Pontifícia Universidade Católica do Rio Grande do Sul
Faculdade de Informática
Programa de Pós-Graduação em Ciência da Computação

PUCRS

Faculdade de Informática

N-Version Programming – A Fault-Tolerant approach to reliability of Software operation

- Limin Chen & Algirdas Avizienis

Rafael Medaglia – 2008/2

PROGRAMA DE PÓS-GRADUAÇÃO
PPGCC
EM CIÊNCIA DA COMPUTAÇÃO

Agenda

- **Fault-Tolerance SW**
- Concepts of N-Version Programming
- N-Version vs. Recovery blocks
- Implementation
- Feasibility Studies
- MESS Program Experiment
- RATE Program Experiment
- Conclusions

Fault-Tolerance SW

"… executing duplicated copies of a program does not improve the reliability of operation with respect to software defects."

Using redundant software to achieve fault-tolerance constraints:

1 – Complete self-checking is not required;

2 – Does not rely on run time diagnosis;

3 – Same function counts on independently developed alternative routines.

Agenda

- Fault-Tolerance SW
- **Concepts of N-Version Programming**
- N-Version vs. Recovery blocks
- Implementation
- Feasibility Studies
- MESS Program Experiment
- RATE Program Experiment
- Conclusions

Concepts of N-Version Prog. (1/4)

Uses N >= 2 independent implementations of the same program, where each N is called a "version" from the specification.

Who it works?

The program is defined (in a specification language) as completely, unambiguously and trying not to narrow down the possible implementations.

Concepts of N-Version Prog. (2/4)

To allow the use of N-version programming, some points must be define:

1. The function to be implemented must be clearly defined;
2. Data format (Comparison Vectors (C-Vectors) and Comparison Status Indicators (CS-Indicators))* and synchronization mechanisms;
3. The Cross-Check Points (CC-Points)*;
4. The comparison algorithm (that also defines the acceptable discrepancy range for numerical comparisons);
5. The voting;

Concepts of N-Version Prog. (3/4)

The performance criteria must be taken in consideration prior to integrate each version.

A program to manages the N-Versions is required, it is called the drivers. The N-version program is composed by the driver and the versions.

Concepts of N-Version Prog. (4/4)

Mechanisms for the N-Version programming:

1. C-Vectors – Data structure for one version local state;
2. CS-Indicators – Provide instructions after voting/matching C-Vectors. Those instructions can imply in
   1. Continue execution;
   2. Terminate execution in 1+ versions;
   3. Continue after changing values in the C-Vector, based on the majorities' decision.
3. CC-Points – Synchronization points in the N-Versions.

The initial specification must be reliable to the point that two versions cannot present the same defect.

Agenda

- Fault-Tolerance SW
- Concepts of N-Version Programming
- **N-Version vs. Recovery blocks**
- Implementation
- Feasibility Studies
- MESS Program Experiment
- RATE Program Experiment
- Conclusions

N-Version vs. Recovery blocks (1/2)

- *Recovery blocks*
- *Advantage:*
  - Lower cost, once the replaced modules can be used as alternatives.
- *Disadvantages:*
  - Considerable storage overhead (system states prior to achieve a result);
  - Needed precautions to coordinate process within a nested recovery block structure;
  - Some intermediate outputs might not be reversible in real-time environments;
  - Due to the potential flaws listed above, the recovery blocks technique is limited to some types of applications.

N-Version vs. Recovery blocks (2/2)

- *N-Version Programming*
- *Advantages:*
  - No self-checking is required;
  - Some redundancy can be eliminated;
  - It's possible to immediately mask some software faults (no delays);
  - For aerospace application, that count on replication of HW, the use of N-Version Programs may allow the systems to tolerate both HW and SW faults.
- *Disadvantages:* ???

Agenda

- Fault-Tolerance SW
- Concepts of N-Version Programming
- N-Version vs. Recovery blocks
- **Implementation**
- Feasibility Studies
- MESS Program Experiment
- RATE Program Experiment
- Conclusions

Implementation (1/4)

Special Mechanisms

- The three mechanisms mentioned before (C-Vector, CS-Indicators and CC-Points) must be implemented in a way that:
  - The driver must be able to access/manipulate the C-Vector of each version;
  - The CS-Indicators must be followed;
  - The CC-Points are equivalents.

Implementation (2/4)

Inexact Voting

- Dealing with numerical values there are two possible deviations:
  - The “expected” ones, that will happen due to inexact HW representation or data sensitivity of an algorithm;
  - The “unexpected” ones, which are a consequence of inadequate design/implementation.
- As independently of the deviations being expected or not it will cause the versions results to disagree, an inexact voting mechanism must take place when handling non-identical results (Adaptive/Non-Adaptive).

Implementation (3/4)

Adaptive: A weight value is calculated dynamically to each one of the results.

- Favors acceptable results;
- Determining the optimal tolerance parameter is difficult;
- If the results are used as input for the other cycles the accumulation of residual noise can be unacceptable;
- Implemented in software it can be quite slow.

Implementation (4/4)

Non-Adaptive: Uses an allowable discrepancy range and differences of pairs.

- The allowable discrepancy range is hard to calculate;
- Have two possible strategies:
  - Finding the minimal difference between pairs;
  - Or the finding maximum difference between pairs.

The two alternatives, adaptive and non-adaptive voting, have pros and cons and to use it successfully the designer must know:

1. How data sensitive is the algorithm;
2. HW limitations;
3. Allowable range of discrepancies for each instance.

Agenda

- Fault-Tolerance SW
- Concepts of N-Version Programming
- N-Version vs. Recovery blocks
- Implementation
- **Feasibility Studies**
- MESS Program Experiment
- RATE Program Experiment
- Conclusions

Feasibility Studies

Doing some experiments with this technique the followings objectives were kept in mind:

1. Generality and ease of use of N-Version programming;
2. Gain qualitative and quantitative data on effectiveness;
3. Identify problems and difficulties with the technique.

The experiments problems were selected aiming to:

1. Have enough complexity to present residual defects;
2. Be of manageable size, so that the instrumentation is facilitated;
3. Problems should allow convenient generation of multiple versions.

For convenience of the experiments, N is assumed as 3 (versions).

Agenda

- Fault-Tolerance SW
- Concepts of N-Version Programming
- N-Version vs. Recovery blocks
- Implementation
- Feasibility Studies
- **MESS Program Experiment**
- RATE Program Experiment
- Conclusions

MESS Program Experiment

The 3-version MESS Program Experiment

The MESS (Mini-Text Editing System) was an assignment to a 1976 UCLA graduate seminar course. Observations from the results:

1. The methodology is relatively simple and can be generalized to similar applications;
2. The results encourage further investigation on its effectiveness;
3. The 3-Version programming was applied at subroutine level, showing that applying N-Version to certain critical parts of programs can be a valid alternative.

Agenda

- Fault-Tolerance SW
- Concepts of N-Version Programming
- N-Version vs. Recovery blocks
- Implementation
- Feasibility Studies
- MESS Program Experiment
- **RATE Program Experiment**
- Conclusions

RATE Program Experiment (1/3)

The 3-version RATE Program Experiment

The RATE (Region Approximation and Temperature Estimation) also was given as an assignment in the UCLA, but in the 1977. The problem consists of using a pre-determinate equation for the temperature.

All versions where implemented in the same language and ran in the same computer.

RATE Program Experiment (2/3)

Results:

#Cases	#Bad Versions	Acceptable Results	Not Acceptable Results
290	0	290	0
71	1	59	12*
18	2	0	18
5	3	0	5

For more details check the article.

RATE Program Experiment (3/3)

Observations from the results:

1. *Sometimes one of the version developed an error that caused the O.S. system to take over the execution, halting not execution of the 3 versions involved, even only one was causing it;
2. The logic implemented for each version may be correct, incorrect or missing, when it's missing, the combination of one version missing something, one incorrect and one correct might outvote the correct version.

Agenda

- Fault-Tolerance SW
- Concepts of N-Version Programming
- N-Version vs. Recovery blocks
- Implementation
- Feasibility Studies
- MESS Program Experiment
- RATE Program Experiment
- **Conclusions**

Conclusions (1/2)

As mention before some results encourage the utilization of the N-Version programming, among the more relevant there is the fact that it seems applicable to similar applications and using in the subroutines levels is functional. As negative aspects some sort of failures can lead to no/bad results.

Conclusions (2/2)

Some other observations:

1. In real-time environments a system failure may be caused by performance limitations other than functional problems;
2. When there is not an unique path for the solution the N-Version programming may not be applicable;
3. In some cases the sequence of outputs may not follow a specific order, in that case the versions cannot be readily compared;
4. If the allowable range of discrepancy for inexact voting cannot be easily determinate, it's difficult to reach the acceptable results.