GRESCHKE77

Greschke,C., Morris,J., and Satterthwaite,E.;
Early Experience with Mesa;
CACM, 20,8 (August 1977)

In this paper, the authors' first hand experiences in using Mesa is discussed. Some of the design decision made and lesson learnt are also touched on. Mesa is a modular, strongly typed language which is strongly influenced by Pascal or Algol 68 (for its local structure) and Simula (for its global structure).

First, the authors present their believe that module is essential in a system programming language; to provide a certain degree of protection and abstraction capability. Modules in Mesa are of two kinds: definitions and programs. With definitions module defines the interface to an abstraction, and program modules being the implementers. The mapping between definitions modfule and programs module is not one-to-one, there can be multiple i instances of program module which claim to implemente and abstraction, a definition module.

Then the authors' mixed feeling of strict vs nonstrict type checking is discussed with Mesa type system as an example. Mesa supports a large set of primitive types and provides a set of type operators to construct new type. Primarily, the authors prefer strict type checking in program languages as in Mesa since this is less error prone and produce more reliable code. But they also realize that flexibility and performance issues in a system language, they leave a loophole in Mesa to allow one to violating the type system.

Finally, the authors extend the above discussion to the problem of mutable variant record. They argue that it is too expensive to patch the loophole caused by this problem and it is more powerful if used with discipline.

As a concluding comment, this paper presents a history of a programming languages - Mesa; from the design stage through its development and observations made from the resulting product. Although Mesa is not reimplemented or redesigned, the authors clearly point out the important of feedback in programming language design.