Software Fault Tolerance: Has it Arrived Yet?

V. S. S. Nair
Department of Computer Science and Engineering
Southern Methodist University
Dallas, TX 75275

Recently, it has been observed that the continuous operation of a system is hampered more frequently by software failures than by hardware failures. The increasing performance requirements and the complexity of software systems are not helping the situation either. Even though many techniques have been proposed and developed, state-of-the-art in software fault tolerance lags behind that in hardware fault tolerance. This can mainly be attributed to the larger number of erroneous states that software systems can assume when compared to hardware systems. Further, one observes that the lackadaisical treatment of the problem in the past has also contributed to the predicament. Until recently, software fault tolerance has mainly been an academic affair. Now that software has become central to the operation of most of the electronic systems, the industry community has started emphasizing on software reliability and fault tolerance. The realization that a fully tested software is an unfulfilled dream and measures have to be taken to tolerate software design faults once the product is out in the field, brings in much needed attention and enthusiasm.

Fault avoidance has been the traditional design approach for developing reliable software. Unfortunately, as mentioned earlier, the number of faults are potentially innumerable and avoiding all those faults at the time of software development is virtually impossible. In order to supplement the fault avoidance techniques, the software is subjected to elaborate test procedures so as to detect and remove remaining faults as much as possible. With the knowledge that there may still be faults present even in the critical sections of the software, a host of techniques have been proposed for tolerating such faults. The techniques include Recovery Block Schemes, N-version programming, Self-Checking programs and Exception Handling. The panelists discuss the merits and limitations of various existing approaches.

Dr. Vicki Rainey from the E Systems Inc. shares some of her experiences with software defects in large projects. She points out the need for changes in software process in order to address the problem of software reliability and fault tolerance. Prof. Bastani describes the concept of check-pointing and recovery which are essential components of any fault-tolerant software. Further, he discusses the issues related to real-time systems. Prof. Kim presents his view of software fault tolerance calling it a Long Dream. He also presents techniques for fault detection and recovery in real-time systems using distributed recovery blocks.

In addition to presenting some statistical studies performed on real systems, Dr. Luke describes an implementation of software fault tolerance at the operating system level. A quantitative evaluation of software fault tolerance is presented by Prof. Joanne Dugan. She presents various models for recovery blocks, N-version programs, and self-checking programs to illustrate the reliability improvements that can be achieved through fault tolerance.