COMP-667 Software Fault Tolerance

Assignment 2: N-Version Programming

(20% of Final Grade)

Problem Statement

Write an implementation of the n-version programming (or n-copy programming) software fault tolerance technique seen in class in the programming language of your choice. The idea is that the implementation should be in the form of a reusable "library" (e.g. object / package / template / generic / framework / aspect).

Both distributed and single-process/multi-threaded solutions are acceptable, as well as both preemptive and non-preemptive implementations. Your library should provide at least two common voting strategies, and also allow the user to implement their own voter, if desired. To show that your idea works, write a small sample application that makes use of your library.

Grading

The grade will be based on:

- **Correctness**:
  - Does your infrastructure correctly implement n-version programming (or n-copy programming)?
  - Are there no deadlocks / livelocks?
- **Ease of use**:
  - Does your infrastructure have a simple, yet elegant interface?
  - Can the programmer use your infrastructure without having to do an excessive amount of programming on his own?
  - Are the design decisions reasonable (and documented)?

- **Safety**:
  - Does your infrastructure prevent a programmer from making mistakes when using it?
  - Are programming conventions verified (statically / at run-time)?
  - Is your infrastructure thread-safe?
- **Sample Application**:
  - Is there a clear separation of code between the application and the library?
  - Note: the sophistication of the sample application does not affect the grade.

Hand-In

Please hand in your solution before Friday February 24th! Send an email to Joerg.Kienzle@mcgill.ca and Wisam.Al.Abed@mail.mcgill.ca with the title "COMP-667 Assignment 2 of <your name>" containing:

- Source code
- Instructions that explain how to compile and run the code (if you are using other languages than C, C++, Ada, Java, AspectJ, be prepared to give me a small demo)
- Text explaining your design decisions
  - Justify your interface
    - Explain what makes your library easy to use
    - Explain what makes your library safe to use
  - Justify your implementation decisions
    - Explain why you use the programming language features that you use
- Text explaining the sample application