Preface

The theme of effective programming, begun in the previous issue, continues herein with three papers describing code inspections, composite design, and program dynamics.

The technique of design and code inspections, originally developed at the IBM Laboratory in Kingston, New York, can produce substantial improvements in program quality and programmer productivity. As described by Fagan, these improvements are made possible through a systematic and efficient design and code verification process, requiring well-defined roles for inspection participants. The author shows that inspections can result in an initial error reduction followed by declining error rates

Composite design, a program design technology for structuring a program into a hierarchy of highly independent modules, has been used to produce programs of high reliability with reduced costs for development, maintenance, and modification. Glenford Myers, the originator of the term *composite design*, examines the relationships between this technology and six commonly used programming languages. Strengths and weaknesses of these languages are discussed, and language facilities for greater use of the potential of composite design are suggested.

The third paper on effective programming, by Belady and Lehman, provides a macroscopic view of the dynamics involved in modifying and maintaining a very large programming system. Currently, the process of large-scale program development and maintenance leads to uncertainties; costs are high, and the output is not fully satisfactory. The authors argue that a full under-

standing of *what* is happening is a prerequisite to discovering *why* things happen. Using data based on the history of OS/360, Belady and Lehman postulate three laws of program evolution dynamics: the Law of Continuing Change, the Law of Increasing Entropy, and the Law of Statistically Smooth Growth By pursuing the consequences of these laws the authors develop models of the programming process.

In addition to the three papers described above, which were formally refereed, this issue includes sections called **Readings** and **Forum.** The selection for **Readings** is a short paper on data base systems, citing selected references on hierarchical, network, and relational data bases. The **Forum** consists of a letter and a response dealing with the important topic of system integrity as implemented in virtual machine monitors.

For assistance with the programming papers in this and the previous issue, the Editor thanks Geraldine Horne, Joan Rennaker, and Joan Woods.

George McQuilken
Editor