Fault-Tolerant Software for Real-Time Applications

H. HECHT

The Aerospace Corporation, El Segundo, California 90245

To deal with hardware reliability requirements of critical real-time computer applications, fault-tolerance provisions have become a widely accepted practice. Failures of software executing on these computers is equally critical, and the extension of fault tolerance to software is therefore desirable but it needs to be implemented in a specific manner. Redundancy in fault-tolerant software requires programs that are deliberately different from the original ones which they are intended to back up. Error detection and rollback provisions must be as independent as possible of the software segments which they protect. The recovery block concept pioneered by Randell meets these requirements.

Skeleton routines are presented that illustrate the application of the recovery block to real-time programs, particularly those dealing with navigation and attitude control. The concept is seen to be compatible with certain ad hoc fault-tolerance techniques that are currently employed. A technique for reliability analysis of the resulting software system is developed. While specific software failure data are unfortunately not yet available, the exercise of this reliability model with a range of hypothetical failure rates shows that a very appreciable desensitization of the overall computer performance to software errors is possible by use of these fault-tolerance provisions. Economic factors of applying fault-tolerant software are discussed. Memory cost for the additional software segments is an obstacle at present but will probably shrink to insignificance in the future. A positive approach to the fault-tolerant software concepts seems warranted. A program for research to more fully explore the capabilities and limitations is suggested.

Keywords and Phrases: Fault-tolerant software, independent test and evaluation, primary and backup modules, reliability, reliability models

CR Categories: 4.30, 4.35, 6.0

INTRODUCTION

Techniques for dealing with hardware faults in computers for critical applications have been under investigation for a considerable time [1]. General methodologies have recently been published [2, 3], and the field is served by an established forum of annual meetings [4]. The emphasis on hardware fault tolerance is, of course, motivated by the inevitability of failures in physical components and more specifically by the failure rate of computers of a reasonable performance level which inhibits their single-string employment in applications requiring uninterrupted service for long periods of time. However, the true requirements of these applications are for continuity of correctly computed output, and that implies correct performance of both hardware and software. At the same time as remarkable progress has been made in both reducing hardware failure rates and developing fault-tolerant architectures, the user community has become increasingly aware that perfect software reliability cannot be