The IBM large-systems software development process: Objectives and direction

by W. S. Humphrey

This paper introduces a special issue of the* IBM Systems Journal *on the IBM large-systems software development process. The issue provides an overview of the subject and a summary of the key principles of the IBM software quality and productivity efforts in large-scale systems programming. The major topics addressed in this issue are the software development process, software development tools and methodologies, quality and productivity measurements, and programmer education.

The enormous growth of data processing in the last thirty years has been made possible by impressive improvements in the function and performance of both hardware and software. Hardware speed has grown three orders of magnitude during those thirty years, a factor of ten each decade. This has resulted in a steady growth in programming workload to meet the escalating needs for system function. Today, the IBM large-systems organizations produce several million lines of new or changed code every year. The annual output now exceeds the total size of the large operating systems of only a few years ago. Even this rate of development, however, does not approach the demand.

There is no foreseeable limit to the amount of programming needed, and there is, as yet, no slackening of demand. The primary limit on the functional capability of programming systems today is the availability of skilled and trained programmers and the environment that supports them. The fundamental issue is an economic imperative: Produce function at a cost, quality, and schedule that meet users' needs. This can best be done in a development environment that is staffed with well-trained people, properly structured, and fully instrumented. This process is the subject of this issue of the *IBM Systems Journal.*

Status and challenge

Programming is a people-intensive discipline, and personal work habits require time to change. Even so, there has been great progress in the last twenty years. Programming has evolved from a highly individualistic profession into an orderly engineering process. Programmers once had trouble producing or meeting responsible plans, and cost estimates were often little more than guesses. Today, however, programming estimates are as accurate as those in engineering, where cost deviations are typically only a few percentage points. Whereas initial program line-of-code estimates are often optimistic by twenty percent or more, manpower estimates are within ten percent, and dollar costs are typically in error by less than five percent over the full development life cycle. This level of predictability is a dramatic improvement over that of the past.

The first step in improving the programming development process was learning how to make and meet schedules and estimates. Having acquired the ability to predict resources and produce consistent results, programming is now in a position to make significant further advances.

With schedules and estimates under better control, programming is entering an age of process management, which is the theme of this introductory paper.

In the last eight years, the overall size of large IBM operating systems has nearly doubled.

While this has been a recent management focus, enormous progress has already been made. For example, in the last eight years, the overall size of large IBM operating systems has nearly doubled. At the same time, the total number of their reported defects has decreased by nearly twenty percent. Thus, on a defects-per-line-of-code basis, quality has improved by a factor of approximately two. In programs now completing development, quality is three to five times better than it was only eight years ago.

Our experience with programmer productivity during the last eight years has also been very positive. During that time, IBM's large-systems programming staff has grown less than ten percent per year. Code shipments, however, have increased at nearly twice the rate of the staff growth. Although productivity improvements in individual programs and even entire laboratories may fluctuate widely about this trend, the average rate of productivity improvement has held for nearly a decade for a population of several thousand programmers. This is a reflection of improved programmer skill levels, improved methods, and better tools. We also believe these improvements will continue for the foreseeable future.

At the same time, the skills of programming managers have continued to improve. Our managers today have generally worked previously as programmers, and they have grown in their appreciation for business and marketing issues. The typical programming manager has a technical education and often has an advanced degree. Managers now have an understanding of programmers and their jobs, have extensive management training, know the products, and have a technical as well as an intuitive sense of the programming process and how to improve it.

This intuition is being further augmented by an increasingly quantitative understanding of the management process. The evolution from a nonquantitative to a quantitative and statistical basis for programming process management is already showing excellent results.

The quality software process

Significant progress toward the goal of a zero-defect development process will require advances in tools, methods, and process. Of these three, process is the key, for the effectiveness of tools and methods can be ensured only through an orderly process.

Much of the work in IBM toward improving programming productivity and quality is based on principles of process management that have been successfully used in several fields. Remarkable advances in semiconductor technology, for example, have been achieved largely through process analysis and improvement. These advances involve process measurements, statistical analysis, and process control via data feedback.

In systems programming, significant quality improvements have been achieved with traditional methods, but this trend has slowed. We are now focusing on the programming process by defining and documenting the steps in the process itself and by educating the people. Key measurements are established, data are gathered and analyzed, and corrective actions are implemented. We use the following set of process management principles:

People management

- Professionals are the key to the programming process, and they must be intimately involved in its improvement.
- Management must focus on defects not as personal issues but as process problems.

Process methodology

- The process is formally defined.
- Goals and measurements are established.
- Statistical data are gathered and analyzed to identify problems and determine causes.

Process control

- Management practices are established to control change.
- Periodic process assessments are planned to monitor effectiveness and identify needed improvements.
- Procedures are established to certify product quality and implement corrective actions.

Process support

- Special process groups are established.
- Needed management and professional education is provided.
- The best tools and methods are obtained and used.

The future

The goal of a zero-defect programming development process is to produce large-scale systems with the assurance that they are defect-free. This quality goal is important for several reasons. Since defects are disruptive and expensive to our customers and to IBM, quality improvements are an economic necessity. Of even greater importance is the improved usability of our programs. Although significant advances in usability have been made, another of our continuing goals is to deliver sophisticated functions to untrained users. These users might benefit from more automatic and error-free operation, but the need for simpler and more automatic systems that provide increasing function involves a basic conflict. Resolution of this conflict will require enormous increases in the amounts of code delivered. To be effective, usability advances must be built on a foundation of quality, and if the code is to be trouble-free, its quality must improve at a far greater rate than it has to date. The scale of large systems has grown by three orders of magnitude in the last thirty years, and this rate of growth is likely to continue or even increase in the future.

An orderly and structured process addresses programming quality by providing an environment that fosters predictable results and eliminates mistakes. To use the analogy of advanced medical research, for example, it is understood that a controlled environment is essential for competent work. A similar environmental discipline is now recognized as an important element in good programming. Without change control, test-case tracking, statistical data bases, and structured requirements, the programmers often repeat past mistakes or re-solve the same problems. Process discipline reduces such waste, permits more orderly learning, and allows the professionals to build on the experiences of others. That discipline also provides the essential foundation for increased mechanization of the programming job itself.

There is no magic route to process discipline. It requires our dedication to continuous growth and improvement. Gains, once made, must be retained, and the ingenuity of our people must not be lost solving problems that have already been solved. Although many technical hurdles lie ahead, our continuing challenge is the fostering of an orderly and reproducible programming process. Upon this foundation continued improvements in programming quality can be built.

Watts S. Humphrey *IBM Information Systems and Technology Group, Financial Plaza, Box 390, Poughkeepsie, New York 12602.* Mr. Humphrey is Director of Programming Quality and Process. He joined IBM in 1959 and began his career with the Advanced Systems Development Division. Among the positions he has held, Mr. Humphrey has been Director of Programming in the Systems Development Division, Director of the Endicott Laboratory, and Director of Policy Development on the IBM Corporate Staff. In November 1982, he was appointed to his present position, reporting to George F. Kennard, Group Director of Quality and Assurance. Mr. Humphrey is the author of the book *Switching Circuits with Computer Applications*, published by the McGraw-Hill Book Co., New York (1958). He has also written a number of technical and management papers, the most recent of which, on the subject of Japanese management, was published in *Manufacturing Engineering* in April 1982. He is currently writing a book on technical management. Mr. Humphrey received a B.S. in physics from the University of Chicago in 1949, an M.S. in physics from the Illinois Institute of Technology in 1950, and an M.B.A. from the University of Chicago in 1951. He holds a number of patents in computer design. Mr. Humphrey is a member of the ACM and is an IEEE Fellow.

Reprint Order No. G321-5239.