Informality in program specifications

by ROBERT BALZER, NEIL GOLDMAN and DAVID WILE

Information Sciences Institute
Marina del Rey, California

A critical step in the development of a software system occurs when its goal-oriented requirements specification is transformed into a process-oriented form that specifies how the requirements are to be achieved. Only after this transformation has occurred can the feasibility of the system be analyzed and the consistency of the process specification with the requirements be verified. The key to this transformation is expressing the process-oriented specification abstractly so that its functionality is completely determined while the class of possible implementations remains broad.

We believe that such abstract process-oriented specifications are the key to rationalizing the software development process. Such specifications are, in reality, programs written in a very high level abstract programming language. As such, they could provide an effective interface between the two major software concerns: functionality and efficiency. These concerns should be decoupled so that the functionality of a system can be addressed before its efficiency has been considered. Once functionality has been accepted, it can be preserved while the system is optimized. Thus, since the abstract process-oriented specification is a program, its consistency with the requirements could be formally verified, informally argued, or tested by actually executing the specification. Furthermore, the end user could be given hands-on experience exercising the specification to see if it behaved as intended. Deviations and/or inconsistencies could be corrected in the specification before any implementation occurred.

Once the system's functionality has been accepted by the user, the efficiency of the system in meeting its performance requirements remains an issue. Such efficiency must be gained without altering the system's accepted functionality. We have argued elsewhere that a computer-based tool can be built which guarantees maintenance of functionality while a program is optimized without sacrificing the programmer's ingenuity or initiative in determining how best to achieve efficiency.

In this work we are concerned primarily with the procedure by which such process-oriented specifications are obtained and with computer-based tools for their construction. We will begin by determining some attributes of a suitable process-oriented specification language, then examine why specifications would still be difficult to write in such a language. We will argue that the key to overcoming these difficulties is the careful introduction of informality based on partial, rather than complete, descriptions and the use of a computer-based tool which utilizes context extensively to complete these descriptions during the process of constructing a well formed specification. We will then present some results obtained by a prototype of such a computer-based tool on a few informal example specifications. Finally, we will discuss some of the techniques used by this prototype system.

REFERENCES
