DISCUSSION OF SESSION ON RELIABILITY AND WARRANTY OF NUMERICAL SOFTWARE

by

Wayne R. Cowell

Prepared for

TC-2 W.C. ON

Performance Evaluation of Numerical Software

Badem. Austria

December 10-16, 1978

U of C-AUA-USDOE

ARGONNE NATIONAL LABORATORY, ARGONNE, ILLINOIS

Operated under Contract W-31-109-Eng-38 for the

U. S. DEPARTMENT OF ENERGY

[illegible]

DISCUSSION OF SESSION ON RELIABILITY AND WARRANTY OF NUMERICAL SOFTWARE*

Wayne R. Cowell
Applied Mathematics Division
Argonne National Laboratory
Argonne, Illinois

Existing numerical software has originated, for the most part, in publicly funded research and development projects at universities and government laboratories. It has been shared informally within the research community, distributed by various government-funded centers and, increasingly, is being incorporated into proprietary libraries and disseminated through business enterprises. Numerical software is thus emerging as a product, which is represented as having certain characteristics by vendors, for the purpose of promoting its sale or licensing to users, for whom the performance of the software may be critical. The attendant questions about protection of intellectual property, warranty of performance, and liability for misrepresentation become of concern to software suppliers and users. The search for constructive approaches involves the law and computer science, both highly technical subjects. When they become interdependent, one is struck by the ways in which their respective complexities reinforce one another. It appears to this discussant, upon reading the papers in this section, that the application of the law to issues arising from software technology is at a very early stage of development. The framework is there but progress toward the establishment of just relationships between sellers and buyers will hinge on further experience in and out of court, clarified by research in computer science.

Niblett discusses various means of protecting property rights in numerical software namely, patents, trade secrets (confidential information), copyrights, and trademarks. These are applicable or not under various circumstances but it would seem that experience in the courts is still too limited to permit easy prescriptions. Thus, for example, the Patents Act of 1977 (U.K.) obviates the possibility of patenting computer programs as such but does not settle the question of patentability of a device which realizes a program in a novel way, such as a microelectronic chip. Battiste notes, in his paper, that chips with elementary function capability are nearly at hand, presaging a mathematical software to mathematical firmware movement. Thus, it seems likely that the question of patenting programs on chips may soon attract considerable attention.

Niblett points out that the exposure of software required by thorough testing may be incompatible with the protection of that software as a trade secret. He argues that the most effective way, at least in the U.K., of asserting intellectual property rights in numerical software is by means of copyright laws. Responding to a question from the floor, Niblett stated that civil cases involving copyright of programs have not yet been considered by the courts in the U.K., but there have been a number of out-of-court settlements on the basis that copyright subsists in a computer program and has been infringed.

The impression on this discussant is that the traditional means of protecting intellectual property are not well adapted to the protection of computer programs. Programmers do not yet enjoy the same degree of protection of their work as do inventors and artists. Like the inventor's work, a computer program is sterile unless it is used but its use (execution) requires that it be replicated, infringing, in some sense, the right to copy. Therefore protecting the program as if it were a literary work (as Niblett says is possible) is not fully satisfactory. Consider how much source code, intended for submission to a compiler (and thus

*Work performed under the auspices of the U.S. Department of Energy, with partial support from the U.S. National Science Foundation.

copied), is published in books and journals for which the right to copy is reserved by the publisher. At the same time, patent law is not applicable to algorithms, and thus it appears that only firmware will have reasonable protection, in the traditional sense, for some time to come.

All three authors in this section treat the question of guaranteeing the performance of software products. Battiste's emphasis is on the computer science-related difficulties of providing reasonable warranties. Tapper and Niblett examine the legal ramifications of liability for misrepresentation of software. The effect is sobering, especially if we accept that the body of law reflects societal expectations of software performance. Where these expectations are unrealistic, it means that the public view of computing is unrealistic, probably because computing experts have oversimplified the presentation of computing to the public. (They might not otherwise have been heard at all.) These papers lead us to believe that the use of the law will become more reasonable and just when the public has a better understanding of computing realities.

Tapper illustrates this point in his discussion of the legal proof of negligence. It is for the plaintiff to prove negligence when negligence is alleged. He can sometimes rely on a doctrine which asserts that the defendent has the burden of explaining any accident which has occurred contrary to normal expectations in a situation in which he (the defendent) might have been expected to be in control. Is defective software contrary to normal expectations? The question has not been tested in court. Tapper goes on, "So long as people remain convinced that computers have an existence and personality of their own different from that of their operators and programmers so will it be common to ascribe error to the intrinsic fallibility, unpredictability, and malevolence of the machines themselves rather than to the negligence of the human beings concerned."

As is apparent from other sessions at this conference, there is beginning to emerge from research efforts a sense of what acceptable software performance means. Unfortunately, much of this understanding is highly technical, quite dependent on the area of computation, and difficult to communicate in lay terms. The challenge is to translate this technical understanding into guidelines for software performance that will enable reasonable people to have reasonable expectations. We can hope that, eventually, the application of the law will incorporate this increase in reason.

Tapper's paper considers liability for computer software misrepresentation both in the law of contract and in that of tort. He cites a number of cases in which some question of liability for misrepresentation arose and he discusses their implications for contracting and for seeking remedies for alleged wrongs. There are guidelines here that will serve as indicators for users and vendors but it is clear that legal counsel should be sought before signing, selling, or accusing. Indeed, the Misrepresentation Act 1967 (U.K.) has been described as leaving the situation "almost incredibly complex" as it provides "a gamut of five different causes of action from which an action for damages might result, some sounding in tort and others in contract according to the circumstances of the misrepresentation." One may encounter the need to distinguish between "recklessness" and "negligence." Again, one might have entered into a contract with a representer after a misrepresentation was made. Attempting to escape liability, the representer might claim that he believed "on reasonable grounds" that the statement was true. Computer scientists, upon reading the paper, may ask what constitutes reasonableness in the representation of computer software. They will be impressed (though, on reflection, not surprised) by the extent and intricacy of the legal machinery that deals with liability. They will also recognize terms (for example, "precise specification of the performance of a program") that must derive meaning from their science if the machinery is to be successfully applied in their field.

The extent of liability is a serious concern. Tapper points out, for example, that a transposition of two characters in an operating system could easily go undetected through all reasonable trials [that word "reasonable" again!], and still in operation lead to disastrous errors causing enormous expense to users. This discussant was comforted when Tapper cited a case in which the court was very careful to focus the extent of liability narrowly, precisely because of the potential for almost unlimited liability through the use of faulty software. Too broad an extent of liability would place a heavy burden on programmers.

There appears, to this discussant, to be a danger that vendors will be hamstrung by capricious liability suits (or the threat of them) until their best efforts to represent complex software realities are distinguished from fraudulent, careless or stupid misrepresentation. Battiste, in his paper, expresses this danger and argues strongly for the development of a technical basis permitting the description of adequate software performance in terms suitable for reasonable warranties.

Battiste characterizes various areas of mathematical computation from special function approximation to multivariate analysis in terms of suitability for reasonable warranty. He singles out special function approximation as the area in which software performance is best understood by virtue of an advanced state of testing methodology. Therefore, he asserts, special function software may be the first for which reasonable warranties are offered. However, Battiste warns that this does not imply that warranty is possible for programs constructed from special function kernels. In this regard, he quotes J. T. Schwartz on the subject of integrating large collections of simple programs into coherently functioning wholes. Battiste also reminds us that the perfection of contracts is not the only purpose served by an ability to describe the performance of numerical software; the software that is cast in firmware will be the best only insofar as we are able to measure and describe what is best.

These papers should serve to raise the consciousness of software experts about the social implications of their work. The need to improve the quality of software and to express clearly what quality means take on new urgency when seen in terms of satisfying the demands of the law as well as in scientific terms. These papers should also encourage vendors and users to insist on well-drawn contracts that clearly define their responsibilities to each other, within the limits of technical understanding.