NOM SIMULATOR TEST PLAN

Sections

[1] page 8

A.1 Introduction

1.1 Scope

This test plan will test the system itself for performance, stress, and reliability (Core Simulator). It will also test each feature and intelligent component. Test cases and their corresponding test logs are provided for each feature and components tested.

[1] page 9

A.2 Test Plan

2.1 Features and components to be tested

The following list describes the features that are tested.

- **2.1.0** Search engine
- **2.1.1** NOML file uploader
- **2.1.2** Molecule editor
- **2.1.3** Molecule validator
- **2.1.4** Chat room
- **2.1.5** File sharing and discussion board
- **2.1.6** Sending email agent
- **2.1.7** Running time prediction finder
- **2.1.8** Similar simulation finder
- **2.1.9** Automatic restarting agent
- **2.1.10** Manual restarting agent
- **2.1.11** Runtime reports
- **2.1.12** Completed reports
- **2.1.13** Load balancing
- **2.1.14** Core simulator

2.2 Features not to be tested

None

[1] pages 10-11

[2] pages 112-3

2.3 Approach
We will be using an Execution Based Testing approach, in this approach the tester takes the product, runs it with known or random input data examining the output. Using this approach we will be focusing in the following four behavioral properties which are fault recovery, utility, reliability, robustness, and performance (Goodenough, 1979)

[1] pages 10-11
[2] pages 112-3

2.4 Fault Recovery
Fault recovery testing verifies that following an error or exception (such as a system crash caused by loss of power or a hardware failure) the application under testing can be restored to a "normal" state, such as the initial state of the application when its executed, and the application continues to perform successfully. This phase may also be applied to verify the successful roll-back and recovery of the data used or manipulated by the application following an error or exception.

[1] pages 10-11
[2] pages 112-3

2.5 Utility. This is the extent to which the user's needs are met when a correct product is used under condition permitted by the specifications. In this phase we will be focusing on how easy and user friendly is the product, and weather the product performs correctly when subjected to inputs which are valid in terms of the specifications.
Note: specify the conditions permitted???

[1] pages 10-11
[2] pages 112-3

2.6 Reliability. This phase is a measure of the frequency and criticality of product failure, which is not supposed to occur if the software is working under permissible operations, conditions, and specifications. It will consist of testing to know how often the system will fail because of data load, not enough database space, hardware constrains etc. After this test has been done we will observe and document the effects of the failure. After both these tests have been performed we will keep track of the time it took to repair.
Note: specify the effects either on interface core simulator etc. exp: when many scientists commit a simulation at the same time the server may get confused and send wrong info. To the wrong person

[1] pages 10-11
[2] pages 112-3

2.7 Robustness. In this phase we are performing stress testing, volume testing, and checking for response time error.

> **2.7.1 Stress testing.** This phase will consist of sending a large volume of data trough the system and see the consequences.
>
> **2.7.2 Error response**. This phase consists of testing the system using invalid data and observe what is the response of the system. Does it return an error message with details of the error or does return just garbage.

[1] pages 10-11
[2] pages 112-3

2.8 Environmental needs
This section contains the hardware, the communications and system software. This specification will also identify any test tools needed if applicable.

A.3 Test Design Specifications

3.1 Feature(s) to be tested. This section will identify the test items and describe the features and combination of features which are going to be object to test.

3.2 Approach refinement. This section will refine the approach described in the test plan. We will include the specific testing techniques to be used and the method of analyzing the test results.

3.3 Test case/procedure identification. In this section we will give a brief description of each test case associated with its particular design and procedure.

[1] page 12

A.4 Test Case Specification

4.1 Test case and feature specification identifier. Test case number, feature id, and name

4.2 Input specifications. This section will specify each input required to execute the test case. Note: explain how each input will be identified either by name, value, file etc.

4.3 Output expectations. This section will specify the outcome of the feature's input execution (time response, valid or invalid etc.)

4.4 Environmental needs (software and hardware). Specify Hardware configurations, size, memory space etc. Software this will specify the software required to execute this test case. It will include operating system, compiler, and test tools if applicable.

[1] page 12

A.5 Test Log

5.1 Test log identifier. Number assigned to feature to identify it within paper.

5.2 Execution description. This section will specify the execution process used for the item tested.

5.3 Activity and event entry. This section will provide a brief description of the events and activities that occurred during the test.

5.4 Procedure results. For each execution we will record he visually observable results. (For example, error messages generated, abort, and requests for operator action). We will also provide if the test was successful or unsuccessful.

[1] page 14

A.6 Test Summary Report

6.1 Test summary report identifier.

6.2 Summary of results. This section will contain a summary of all the testing. It will include any incidents and their resolution.

6.3 Evaluation. This section will provide an overall evaluation of each test feature item including its limitations.

6.4 Approvals. This section will specify

[1] pages 15-16

Bibliography

[1] *Software Engineering Standards.* Institute of Electrical and Electronics Engineers, Inc., (829) pages 9-16, 1984.

[2] Schach R. Stephen. *Software Engineering.* Asken Associates Inc., pages 112,113, 1993.