New techniques are often proposed to overcome the limitations associated with previous techniques, to provide fault tolerance for specific problem domains, or to apply new technologies to the needs of software fault tolerance, while attempting to maintain the strengths of the foundational techniques.

- variants of the N-version programming (NVP) technique
- resourceful systems,
- the data-driven dependability assurance scheme,
- self-configuring optimal programming (SCOP)
N-Version Programming Variants

- Numerous variations on the basic NVP technique have been proposed.
- These NVP variants range from simple use of a decision mechanism (DM) other than the basic majority voter to combinations with other techniques (the consensus recovery block (CRB) and acceptance voting (AV) techniques) to those that appear to be an entirely new technique (for example, the two-pass adjudicators (TPA)).
In this section, we will examine one such NVP variant, the NVP-TB-AT (N-version programming with a tie-breaker and an acceptance test (AT)) technique.

The technique was developed to illustrate performability modeling and making design modifications to enhance performability.

The NVP-TB-AT technique was developed by combining the performability advantages of two modified NVP techniques, the NVP-TB (NVP with a tie-breaker) and NVP-AT (NVP with an AT).
N-Version Programming Variants

- Hence, NVP-TB-AT incorporates both a tie-breaker and an AT.
- When the probability of related faults is low, the efficient synchronization provided by the tie-breaker mechanism compensates for the performance reduction caused by the AT.
- The AT is applied only when the second DM reaches a consensus decision.
- When the probability of related faults is high, the additional error detection provided by the AT reduces the likelihood (due to the high execution rate of NVP-TB) of an undetected error.
N-Version Programming Variants

- NVP-TB-AT is a **design diverse, forward recovery technique**.
- The technique uses multiple variants of a program, which run concurrently on different computers.
- The results of the first two variants to finish their execution are gathered and compared.
- If the results match, they are output as the correct result.
- If the results do not match, the technique waits for the third variant to finish.
- When it does, a majority voter-type DM is used on all three results.
- If a majority is found, the matching result must pass the AT before being output as the correct result.
N-Version Programming with Tie-Breaker and Acceptance Test Operation

- The NVP-TB-AT technique consists of an executive, n variants (three variants are used in this discussion) of the program or function, and several DMs: a comparator, a majority voter, and an AT.

```plaintext
run Variant 1, Variant 2, Variant 3
if (Comparator (Fastest Result 1, Fastest Result 2))
    return Result
else Wait (Last Result)
    if (Voter (Fastest Result 1, Fastest Result 2, Last Result))
        if (Acceptance Test (Result))
            return Result
    else error
```
NVP-TB-AT entry

Distribute inputs

Version 1 →

Version 2 →

Version 3 →

Gather results (of two fastest versions, then slowest)

Results from two fastest versions

Comparator

No consensus

Voter

Majority output selected

AT

Result accepted

Success: Consensus output

Success: Accepted output

Exception raised

Result not accepted

Failure exception

NVP-TB-AT exit
N-Version Programming with Tie-Breaker and Acceptance Test Operation

- **AT** Acceptance test;
- **Vi** Variant i;
- **N** The number of versions \( n = 3 \);
- **NVP-TB-AT** NVP with tie-breaker and acceptance test;
- **Ri** Result occurring in the ith order; that is, \( R_1 \) is the fastest, \( R_3 \) is the slowest;
- **R** Result of NVP-TB-AT.
Failure-Free Operation

- Upon entry to the NVP-TB-AT, the executive performs the following: formats calls to the three variants and through those calls distributes the input(s) to the variants.
- Each variant, Vi, executes. No failures occur during their execution.
- The results of the two fastest variant executions (R1 and R2) are gathered by the executive and submitted to the comparator.
- R1 = R2, so the comparator sets R = R1 = R2, as the correct result.
- Control returns to the executive.
- The executive passes the correct result outside the NVP-TB-AT, and the NVP-TB-AT module is exited.
Partial Failure Scenario.

Results Fail Comparator, Pass Voter, Pass Acceptance Test

- Upon entry to the NVP-TB-AT, the executive performs the following: formats calls to the three variants and through those calls distributes the input(s) to the variants.
- Each variant, Vi, executes. No failures occur during their execution.
- The results of the two fastest variant executions (R1 and R2) are gathered by the executive and submitted to the comparator.
- R1 <> R2, so the comparator cannot determine a correct result.
- Control returns to the executive, which waits for the result from the slowest executing variant.
- The slowest executing variant completes execution.
- The result from the slowest variant, R3, is gathered by the executive, and along with R1 and R2, is submitted to the majority voter.
Partial Failure Scenario. Results Fail Comparator, Pass Voter, Pass Acceptance Test.....

- R3 = R2, so the majority voter sets R = R2 = R3 as the correct result.
- Control returns to the executive.
- The executive submits the majority result, R, to the AT.
- The AT determines that R is an acceptable result.
- Control returns to the executive.
- The executive passes the correct result outside the NVP-TB-AT, and the NVP-TB-AT module is exited.
Failure Scenario. Results Fail Comparator, Pass Voter, Fail Acceptance Test

- Upon entry to the NVP-TB-AT, the executive performs the following: formats calls to the three variants and through those calls distributes the input(s) to the variants.
- Each variant, Vi, executes. No failures occur during their execution.
- The results of the two fastest variant executions (R1 and R2) are gathered by the executive and submitted to the comparator.
- R1<> R2, so the comparator cannot determine a correct result.
- Control returns to the executive, which waits for the result from the slowest executing variant.
- The slowest executing variant completes execution.
- The result from the slowest variant, R3, is gathered by the executive, and along with R1 and R2, is submitted to the majority voter.
- R3 = R2, so the majority voter sets R = R2 = R3 as the correct result.
- Control returns to the executive.
- The executive submits the majority result, R, to the AT.
- R fails the AT.
- Control returns to the executive.
- The executive raises an exception and the NVP-TB-AT module is exited.
Failure Scenario.

Results Fail Comparator, Fail Voter

- Upon entry to the NVP-TB-AT, the executive performs the following: formats calls to the three variants and through those calls distributes the input(s) to the variants.
- Each variant, Vi, executes. No failures occur during their execution.
- The results of the two fastest variant executions (R1 and R2) are gathered by the executive and submitted to the comparator.
- R1 <> R2, so the comparator cannot determine a correct result.
- Control returns to the executive, which waits for the result from the slowest executing variant.
- The slowest executing variant completes execution.
- The result from the slowest variant, R3, is gathered by the executive, and along with R1 and R2, is submitted to the majority voter.
- R1<> R2 <> R3, so the majority voter cannot determine a correct result.
- Control returns to the executive.
- The executive raises an exception and the NVP-TB-AT module is exited.
N-Version Programming with Tie-Breaker and Acceptance Test Example

```
(8, 7, 13, -4, 17, 44)

Sum of inputs = 85, distribute inputs

Variant 1: Bubble sort
Variant 2: Quicksort
Variant 3: Original incremental sort

(-4, 7, 8, 13, 17, 44)

Comparator: Result = no match

R_{2j}: -4 7 8 13 17 44
R_{3j}: -4 -7 -8 -13 -17 -44
R_{1j}: -4 7 8 13 17 44
R_{2j}: -4 7 8 13 17 44
R_{3j}: -4 -7 -8 -13 -17 -44

\[ R = (\underbrace{-4, 7, 8, 13, 17, 44}_R) \]

Majority voter: Result = \( R_{1j} \) and \( R_{2j} \) match

AT: Sum of inputs = 85 = sum of outputs

Output: (-4, 7, 8, 13, 17, 44)
```
Architecture

- NVP-TB-AT is a multiprocessor technique with software components residing on \( n = 3 \) hardware units and the executive residing on one of the processors.

- Communications between the software components is done through remote function calls or method invocations.
Question ?