Parnas: SDI "red herrings" miss the boat

To the editor:

From its title, "Can software for the Strategic Defense Initiative ever be error free?" to its conclusion, "...the shield will leak," Ware Myers' article in the November 1986 Computer discusses red herrings as if they were real issues. Such articles help SDI supporters to distract the public from what SDI critics, such as myself, are really saying.

The observation that the software would not be error free or perfect is trivial. The software that I use to write this letter is not error free but is obviously useful. No serious critic based his statements on an assumption that perfection was required. A statement about "error free" software does appear in the unclassified portion of the Fletcher Report. One SDI officer claims to have written those words while drunk.

As one who is familiar with some of the military software that the US now uses, I never would seriously consider requiring that SDI be "leak-proof" under all circumstances. I don't even expect that of my raincoat. President Reagan is said to make such statements, but no experienced engineer takes them seriously. Phrases such as "perfection," "leak-proof," and "Astrodome" have been introduced into the discussion by SDI supporters.

We all know that weapons, like other tools, don't have to be perfect to be useful. The arguments used by SDI supporters allow one to conclude that, for any value of N, a shield with an effectiveness of N percent would be useful. For example, a partially effective shield could provide strategic parity or superiority even if the other side has more warheads. Many engineers working on SDI have substituted such goals for President Reagan's.

However, such arguments are based on the assumption that we know, with reasonable confidence, what N is. We have to be very confident that N won't turn out to be zero or much lower than expected when the system is really needed. All of the imperfect devices on which we rely have that property; we are quite confident that they won't fail catastrophically. My raincoat may leak in heavy rains, but it is not made of water soluble material that dissolves in the first real rain. My car may not get me to work on time, but I have reasonable confidence that it will not explode when I start it. Were that not the case, I would not use it.

In fact, those who talk about specific leakage rates for SDI are guilty of gross oversimplification. The leakage rate is obviously going to depend on the structure of the attack and the countermeasures used by the attacker. A single figure, without a clear statement of assumptions about the threat, is meaningless and misleading. Those who quote numbers such as "90 percent" have no basis for them.

My papers did not conclude that there could be leaks. My assertion was that the system would not be trustworthy, that we would never have any confidence that the system would not fail. No SDI supporter has explained how they would establish confidence in such a system. Instead, supporters reply that it does not have to be leak-proof.

In fact, all of our experience with easier software projects supports my personal feeling that the likelihood of a catastrophic failure in an attack by a serious and sophisticated enemy is close to one. Mathematical theory confirms what we know from practical experience. Small errors in software do not necessarily have small effects. There is no reason to believe that a small error will simply increase the "leakage rate." Some big errors may not matter at all, but some small ones can be disastrous. Myers quotes Fred Brook's excellent question, "How good is good enough?" I think that the answer is clear. To be good enough, we have to know, with high confidence, how good it is. To be good enough, we must know that the system will not fail catastrophically.

Myers' discussion of Safeguard experience is misleading. He cites a low error rate after deployment and ignores the fact that this system was, thank goodness, never used. It was subject to "demonstration tests" and used, briefly, in peacetime. This is analogous to throwing a glass of water at a raincoat and noting that most of it drips off. My sweater passes that test. Safeguard used nuclear tipped missiles, but the effect of nuclear explosions on the data processing capability was never tested. As Brooks pointed out in his essay, "Plan to Throw One Away," the serious errors in software are discovered in real use, not in simulation or demonstration tests. His observation is especially relevant in situations where the characteristics of the problem are determined by a sophisticated enemy who is deliberately trying to cause such errors. Safeguard data does not tell us anything about the errors that show up when you first actually use a system for "production."

In my view, the whole discussion of error rates is pseudoscience. Quantitative arguments always seem more professional than qualitative ones, but they are only useful when the numbers are meaningful. In the case of error rates, a little thought will show that we don't even know how to count errors. If I write a program on the assumption that an array has bounds [0:99] and testing shows that the bounds should have been [1:100], how many errors is that? There might be 88 places in the code that have to be changed. We might find those places, a few at a time in a sequence of 35 tests. Do we count that as a single error, 35 errors, or 88? Often errors in programs can be corrected in several different ways. In the example, I could change the declaration, or change 88 statements in the program. Is that one error or 88? When you look at real examples, you find that the error counts are quite arbitrary; they depend on how the errors were discovered and how clever the programmer was when he made his corrections.

Error statistics make excellent diversions but they do not matter. A low error rate does not mean that the system will be effective. All that does matter is whether software works acceptably when first used by the customers; the sad answer is that, even in cases much simpler than SDI, it does not. What also matters is whether we can find all the
"serious" errors before we put software into use. The sad answer is that we cannot. What matters, too, is whether we could ever be confident that we had found the last serious error. Again, the sad answer is that we cannot. Software systems become trustworthy after real use, not before. They become trustworthy when the operating conditions are known and stable, not when they are unknown and subject to change by an opponent.

The architectural issues raised by the SDI supporters and reported by Myers are also irrelevant diversions. Some claim that progress has been made by switching from a centralized architecture to a decentralized one. Examination of the original study, the Fletcher Report, shows that SDI designers always had planned a decentralized architecture. Some claim that a hierarchy modeled on a military command structure is an improvement over the Fletcher design. Examination of the Fletcher Report shows that they, quite wisely, rejected such a hierarchical scheme as having an Achilles heel, the computers at the root of such a hierarchy.

All of the architectures that have been discussed propose a collection of subsystems, each controlled by a large, and largely untestable, software package. Confidence in either the individual packages, or their successful cooperation, cannot be established. The mathematical properties that make software the biggest source of trouble in large systems are shared by all software systems. Physical distribution is often useful but it does not simplify the development task.

I am happy to see professional journals such as Computer discussing the SDI software, but I hope future conferences will avoid the trap of discussing the non-issues introduced by politically motivated persons.

David L. Parnas
Queen's University
Kingston, Ontario

Author: But the public has been "hooked"

To the editor:

I am pleased that Dave Parnas—the originator of the thought that SDI software will present serious problems—has seen fit to expand his argument as a response to my article. My article had a narrow focus: Will there be errors? Essentially all authorities agree that there will be errors, but there is one authority, namely the President, who has sold SDI to the public on the basis that it will render nuclear weapons "impotent and obsolete."

Some 60 to 70 percent of the people are said by polls to support this program. I think they expect perfect protection. A dozen or so weapons getting through as a result of errors, destroying most of a dozen or so areas, would not be considered by the average citizen to be even adequate protection, let alone perfect. The fact that all the technical authorities in the engineering and defense communities would be pleased with a system of such high capabilities would cut little ice with the average citizen.

My article gathered up evidence from both those opposing SDI and those working on it and concluded that there will be errors. The fact of errors changes the nature of the debate on the political level over whether we should have such a system. Given the fact that there will be errors, it is then up to the citizens, the political community, strategic thinkers—whoever—to consider next steps. That is probably a discussion outside the area of expertise of Computer.

Ware Myers
Claremont, Calif.