POPEK77
Popek, G.J., J.J. Horning, B.W. Lampson, J.G. Mitchell and R.L. London; Notes on the Design of Euclid; Proceedings of an ACM Conference on Language Design for Reliable Software, (edited by D.B. Wortman). Combined issues SIGPLAN Notices 12, 3 (March 1977), Operating Systems Review 11, 2 (April 1977), Software Engineering Notes 2, 2 (March 1977) pp. 11-18.

The programming language Euclid, developed from Pascal, was intended to be suitable for program verification and for system programming. This paper, written by the designers of the language, describes the design goals for Euclid, lists some of the principle differences between Euclid and Pascal, discusses the reasons behind, and consequences of, several of these differences, and summarizes the perceived success of the design effort.

One intent of the language was to improve the reliability of the programming process by increasing the class of compiler-detectable errors, and by requiring that more of the information needed for understanding and maintenance be included in the text of the program. Since it was intended that all Euclid programs be verified before use (either manually, or by machine), it was felt that this, too, would contribute to program reliability. While these three goals (reliability, understanding and verifiability) are closely linked, verifiability is the only one which can actually be measured, so the paper tends to concentrate on verification-related design decisions.

To some extent, another major goal of Euclid (the construction of "acceptably efficient system programs") is at odds with the preceding three. While the paper does not discuss these issues in much detail, mention is made of features which do contribute to reliability (such as having explicit machine-dependent modules which contain, or limit, the effect of machine-dependent instructions).

The paper illustrates how some of the goals and assumptions directed the actual design. For example, the assumption that all programs would be verified, specifically by the axiomatic approach of Hoare and Wirth, led to the inclusion in the language of a syntax for expressing specifications and intermediate assertions. By having assertions as part of the language, they can be included in the compilation, and if any assertion is evaluated to False during execution, the program terminates. The same assumption also led to the exclusion of exception handlers (since verified programs should not have run-time software errors), to an "unusual" approach to uninitialized variables and dangling pointers, and necessitated that pains be taken to ensure that there were no discrepancies between the definition of the language and the enforcement by the implementation (such as is the case with Pascal).

The authors concede that Euclid was not a dramatic advance in the state of the art, but it did demonstrate at least three important points with respect to reliability:

it is possible to design a useful language with all features verifiable in principle (except, perhaps, machine-dependent ones)

it is possible to eliminate aliasing in a practical programming language

variant records can be made type-safe

Two of their other design goals (to make "minimal changes and extensions to Pascal", and to keep the effort "quite limited: only a month or two in duration") were not met, but the authors feel that their final design only met with success because of the original inclusion of those goals. Specifically, had their original aim been any more ambitious, they might never have completed the design.