Abstract

Software fault tolerance is an important criterion for the dependable systems, especially in real-time and critical systems. There are few techniques that are used to implement fault tolerance in software, such as the most two common techniques: “N-Version Programming” and “Recovery Block”, also there are other driven techniques from these two techniques, as well as, other supporting methods like Exception Handling. Development programs must consider the development risks associated with using conventional software fault tolerance techniques that theoretically can outcome in a better system; but also, that could drive the entire effort of the development to fail because of the design team inability to manage the system complexity within a reasonable cost and time frame. Also, these conventional techniques cannot always guarantee producing a correct or an acceptable output. So, this paper proposes to design a fault tolerance technique that consists of two layers: the first layer is the special layer that derived from the other known techniques in a way that use the positive characteristics of these techniques, with the consideration of keeping the complexity of the system in minimum degree. This layer can be named the 2-Version Software with Acceptance Test Support. The other layer
is the general layer that can be used with the software fault tolerance technique that proposed in the first layer or with any other “software fault tolerance” techniques. The second layer propose is the design of a software fault tolerance mechanism that concerns on the use of unusual (intelligence) ways for system recovering from design faults, also allowing the system operator to interfere in the process of system recovering. The developed mechanism will be used to support the operation of the conventional “software fault tolerance” techniques.

References


Index Terms

Computer Science Software Engineering

Keywords
Software Fault Tolerance Techniques, N-version programming, NVP, Recovery Blocks, RB, N-version software, NVS.