Object Oriented Development of Engineering Software Using CLIPS

C. John Yoon
Assistant Professor
Department of Civil Engineering
Polytechnic University
Brooklyn, NY 11201

Object oriented approaches have emerged as one of the most promising paradigms for developing softwares that are reliable, flexible and maintainable[1]. During the early 1980's, there was much debate on what is object oriented programming; that debate is now more or less settled. Currently, active research centers around effective software design and development methodology using an object oriented programming paradigm [2].

The origins of CLIPS, the C Language Integrated Production System, date back to 1984 at NASA's Johnson Space Center and as the acronym indicates, it started as an expert system shell [3]. CLIPS is now used by over 3000 users throughout the public and private community including all NASA sites, branches of the military, numerous federal bureaus, government contractors, universities and many companies. CLIPS has grown to accommodate many other functions, such as utilities for embedded applications. The most recent release of CLIPS, version 5.0, introduces object oriented programming with COOL (CLIPS Object Oriented Language).

The work during the tenure involved (1) critical evaluation of COOL as an object oriented language, (2) investigation of object oriented software design and development methodolgy for engineering software and (3) implementation of a method for optimal sensor placement proposed by Kammer[4] with COOL.

As an object oriented language, COOL is more powerful than other widely used object oriented languages such as Smalltalk and C++. In Smalltalk, literally everything in it is an object and in that sense, Smalltalk is a pure object oriented language. Because a construct such as a class in COOL is not an object, COOL is not a pure object oriented language. However, COOL supports inheritance, encapsulation, abstraction, polymorphism, and dynamic binding - the five primary characteristics generally agreed that an object oriented language must support. COOL provides a rich set of utilities and tools, such as browsers, that are essential in developing an object oriented program. Lack of these tools

has been a criticism for many existing object oriented languages such as C++.

Engineering applications involve numeric complexity and manipulations of a large amount of data. Traditionally, numeric computation has been the concern in developing an engineering software. As engineering application software became larger and more complex, management of resources such as data, rather than the numeric complexity, has become the major software design problem. Object oriented design and implementation methodologies can improve the reliability, flexibility and maintainability of the resulting software; some tasks, however, are better solved with the traditional procedural paradigm. CLIPS, with deffunction and defgeneric constructs, supports procedural paradigm. The natural blending of object oriented and procedural paradigms has been cited as the reason for popularity of the C++ language. COOL's object oriented features are more versatile than C++'s. A software design methodology based on object oriented and procedural approaches appropriate for engineering software, and to be implemented in CLIPS, has been outlined. A method for sensor placement for Space Station Freedom that was proposed by Kammer is being implemented in COOL as a sample problem.

References

1. Cox, B.J., Object Oriented Programming, Addison-Wesley, Reading, MA, 1986.

2. Yoon, C.J., "An Object Oriented Design and Implementation of a Finite Element and Graphics Data Translation Facility", accepted for publication, special issue of the ASCE Journal of Computing in Civil Engineering devoted to object oriented concepts and applications.

3. Giarratano, J.C., CLIPS User's Guide, Version 5.0, Vol. 2, NASA Johnson Space Center, May 1991.

4. Kammer, D.C., "Sensor Placement for On-Orbit Modal Identification and Correlation of Large Space Structures", Journal of Guidance, Control, and Dynamics, Vol. 14, No. 2, Mar-Apr 1991.