EMERGING TECHNOLOGIES ENRICH OPPORTUNITIES FOR SOFTWARE QUALITY IMPROVEMENTS

Joseph R. Schofield, Jr.
Sandia National Laboratories

The need for quality improvements in software engineering continues. A frequently quoted 1979 General Services Administration (GSA) survey described how only 5 percent of the software procured as part of that survey was used. More recent and well-publicized events at the Social Security Administration and the U.S. Patent & Trademark Office indicate that all is not well today, either. Software mismanagement is not limited to the government ranks. KPMG Peat Marwick recently revealed that 35 percent of its 600 largest clients had major runaway software projects. The software-triggered self-destruction of the Phobos I in September of 1988, dispels the notion that software problems are limited by national borders.

Historically, software engineers have relied, in part, on structured techniques, standards, peer reviews, customer sign-offs, and testing to validate the software development process. Still other techniques—maintenance programmers at development reviews, independent software auditors on development teams, shared testing responsibility, and separation of development and execution responsibilities—have provided some additional benefit.

Yet the experiences of others claiming new successes provoke further interest. One area of progress is emanating from applying the 3 R's: reverse engineering, reengineering, and reusability. Reverse engineering facilitates the return from one life cycle stage to an earlier stage (returning to design from coding or returning to specification from design). Reengineering implies ex-

"*A DATAMATION* STUDY FOUND ONLY 15 PERCENT OF PROGRAMMING CODE WAS UNIQUE, NOVEL, AND SPECIFIC—IN 1983!"

tracting the inputs from one stage of the life cycle (this approach may include reverse engineering tools) and applying automated tools to reconstruct a more supportable system. Reusability promotes the development of plug-compatible software modules that reduce the need to develop, test, debug, and support code that duplicates existing functionality.

Toshiba and DuPont are among the corporations that have reported success applying varying components of the three R's to software development. The use of these techniques is appropriate. A *Datamation* study found only 15 percent of programming code was unique, novel, and specific—in 1983! Literally dozens of three R products are available in today's market.

The three R's also significantly impact growing software "maintenance" concerns. While appreciating Swanson's characterizations of software "maintenance" as adaptive, perfective, and corrective, closer analysis leads to the conclusion that "maintenance" is actually development. Certainly adaptive and perfective software changes are, by definition, developmental. Corrective changes are the result of postponed development activities—specifically testing. Due to insufficient software testing, the code was prematurely granted production status while a development stage was incomplete.

"COMPUTER-AIDED SOFTWARE ENGINEERING (CASE) TOOLS ALSO HAVE A POTENTIAL TO IMPROVE SOFTWARE QUALITY."

Computer-aided software engineering (CASE) tools also have a potential to improve software quality. The literature evidences other meanings of the acronym CASE, including computer-assisted software engineering and computer-aided system engineering. Terminology aside, the complete systems development life cycle (SDLC) coverage of robust CASE tools enhances not only productivity, but quality as well. Many of these improvements are directly attributed to automated completeness and consistency checks through the repository, code generation, and testing facilities.

Naturally, most tools touted as CASE were available many years before its inception. Whether the marketers of these products are attempting to capitalize on the CASE movement is immaterial. What is important is the substance of their products, proven and reliable technological application.

The CASE movement is hardly exhausted. Although most segments of the market appear to be saturated with a range of tools, mergers will continue. The purpose

Continued

behind many of the mergers is to join vendors that complete each other's product lines. The most likely scenario is a union between a vendor that addresses the "front end" (prototyping, repository, design emphasis) of the SDLC with a vendor that addresses the "back end" (coding, testing, documentation phases). When pondering the effectiveness of CASE solutions, integration between product set is imperative for optimal application of the tools. The appeal to fragment or redirect the CASE market with I-CASE (integrated CASE) seems unnecessary. The distinction, not the concept, is better left ignored.

"NATURALLY, MOST TOOLS TOUTED AS CASE WERE AVAILABLE MANY YEARS BEFORE ITS INCEPTION."

The application of complete SDLC-coverage CASE tools is not for those with shallow pockets, though the return on investment can be great. This initial investment supports the notion that CASE implementations tend to increase, as does the size of the enterprise. The efficiencies related to production systems developed using third-generation languages deserve thunderous applause—especially from those sites crippled by 4GL implementations or bankrupt from underwriting hardware upgrades. Consider the other attributes of 4GLs: nonstandardization, DBMS dependency, general lack of portability, and it becomes obvious that the impact on software quality is largely pejorative.

Mind Your P's and Q's—The Software Engineering Flavor

The traditionally difficult task for children to distinguish between the letters "p" and "q" can be extended to productivity and quality in the software world. In software engineering, these two keywords are effectiveness and efficiency, respectively. Doing the right things (productivity/effectiveness) and doing the right things right (quality/efficiency) have been the lure of multiple generations of software vendors and associated folklore.

Software development attentiveness needs to be measured. Software metrics facilitate this increased vigilance. Quantifying the function points delivered and/or supported by the software engineer, budget, failure, or customer downtime are but a few indicators of the state of health of a software environment. Recognizing process improvement or the need for refinement are also substantial reasons for implementing metrics. Most organizations have indicated a desire to dedicate more effort to software measurement in coming years.

Attending to our p's and q's is more than a matter of measurement and statistical significance—it will also lead to improved perceptions. Perceptions do not take a back seat to statistical sampling in the eyes of the customer; rather, perception is reality. The National Quality Award Program, signed by then President Reagan, attests to the importance of quality measurement and perception. Officially designated "The Malcolm Baldrige Quality Improvement Act of 1987," the award program assesses seven categories, including leadership and customer satisfaction.

The ability to recognize and satisfy customer desires, or to inspire and motivate through leadership, is not indigenous. Fundamental p and q improvements begin in the educational process. Illiteracy in the U.S. is 26 times (by percent) greater than in Japan. When compared to Germany (again by percent), over three times as many U.S. citizens do not graduate from high school. Children in the U.S. rank 15th in science and mathematic skills for industrialized nations. While children attend school about 180 days a year in the U.S., it's 240 days in Japan, and 250 days in Korea.

In addition, p and q improvements are limited by the nonapplication of ingenuity and creativity in industry. According to an article in the *Wall Street Journal,* work rules and job restrictions account for 30 to 50 percent of the productivity differences between U.S. and Japanese workers.

The effect of these differences can be illustrated by national productivity increases over the past 10 years as reported by economist Lester Thurow. Compare the .8 percent increase in the U.S. to Germany's greater than 3 percent, Japan's greater than 4 percent, and Korea's greater than 12 percent.

"WHETHER THE MARKETERS OF THESE PRODUCTS ARE ATTEMPTING TO CAPITALIZE ON THE CASE MOVEMENT IS IMMATERIAL."

While p and q improvements are benefits of CASE implementations, the future holds still a greater potential through the application of artificial intelligence (AI). International Resource Development, Inc., Arthur D. Little, Inc., DM Data, Inc., and New Science Associates have each projected AI as a $4 billion industry in the early 1990s. In contrast, CASE was being projected in 1987 as a $1 billion industry by 1990.[1]

If those figures are not impressive, consider that AI analysts are receiving $46-$60 per hour, with industry-recognized consultants receiving $1500-$3500 per day. These and other figures were the focus of nine articles in the July/August issue of *U.S. Woman Engineer.*

Unfortunately, some software vendors are claiming their packages contain AI ingredients. Customized interfaces, training modules, help screens, and process edits do not constitute AI existence. The definition of AI in the *Encyclopedia of Computer Science* is more appropriate: "...a system is judged to have the property of intelligence based on observations of the system's behavior, if it can adapt itself to novel situations, has the capacity to rea-

Continued

son, to understand the relationships between facts, to discover meanings, and to recognize truth...."

Given this definition, AI systems:

- accumulate knowledge and gain experience;
- will not always solve a problem identically, rather accumulated knowledge base will provide better solutions over time;
- are not limited by IF ... THEN ... ELSE pruning algorithms;
- resolve ambiguities after accepting conflicting data;
- substantiate decisions through knowledge of self;
- develop solutions to problems not yet posed;
- study the environment and customers with an intent to improve; and
- are capable of responding with an "undecided" or "I don't know." AI knows its limitations.

Three significant obstacles to applying AI today include domain dependency, a natural language interface, and representation through predicate calculus.

Domain dependency implies that "worldly knowledge" beyond the subject being evaluated may not be applied to the resolution process. Even mere mortals can apply simple driving rules, after leaving the parking lot, to navigating a shopping cart through a busy supermarket.

"ACCORDING TO AN ARTICLE IN THE *WALL STREET JOURNAL,* WORK RULES AND JOB RESTRICTIONS ACCOUNT FOR 30 TO 50 PERCENT OF THE PRODUCTIVITY DIFFERENCES BETWEEN U.S. AND JAPANESE WORKERS."

Apparently unstructured and inconsistent rules in the English language will continue to delay the implementation of the natural language interface (NLI) or software engineering through vocalization. Together AI and the NLI could provide machines to simultaneously monitor the environment, recognize when it is being addressed, when the request is authorized, when the speaker is authentic, and when to ignore superfluous intermittent conversation (background "noise") regardless of its source.

Given ongoing abuses to the English language, it may be difficult to imagine AI being applied to generate or understand informal conversation.

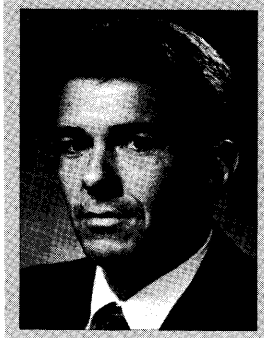
Expressing specifications, objectives, and approaches using predicate calculus is neither practical nor satisfying. An improved interface, within the guise of the NLI, is necessary before widespread application of AI is possible.

Yet AI is already credited with major success stories. Primarily, rule-based systems are at the core of these products. Rule-based systems, depending on their level of knowledge, can more clearly be classified as knowledge-based or expert systems. AI systems await availability.

The future is truly rich with opportunities for software quality improvements. While traditional structured techniques have helped, automated automation holds greater promise. CASE enhanced with AI characteristics will help meet tomorrow's challenges. The three R's will reduce new code requirements and extend the life of current systems. Finally, educational improvements are necessary if the U.S. is to continue as a major participant in the development and application of software engineering tools.

"GIVEN ONGOING ABUSES TO THE ENGLISH LANGUAGE, IT MAY BE DIFFICULT TO IMAGINE AI BEING APPLIED TO GENERATE OR UNDERSTAND INFORMAL CONVERSATION."

JOSEPH R. SCHOFIELD



Joe Schofield is a computer systems consultant at Sandia National Laboratories in Albuquerque, NM. His responsibilities include application of emerging technology into the software development process, software quality and metrics implementation, and coordinating various planning documents. His education includes an MS in MIS from the University of Arizona. Joe is also a faculty member at the College of Santa Fe.