Appeal to NSF proposal reviewers

I write on behalf of the Advisory Committee for the Division of Computer Research of the National Science Foundation in order to alert Computer Society members to an extremely serious problem:

The computer research community is failing in its responsibility to review research proposals in a thorough and timely manner, thereby placing the research funding process in jeopardy.

The problem has not yet caused funding errors, as I will explain, but the situation is urgent, and your help is needed. To understand the problem, let us review the review process.

The National Science Foundation receives "unsolicited" proposals for research. This term distinguishes the NSF proposals from those received by other agencies in response to specific requests, and it emphasizes that, at the NSF, the investigator alone determines what to study and how to proceed.

Once the proposal is received, acknowledged, numbered, routed and screened by NSF, it is assigned to a "program" or topic area, such as "software engineering." The program director, an NSF staff scientist, selects six or more experts from the computer research community to serve as "peer reviewers" for the proposal. Reviewers are selected to provide a variety of viewpoints: Those who are expert on the specific topic would likely speak to the proposal's interest, correctness, and importance; those whose expertise is more peripheral might speak to the overall value of the research area; others might comment on the practicality of the proposed work or the relationship to known theoretical results. The program director, who makes the award/declination decision, chooses reviewers so as to be able to judge the proposal accurately and in the proper context.

Copies of the proposal are sent to the reviewers together with reviewing guidelines and a deadline for completion. It is then that the problem begins. Many reviewers never respond at all. Others keep the proposal for months and then return it, unevaluated. Still others, after keeping the proposal for months, respond with such a superficial review that it is useless; e.g., "I believe this scientist has done pretty good work, so fund whatever is being proposed."

Even those who do provide comprehensive reviews frequently take six to nine months to do what is usually a half-day task. Some reviewers rate the proposal highly but give only disparaging comments; others give only laudatory comments but rate the research very low. To be sure, many computer researchers provide prompt and incisive reviews, but the committee finds that most can do better.

Has the funding process been harmed by this poor reviewer performance? To find out, one must examine the proposals and reviews independently to determine if the funding decisions were justified. Such an examination has recently been done for each of the eight programs of the Division of Computer Research as part of the mandated, triennial oversight review process. The Advisory Committee, twelve scientists selected from the computer research community, has found that:

- A. Ellis, Xerox PARC; G. H. Golub, Stanford; A. N. Habermann, CMU; K. W. Kennedy, Rice; L. H. Landweber, U. of Wisconsin; W. G. Lehmann, U. of Massachusetts; N. A. Lynch, MIT; R. E. Miller, Georgia Tech; R. F. Riesenfeld, U. of Utah; B. J. Smith, DENELCOR; L. Snyder, U. of Washington; D. Waltz, Thinking Machines.

March 1985
community, has performed oversight reviews as part of its last three semianual meetings. The findings: Nonresponsive or superficial reviews are common in all areas of computer research; the DCR program directors try very hard to compensate for this situation, sometimes making a decision based on as few as three reviews; no evidence was found of large-scale funding errors, but the strain on the funding process was obvious to the committee.

Although no similar, comprehensive analysis has been or is likely to be done for the analogous problem of untimely and superficial journal refereeing, the committee members' own experience as authors and editors forced them to conclude that the failure of the community to refer papers for scientific journals in a professional manner is at least as serious as its failure to review proposals professionally. In addition, the Computer Science Board, an organization of computer science department chairmen representing perhaps a broader sample of opinion, has come to a similar conclusion.

What exactly is the researcher's professional responsibility? Obviously the fact that six scientists are asked to review two-year grant proposals must carry an implied obligation to review six proposals over two years, just as each paper submitted to a journal implies an obligation to referee two or three papers. But even when we are not the immediate beneficiaries, being benefactors must surely serve our interest in the health of the science.

What can be done? The committee has suggested some minor administrative changes for DCR, e.g., protecting the responsible reviewers from an avalanche of requests. The Computer Science Board has also issued a statement of concern and recommended discussing the matter in faculty meetings. Other suggestions are needed. But in the final analysis, we, the overworked computer researchers of the country, must assume our professional responsibility to review our colleagues' proposals and referee their scientific papers promptly and conscientiously. It won't do to plead that we are too busy preparing new proposals or writing important scientific papers because, when they are done, no one else will have time to review or referee them. As Pogo observed, "We have met the enemy, and he is us."

On replacing the term "software maintenance"

In the January 1984 issue of Computer (page 88), John A. Grosberg attacks the term "software maintenance" on the ground that it "does not accurately denote the activities of that phase." In the November 1984 issue (page 5), Greg DeWilde ingeniously defends the term and Grosberg responds. We would contend that a noun does not have to denote all the contents of a thing; it just denotes the thing. That software maintenance has an agreed denotation is suggested by its use in the titles of four books. These books broadly agree in enumerating the activities of software maintenance. As co-authors of one of these books, we decided to stick with the common denotation. If you read the book, we'll take care of the connotations.


Girish Parikh
1787-B W. Touhy
Chicago, IL 60626
Nicholas Zvegintzov
141 St. Marks Place #5F
Staten Island, NY 10301

More "software maintenance"

The software life cycle continues to provoke questions. A recent one disputes the existence of a final phase of that cycle, commonly called "software maintenance." This view may contribute to some of the ongoing misconceptions about maintenance in the software circles. Figure 1 illustrates software maintenance as a full period consisting of the duration of existence of a birth-to-rebirth/death and not just as a phase of the SLC. The ?-maintenance square suggests that every

Lawrence Snyder, Chairman
Advisory Committee for the
Division of Computer Research
National Science Foundation

Figure 1. Phases of the software life cycle. Courtesy of Petrocelli Books.
new development must first go through a “maintenance phase” (in old terminology) before installation. In practice, however, a given system is subject to two or more maintenance activities before death or scrapping: A given maintenance activity/project may include analysis, design, coding, testing, etc., all phases of the SLC.

The auto-repair analogy of Grosberg is, contrary to his intent, illustrative of the problems of software maintenance. Auto maintenance may alter the performance of the original model. An auto repair consists of more than “replacing parts.” Some existing components may be changed or retooled; some foreign components may be introduced. In short, software maintenance shares some of the problems and goals of the older engineering disciplines.

Furthermore, it is misleading to argue “that the only qualification for a ‘software maintainer’ is the ability to type” for who, then, codes the “new” instructions? Who tests the upgraded system? Who installs it?

The word “maintenance,” however, is not absolute; it is, therefore, subject to change. What the critics have so far failed to do is find a semantically sound substitute term. A semantically sound terminology must be expressive of a thought, descriptive of a function, and suggestive of professionalism. Such substitutes as “continued development,” “continuation engineering,” “evolution,” etc., simply fail this test; they are, therefore, undesirable.

Lem O. Ejiogu
Lemcomm Softgram
2621 N. Emmett St.
Chicago, IL 60647.