SOFTWARE **STANDARDS**

Software standards in the global village.

Editor: George Tice
Mentor Graphics
8500 SW Creekside Pl.
Beaverton, OR 97005
Compmail+: g.tice

Are software standards wasted efforts?

George Tice, Software Standards Editor

Is the IEEE software standards effort making a significant contribution toward moving software development and maintenance from art form to engineering discipline? Is there sufficient payback for the individual and organization investment in IEEE software-standards development? Is there satisfactory coordination with other standards developers?

In January, I initiated an effort to identify current usage of the 11 approved IEEE software-engineering standards. *IEEE Software* published a questionnaire on standards' use. In addition, I declared that the ultimate test of software standards is their successful application to software development and maintenance in industry.

Questionnaire results. Measured by the response to the questionnaire, the IEEE software-engineering standards are failing this test. Only 20 readers admit to using at least one of the 11 approved standards. Of the 20, only two were from the leadership of the IEEE software-engineering standards working groups. These 20 were the only respondents to the survey.

The *Guide for Software Requirements Specification* was most popular; the *Glossary of Software-Engineering Terminology* was the runner-up. Nearly half the respondents were from the United States; the other half was from abroad. The majority reported embedded and control software as the type of software they developed.

Several comments were enlightening. One declared that the IEEE has provided his organization with a good platform for its multinational standardization effort. A second described the use of ANSI/IEEE Std 830-1984 as a vehicle for solving the scheduling and user-expectation problems between electrical and software engineers. A third explained that the standards require a basic culture in software engineering that most software developers do not have — thus, the standards are not understood and used, he said. Table 1 (see p. 90) shows the total response.

As a participant in the development of the software-engineering standards since 1981, I find this response discouraging. An easy answer is to fault the questionnaire. The conventional wisdom is that many organizations are successfully using the standards in their development of internal software-engineering standards. If so, let us hear from you.

Your colleagues volunteer many hours to the standards working groups. The dollar value for the development of a single standard is about $250,000. Both the individual and organizational investors need to know that they are in fact making a contribution to the software-engineering discipline.

Perhaps it is true that software standards are successfully applied only when they are mandatory.

The response to the questionnaire indicates that we are wasting our time and money in developing voluntary standards. Perhaps it is still true that software standards are successfully applied only when they are mandatory.

CASE coordination effort. Computer-aided software engineering is one major area where many people are preparing to spend a lot of energy, time, and money on voluntary standards efforts. For example, the IEEE Task Force on Professional Tools recently received IEEE Standards Board approval for a tools-interconnection working group, P1175. The task force has identified 240 existing standards that could affect tools and at least 10 other groups actively working on new CASE standards.

Coordination among the numerous groups working on standards for CASE is an emerging issue. As a result, a coordination meeting for CASE standards efforts organized by Elliot Chikofsky of Index Technology was held July 12 in Cambridge, Mass., as part of CASE 88, the Second International Workshop on CASE.

One hundred twenty-seven people attended the meeting on behalf of the IEEE task force, the ANSI X3H4 committee on information-resource directory systems, the US Defense Dept.'s Common APSE Interface Set for Ada, the European Strategic Programme for Research in Information Technology's PCTE environment standard, the European Computer Manufacturers Association, the CASE technical subcommittee of the Electronic Industries Association's EDIF standard, ISO committee SC7 (software development and system documentation), the Digital/Atherton tool-integration services proposal, the National Bureau of Standards, and the Software Productivity Consortium.

The primary purpose of the meeting, Chikofsky said, was to provide an opportunity for information exchange. Dave Sharon, the meeting chairman, said that the meeting met this objective and was a high-quality learning experience for all participants. Robert Poston, chairman of the IEEE Task Force on Professional Tools, accepted the responsibility for collecting and publishing status information for all the current CASE standards efforts. There was also a group commitment to meet again July 1989 in London during CASE 89.

Thus, the groundwork has been laid for future communication and coordination.

Continued on p. 90

SOFTWARE **STANDARDS**

Continued from p. 88

Table 1.
Software-engineering standards usage survey results.

Standard	Invoked as written	Tailored	Reference only	Total
ANSI/IEEE Std 729-1983 *Glossary of Software-Engineering Terminology*	4	4	8	16
ANSI/IEEE Std 730-1984 *Software Quality-Assurance Plans*	—	5	8	11
ANSI/IEEE Std 828-1983 *Software Configuration-Management Plans*	—	6	6	12
ANSI/IEEE Std 829-1983 *Software Test Documentation*	—	9	6	15
ANSI/IEEE Std 830-1984 *Software Requirements Specification*	2	10	7	19
ANSI/IEEE Std 983-1986 *Software Quality-Assurance Planning*	—	6	5	11
ANSI/IEEE Std 990-1986 *Ada as a Program-Design Language*	—	2	3	6
ANSI/IEEE Std 1002-1987 *Taxonomy for Software-Engineering Standards*	—	—	10	10
ANSI/IEEE Std 1008-1987 *Software Unit Testing*	1	1	9	11
ANSI/IEEE Std 1012-1986 *Software Verification and Validation*	1	3	8	12
ANSI/IEEE Std 1016-1987 *Software Design Description*	—	3	8	11

Type of software developed:		*Country:*	
Defense	6	United States	11
DP/MIS/Business	3	Italy	2
Embedded/control	11	Sweden	2
Scientific	2	Canada	1
Systems	3	Denmark	1
Other	4	India	1
		Mexico	1
		Spain	1

STANDARDS **BULLETIN**

Forth progress

X3J14, the technical committee developing an American national standard for the Forth language, has reported that it has passed proposals to ensure the language will not be restricted to specific hardware architectures. Forth was developed for 16-bit computers and earlier standards assumed a 16-bit architecture, said Elizabeth Rather, the committee chairman. The decision will mean that programmers can port programs across CPUs with differing word lengths.

Other decisions include support of one's-complement architectures (not just the common two's-complement) and an extension to define an interface between Forth and popular operating systems such as MS-DOS, Unix, VAX/VMS, and OS/2 (early Forth systems operated as stand-alone systems).

QUALITY**TIME**

Continued from previous page

representation of sequential and concurrent control with each refinement step, the approach supports a simple form of human verification, one analogous to that used when carrying out long division. Mills has reflected on the approach and pointed to some evidence of success within IBM's Federal System Division ("Keeping It Simple and Correct is Mills's Legacy," Soft News, *IEEE Software*, July, pp. 95-96).

Software testing. The traditional attitude has held, "The purpose of testing is to find as many program errors as possible before the product ships. It starts when the coding is done and ends when you run out of time. It's also a good place to put co-ops and new hires until they can move up to development."

But the emerging attitude asserts, "The purpose of testing is to certify the reliability of the product and to uncover as many deviations from the behavior expected by users as possible. Effective testing requires planning, discipline, creativity, technical training, experience, and adequate resources."

There are two points I want to make here.

First, testing is emerging today as a respectable engineering discipline in its own right. Research activity appears to be increasing, new textbooks are appearing monthly, efforts to establish standards are under way, and there has been an increasing emphasis on technical training in industry.

Second, there seems to be a steadily growing interest in the idea of testing as a tool for statistical quality control. The focus of this type of testing is less on uncovering programming errors than it is on determining whether the product meets user-specified reliability requirements.

While the use of statistical models for predicting reliability is likely to remain controversial for some time to come, hopefully the associated emphasis on user-oriented characterizations of program behavior will not (see "Applying Software-Reliability Models in Industry" by Yuzuru Suzuki and John Musa, QualityTime, *IEEE Software*, July, pp. 87-88).

I hope the majority of QualityTime readers share my optimism with respect to these trends. Perhaps some future contributors will identify other examples of changing attitudes or developments that provide some real encouragement.