design methodologies have been proposed. Various design notations have been developed as well and take many different forms. One such design notation is a Program Design Language (PDL). A PDL may be either an extension of an existing language, adding new constructs to it, or it may be developed specifically for use as a design representation. Besides combining several qualities of a good design notation, a PDL can be used to provide documentation for ease of ongoing maintenance of software. Depending on the final choice of the implementation language, there is a possibility for automatic code generation.

Most existing PDL's have been developed for procedural language design methods, namely top-down design. A set of characteristics for a basic PDL syntax has been provided by Pressman[3]. These characteristics were derived from procedural languages. Modern systems can be developed more rapidly by adapting existing software to meet the needs of the current application domain. In order for an Object-Oriented Design Language to accommodate this goal, one must determine how an existing library of classes can be best utilized. This utilization is dependent upon the type of inheritance that an implementation language has to offer, whether it be single or multiple inheritance.

Certain domain applications lend themselves more readily to a design that is best implemented using single inheritance, and others would benefit most from a design which has incorporated multiple inheritance. As a result of examining the different Object-Oriented Design methodologies that have been developed in recent years, we describe a basic set of characteristics for an Object-Oriented Design Language. These characteristics provide a foundation for the development of an Object-Oriented Design Language.

Keywords: Formal Specification Language, Object-Oriented Design, Program Design Language, Inheritance

BIBLIOGRAPHY

[1] Wing, Jeannette M., "A specifier's Introduction to Formal Methods", COMPUTER, September 1990, pg 8-24.

[2] Meyer, Bertrand, "Reusability: The Case for Object-Oriented Design", IEEE Software, March, 1987.

[3] Pressman, Roger S., Software Engineering: A Practitioner's Approach, 2nd ed., McGraw-Hill, 1987.