Test Plan

For the purposes of CASA project we use no UI tests. We have conducted a number of unit test for the sake of development improvement and later on a number of integration test when individual modules are ready to be integrated. However, the integration portion will be limited because of the rigorous and thorough interconnectivity of individual modules (making in part also unit test very difficult). Our mail focus are therefore acceptance tests for when the program if fully translated.

This is mainly because, that we have a working copy of the CASA program in C. The functions and level of accuracy that the outputs and process of CASA (C) entail, are to a standard to witch we would test also. It is for that reason, that our test strategies are more of a comparative nature. Hence our main test strategy.

Overhaul output test

Test strategy

We generate a number of input examples for the original CASA project and run the program using them. For each input case we record the output (resulting pervasive combinations). Our intent is to compare these and see if they match the outputs in our translated version of CASA.

Test approach

For this test we implement a simple program that reads a manifest of the input and output files and runs the CASA comparing these two. It will have to take into account the fact, that due to the implementation of CASA, the resulting pairwise combinations may wary in their ordering. It will therefore compare whether the result sets contain the same elements, not whether they are in the same order. If both results match, we can assume that the software from this standpoint is working.

We will use a manifest to record both the input and output files location and program arguments.

Test Goals

1. Output combination set must contain same elements as the original output in CASA.
2. (There should be no major change in processing speed, but some variation (2x-3x times slower) is still acceptable.

Unit tests

Test strategy

We created a large number of unit test for individual (non trivial) methods in all classes. The purpose of these is to help with development and check weather new additions and changed do not impact the program stability and correctness.

Test approach

We have selected a number of methods we in each class. The goal is a 100% coverage, but trivial methods are omitted. We run the same method in the original case project with few selected inputs and record their outputs. We than try the same input/output combination in the translated Java.

Goals

1. Improvement and stable development.

Coverage in included in project documentation.