Warning: This talk contains actual content. Any relation to the Jerry Springer show is purely coincidental.

An in-your-face look at the N-version programming idea

Software engineering contribution?
- NOT a contribution:
  - Original hypothesis invalidated
  - N-versions were NOT completely independent
  - Difficult to quantify amount of dependency
  - Not a comprehensive review of reliability improvement methods
  - But, Avizienis names replication domains (time, space, program)
- Contribution:
  - Interesting example of scientific method
  - What can we learn about good science for software eng.? Criteria for success and repeatability
  - Economically motivated
  - N cheaper pieces of software vs. one expensive piece
  - Influential idea
  - Especially voting methods

Timeline (y2k compliant…)


- 1985 Knight, Leveson, and St. Jean publish "A Large Scale Experiment in N-Version Programming"
- 1990 Brilliant, Knight, and Leveson publish "Analysis of Faults in an N-Version Software Experiment" Knight and Leveson publish "A Reply to the Criticisms of the Knight and Leveson Experiment"
- 2001 Littlewood, Popov, Strigini publish "Design Diversity: an Update on Research in Reliability Modeling"

The cast:
Avizienis – Hypothesis proponent
Knight and Leveson - Contenders
Littlewood – Most recent work

The hypothesis (Avizienis)
- In hardware, failures are (typically) independent
  - Large reliability gains can be achieved through redundant replications
- In software, replications of the same code manifest the same design defects
  - 100% dependent failures
- Why not create independent versions of software?
  - From a single specification, independently create N versions of the software
  - Key assumption: Independent design assures independent failures
The protagonists (Avizienis/Kelly)

- **Setup**
  - Specification: formal specification language (OBJ), nonformal (PDL), and prose (English)
  - Transaction-oriented airport scheduler
  - Program: 18 working programs (acceptance test) from UCLA CS
  - Grouped to have similar programming ability
  - Correct result: result from 18-version decision algorithm

- **Results**
  - Classification of errors
    - "OK" – result = correct, "cosmetic error", "good" = Ok or cosmetic, "detected error" – abort on invalid input (self-test), "undetected error" – U (different error values) or U* (same error value)
  - Analysis: 3-version voting of all possible triplets
    - All undetected errors were specification, interpretation (eg checking input parameters), or implementation fault
    - Surprise! Independent faults often produced similar errors

The contenders (Knight/Leveson)

- **Setup**
  - Specification: English? (Just one.)
  - Anti-missile - radar object identification
  - Program: 27 versions from 2 different universities
    - More extensive testing, acceptance test as a quality filter
    - Correct result: Gold standard version

- **Results**
  - Classification of errors
    - Failure: any discrepancy between program and gold standard (including fatal exceptions)
  - Analysis: Located and determine cause of failures
    - Some faults cause statistically correlated failures – 5 expected, 93 encountered
    - Logically related faults are not necessarily failure correlated, correlated failures are not necessarily logically related
    - Correlated failures result from mishandling of same point in input space (errors are in the inputs, not the code)

Questions for today

- **Scientific method:** Whom do YOU believe?
  - What can we learn about how to devise good software engineering research methods?

- **Economics:** Possibility and preference
  - Ultra-reliability – Can you get there from here?
  - 10-9 failure rate guaranteed for a single component?
  - Is it better to create n cheap copies or one good one?

- **Related ideas:** Who needs ‘em?
  - How much were they influenced by N-version?
    - Voting, retry and recovery blocks, separate validation program, analytic redundancy
  - Have they increased the number of solvable software engineering problems?

Timeline (where are they now?)

<table>
<thead>
<tr>
<th>Year</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>1985</td>
<td>Knight, Leveson, and St. Jean publish “A Large Scale Experiment in N-Version Programming”</td>
</tr>
<tr>
<td>1990</td>
<td>Brilliant, Knight, and Leveson publish “Analysis of Faults in an N-Version Software Experiment”</td>
</tr>
<tr>
<td>2001</td>
<td>Littlewood, Popov, Strigini publish “Design Diversity: an Update on Research in Reliability Modeling”</td>
</tr>
</tbody>
</table>

Where are they now?

- Avizienis – UCLA – Dependability and Fault tolerance
- Knight – University of Virginia – Information survivability
- Leveson – MIT – Software safety (emphasis on requirements) – Safeware
- Littlewood – Centre for Software Reliability (London) – Software reliability modeling and assessment