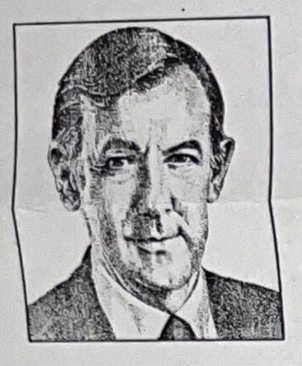
APPLIED INTELLIGENCE

Gain Management Support by Measuring Productivity



JAMES MARTIN

This is the third of a series of articles on rapid application development (RAD), a development methodology designed to be much faster than the traditional method.

Information-systems (IS) organizations strive to develop systems that are faster, higher in quality and lower in cost.

This can be achieved by using new development techniques such as RAD. The RAD life cycle is based on the use of integrated computer-aided software engineering (CASE) tools, small teams of highly trained and motivated personnel, interactive joint application design (JAD) techniques, and management processes focused on breaking down obstacles to high-speed development.

RAD is an effective management technique for application development. It focuses on the management techniques needed to facilitate communication between end users and analysts, motivate the development team, build complete applications within a strictly defined window of time, and ensure that completed applications meet end users' needs.

Approximately 80 percent of RAD's effectiveness is attributable to better management processes; only about 20 percent is due to the use of computer technology.

One of RAD's management techniques is the measurement and reporting of development productivity. Appropriate measurements enable managers to motivate and set goals for developers. They permit the comparison of tools and techniques, and enable managers to detect, examine and, possibly, replicate examples of excellence.

Periodic measurements of productivity enable managers to establish quality standards and reward development teams that achieve high levels of productivity and quality. Measurement techniques also facilitate estimating and planning. They can be used to identify areas in which improvements are needed and to establish an ongoing program for improvement.

All meaningful measurement of development productivity should relate to project complexity. In IS organizations, it is valuable to plot various measurements of productivity against project complexity. But how do we measure complexity?

At the time when all projects were built with third-generation languages such as COBOL, complexity was usually measured by counting the number of lines of code. This metric cannot be used for direct comparison of fourth-generation language (4GL) projects with COBOL projects because a 4GL may require a great deal less code.

Lines of COBOL must be counted with caution. The count should include executable lines of code and the Data Division lines, but should exclude comments.

It's desirable to measure project complexity independent of any particular computer language.

The complexity metric that's most commonly used is the function-point method. This was developed at IBM and improved in GUIDE, the IBM users' organization.

The function-point method lists and counts the elements of a system, including:

- inputs (screens, messages and batch transactions);
- outputs (screens, reports, messages and batch transactions);
- system data stores (logical files);
- data stores shared with other systems (external interface files); and
- inquiries.

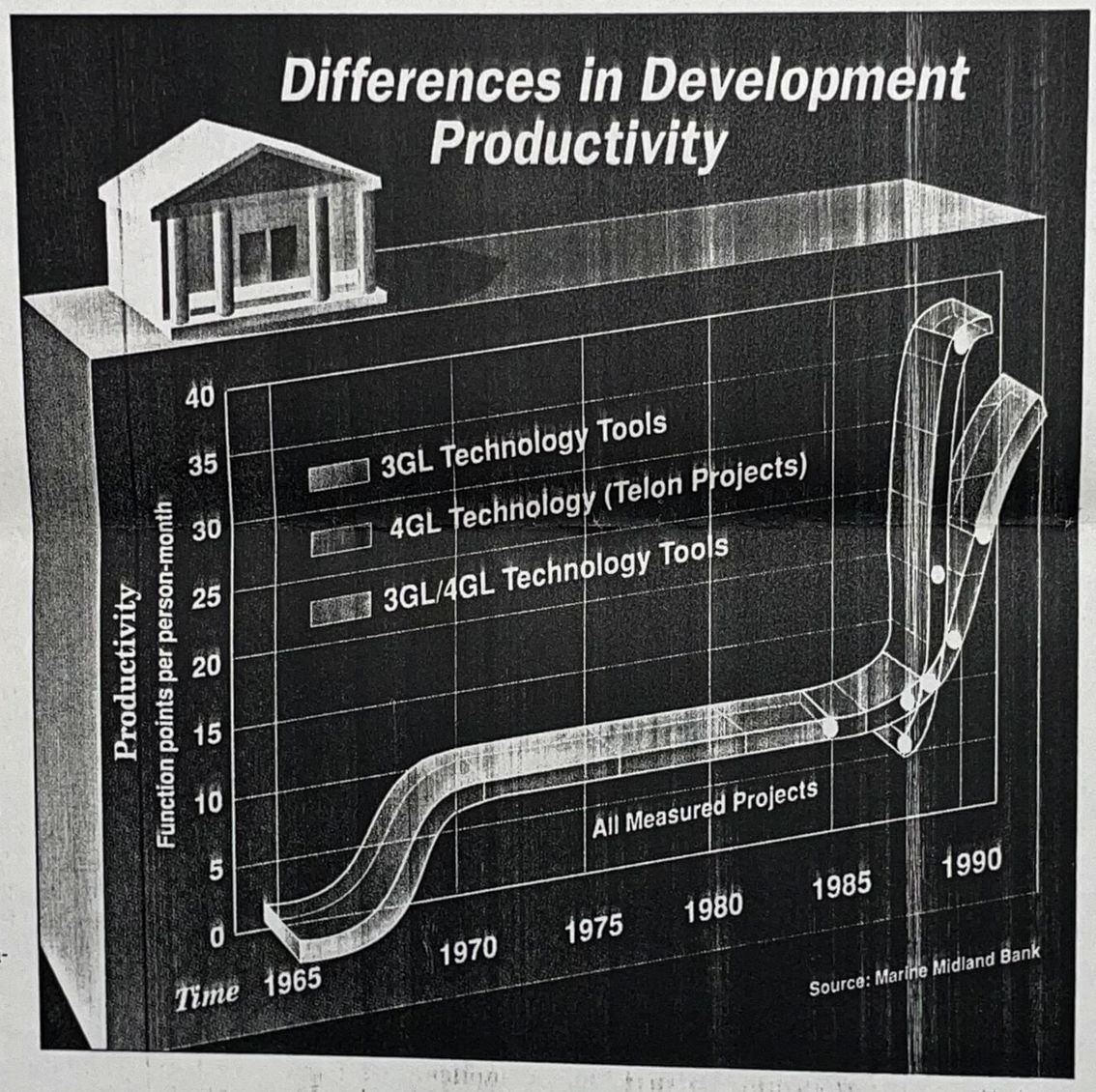
The method assigns a weight to each of these elements based on its complexity, totals the results, and then adjusts the result for factors estimating the internal processing complexity and general system complexity.

Function-point metrics are generally not appropriate for scientific or engineering computing or for systems in which complex algorithms must be coded.

It is useful to have a basis for comparing different projects, estimates or vendors' claims. Some individuals remember measurements such as equivalent lines of COBOL per person-day, relating this perhaps to their own experiences in programming. Other individuals remember function points per person-month. I recommend that IS executives measure and remember function points per person-month.

For systems of average complexity without elaborate user interfaces, one function point per person-month equals about five lines of COBOL per person-day.

For systems with human interfaces such as a mouse, icons and pull-down menus, the ratio becomes higher.



John Avakian

Periodic measurements of productivity enable managers to establish quality standards, identify areas in which improvements are needed and establish a program for improvement.

The U.S. average development productivity is five function points per person-month, corresponding to about 25 lines of COBOL per person-day.

Many organizations developing systems with COBOL, database tools and the traditional IS life cycle average about 40 lines of COBOL per person-day (about eight function points per person-month). Some have used productivity aids with otherwise manual COBOL development and have pushed their productivity up to about 60 lines of COBOL per person-day (about 12 function points per person-month).

The objective of today's power tools and techniques is to achieve substantially higher productivity numbers.

The figure shows measurements of IS productivity at Marine Midland Bank in Buffalo, N.Y. COBOL development had reached a plateau of about 12 function points per person-day across all measured projects.

The first introduction of 4GL technology using the Telon code generator from Pansophic Systems Inc. resulted in slightly lower productivity, but the staff rapidly climbed a learning curve, building to about 40 function points per person-day.

This is a fairly typical number for well-managed use of a 4GL/CASE tool with a code generator. Using an integrated CASE environment and a tighter methodology, the productivity figures are likely to increase.

Highly trained teams using integrated CASE tools and a RAD methodology can achieve much higher productivity figures. Some teams regularly achieve 1,000 equivalent lines of COBOL per person-day (or 200 function points per person-month) on projects of moderate complexity.

Measurements in the Du Pont Fibers division show skilled developers averaging 13.8 hours per function point with a traditional Yourdon structured methodology, while other Du Pont analysts were averaging two hours per function point using its "time box" methodology, which is a version of RAD.

Some large and bureaucratic application-development projects average only two or three function points per person-month, at a cost of more than $1,000 per function point. Some software-development companies charge more than $1,000 per function point. IS organizations typically average about $500 per function point. A reasonable target for a RAD project is about $100 per function point (corresponding to 100 function points per person-month).

CEO Productivity Report

A corporate president should monitor IS productivity because of its vital effect on corporate capabilities.

The CEO report should include five types of measurements:

- speed of development;
- development costs;
- development productivity;
- quality of delivered systems; and
- maintenance.

When this type of report is requested on a quarterly basis by the CEO, the IS manager is motivated to strive vigorously for the most efficient use of powerful development techniques.

Next week I will discuss the automated tools that are appropriate for use in a RAD life cycle. ■

The concepts embodied in RAD are described in a new volume in the James Martin Report Series. For more information on this volume, call (800) 242-1240. For information on seminars, contact (in the United States and Canada) Technology Transfer Institute, 741 10th St., Santa Monica, Calif. 90402 (213) 394-8305. In Europe, contact Savant, 2 New St., Carnforth, Lancs., LA5 9BX United Kingdom (0524) 734 505.