* ***Topic***
* ***Abstract***
* ***LR for justification of problem statement***
* ***Problem statement***
* ***Methodology***
* ***Results & Discussion***
* ***Conclusion & Future works***
* ***References***

***N*-version programming** (NVP), also known as **multi-version programming**  is a fault tolerance technique where multiples version of software system are developed, these versions are independently generated from the same initial specifications. With N-Version Programming, NVP, independent development teams use the same specification to generate multiple implementations. ♦ During development the design teams are kept separate and do not share their designs nor do they discuss the specification’s meaning with each other. In this technique the design teams uses different algorithms and different programming languages to produce multiple versions. These versions are either run sequentially in a single processor or parallel in a loosely coupled processor, then the output simultaneous It is hoped that by performing the N designs independently, the same mistakes will not be made in all the versions The voting module will be able to detect a fault because the same fault is not expected to occur in all the .[Sunita]

How N-version programming works

the decision of output correctness is based on the comparison of all the outputs.

modules. In theory, this technique should be able to detect and recovery from most software faults. However it is very expensive to produce N independent designs and implementations of a software module. Moreover it is not clear if the n-modules will not have correlated faults

RECOVERY BLOCK

This technique was evolved as a result of first long term systematic investigation of multiversion technique initiated by Brian Randell in early 1970s [4]. In this technique, alternate software versions are organized in a manner similar to the dynamic redundancy (standby) technique in hardware. It’s objective is to perform runtime Software Fault Tolerance detection by an acceptance test performed on the results delivered by the first version. If the acceptance test is not passed, state is restored to what existed prior to the execution of that algorithm and execution of an alternate version on the same hardware is followed. Recovery is considered complete when acceptance test is passed. Checkpoint memory is needed to recover the state after a version fails, to provide a valid starting operational point for the next version (Fig 1).

The Recovery Block Scheme (RBS) technique [8,9] combines the basics of both the checkpoint and restart approach with multiple versions of a software component such that a diﬀerent version is tried after an error is detected. Checkpoints are created before a version executes. Checkpoints are needed to recover the state after a version fails to provide a valid operational starting point for the next version if an error is detected. The acceptance test need not be an output-only

197

G.K. Saha

test and can be implemented by various embedded checks to increase the eﬀectiveness of the error detection. Also, because the primary version will be executed successfully most of the time, the alternates could be designed to provide degraded performance in some sense (e.g., by computing values to a lesser accuracy). Actual execution of the multiple versions can be sequential or in parallel depending on the available processing capability and performance requirements. If all the alternates are tried unsuccessfully, the component must raise an exception to communicate to the rest of the system its failure (or crash) to complete its function. Note that such a failure occurrence does not imply a permanent failure of the component, which may be reusable after changes in its inputs or state. The much possibility of coincident faults is the source of much controversy concerning all the multi-version software fault tolerance techniques.