In Memory of Maurice H. Halstead*

I shall be telling this with a sigh
Somewhere ages and ages hence:
Two roads diverged in a wood, and I—
I took the one less traveled by,
And that has made all the difference.

Robert Frost

MAURY Halstead had a dream! By treating computer programs as neither art forms nor as examples of mathematical logic, but instead as basic material which can be investigated with the classical methods of experimental and theoretical natural science, Maury had dreamed of and worked hard toward a unified and coherent new field he called Software Science, in which attributes of a computer program, such as implementation efforts, clarity, structure, error rates, language levels, etc., can be derived from its basic metrics through quantitative hypotheses.

This special collection of papers on Software Science not only contains some of Maury's final contributions to the field he started, but its diversity and sophistication is an assurance that Maury's dream will be carried on.

RAYMOND T. YEH
Editor

*Professor Maurice H. Halstead died from acute myocardial infarction at 8:52 a.m., January 8, 1979.

Guest Editorial on Software Science

Software science is a term used to describe an experimental and theoretical discipline concerned with measurable properties of written material as expressed either in computer programs or in prose. It originated in an unexpected way some seven years ago, when it was realized that a computer program need not be considered solely as an art form or even as an example of mathematical logic, but that instead it could be treated as a structure subject to investigation via the classical methods of natural science.

This required that its properties be measured, not in terms of its execution on any machine, but with metrics directly observable from the program itself. By recalling that early machine language computer programs consisted of nothing but a series of one-word instructions, each of which contained an operator code and an operand address, it followed that it consisted of operators and operands, and of nothing else. Consequently, the easiest measures should be simple counts of the operators and operands in a program. But each class could be counted in two ways, either as the number of different elements in the class, or their total occurrences.

Applying this measurement technique to five versions of the same small routine expressed in different programming languages yielded four values for each version, the vocabulary components \( \eta_1 \) and \( \eta_2 \), and the length components \( N_1 \) and \( N_2 \). At this point a pattern was discernable, in that the total length was clearly a function of the two vocabulary components. At the time, this vocabulary-length relation was considered counter-intuitive, but sufficiently intriguing to warrant similar measurements on other programs. When the same relationship was found to apply to other programs readily available, it was realized that the vocabulary-length hypothesis should be tested against an objectively selected set of published programs.