yardstick", has not been impressive. Researchers have discovered that the human thought processes are much more complex than originally surmised. Humans make use of a wide range of knowledge in functions such as pattern-recognition, generalization, learning, and strategy development. This knowledge is so wide-ranging that it has still to be fully catalogued by researchers, and so computers cannot yet come close to human performance in this area.

Another limitation is that when a computer breaks down, more often than not it becomes totally useless:

> The notion that it will merely be a little sluggish or inaccurate derives from experience with other kinds of devices where a worn wheel or axle may indeed degrade performance. When something goes wrong with a computer, however, it tends to go berserk, not degrade.

Ornstein challenged the assertion that, even if Star Wars computers broke down, 95 per cent of incoming missiles would still be intercepted. He argued that in fact none would be intercepted in the event of a breakdown. Some systems are designed to prevent this sort of total breakdown. In these systems the effects of errors are somewhat alleviated by isolating various parts of the program and providing a minimum of restricted interconnections between the parts. Unfortunately, even these modular systems cannot be completely protected from breakdown. Furthermore, computers are notoriously unpredictable in interpreting unanticipated data. For example, sensors have interpreted the moon rising as an incoming missile. An additional limitation is the personality of the human operator. Although humans are an integral part of any operating system, human reactions are frequently ignored by systems designers and superseded by military protocol.

Ornstein maintained that the most crucial limitation of computers, however, arises from the inherent imperfectability of the software. Most of a computer's design is contained in the software and when a computer breaks down this is frequently the source of the problem. While computers are almost infinitely flexible because of their software, they are also easy to program incorrectly. Superficially it is easy to fix software problems, but the more complex a system becomes the more difficult and expensive it is to deal with these. In fact, a whole technique of "software maintenance" has arisen to deal with problems as they occur. John Shore, the author of *The Sachertorte Algorithm*, has argued that if a car needed as much attention from the manufacturer as computers do, then the car would probably be called a lemon!

In order to explain the fundamental problems that software imposes on systems, Ornstein described the two basic steps of software construction. First, the researcher studies a problem in the real world, decides which aspects of it are relevant and devises rules, then formal specifications, to govern the behaviour of the computer. Second, the researcher takes these