The state of software engineering is diverse, disorderly, and expanding. Coined less than seven years ago, the term is used today to describe such disparate activities as programming methodology, software development, and programming tools and standards. But for all the confusion surrounding it, this new discipline does seem to promise solutions to some of the most vexing problems of modern software development efforts: reliability and complexity. Clearly, the subject deserves to be explored.

To explain what software engineering is about, we need to examine its goals, as well as the fundamental issues and principles derivable from those goals. Then we should illustrate these principles with specific examples. This is indeed what this special issue of Computer attempts to do.

The first paper provides a useful framework for understanding the basic ideas of software engineering and the roles these ideas play. It attempts to expound what the basic principles are which, if followed, will ensure “well-engineered” software development, and how these principles can be applied.

The second paper describes the actual practice of software engineering at Systems Development Corporation. It describes a “software factory” which consists of an integrated and extensible facility of software development tools. The factory supports a methodology which emphasizes discipline and repeatability but with sufficient generality and flexibility. The “software factory” described here is not a factory in the usual sense that it contains an automated production line which fabricates a finished product (software system) by processing raw materials (system specification). Rather, it is one that supports a group of craftsmen (system designers and programmers) with standardized tools which guide and support them in building a system structure that fits together, increases their productivity, and enhances management control of the process. Many of the ideas and principles outlined in this paper can be extracted to fit into the general framework described in the first paper.

The third paper diverges from the general principles discussed in the first two papers and concerns itself with a specific subject area of reliability modeling. The paper discusses the philosophy and general approaches of modeling for reliability. It points out the usefulness as well as dangers of modeling, provides an overall framework for selecting different strategies for improving reliability, and discusses hardware-software tradeoffs in applying redundancy techniques.

Raymond T. Yeh is chairman of the Computer Sciences Department at the University of Texas at Austin. Prior to joining the University of Texas, he was on the computer science faculty at Penn State. His current academic interests include research in program verification techniques and systems, automatic programming, theory of relational data structures, and theory of computation. He is also involved in the development of an interdisciplinary software engineering curriculum.

Dr. Yeh has published one book and over 30 technical papers, and is the author of the IEEE Computer Society’s new quarterly, Transactions on Software Engineering. He is the founder of the Texas Conference on Computing Systems, as well as co-chairman of the Computer Society’s Technical Committee on Software Engineering and the 1975 International Conference on Reliable Software.

He received his BSEE in 1961, his MA in mathematics in 1963, and his PhD in mathematics in 1966—all three from the University of Illinois.