SIRM Unit Tests

Contents

[Introduction 3](#_Toc78452070)

[Summary of Tests Conducted 3](#_Toc78452071)

[Procedure of Tests 4](#_Toc78452072)

[Reduce Parent Efficiency 4](#_Toc78452073)

[Efficiency Percentages High vs Low 6](#_Toc78452074)

[Remediation Factor 7](#_Toc78452075)

[Repair Factors 9](#_Toc78452076)

[Stoichiometric Factor 10](#_Toc78452077)

[Backup Percentages 11](#_Toc78452078)

[Outages 12](#_Toc78452079)

[Appendix 13](#_Toc78452080)

# Introduction

The Stochastic Infrastructure Remediation Model (SIRM) tool allows for a series of interconnected infrastructure sectors to be modeled and considers the realistic variability of the impact of a CBRN event (CBRN stands for chemical, biological, radiological, and nuclear). The SIRM tool has two components an ArcGis portion where there is an input of an area of interest and then a Python GUI tool is called, which will model the rate of recovery of the area.

The focus of the tests was on the Python GUI portion, where several key functionalities of the tool where tested, which are Parent Reduction, Efficiencies, Remediation Factors, Stoichiometric Factors, Backup Percentages, and Additional Outages. Each test will be reviewed in detail, and results of the tests will be recorded.

# Summary of Tests Conducted

Table 1 Summary of Tests



In Table 1 there is a summary of the expectations of the tests and results of the tests that where conducted, 7 tests were conducted and all passed, an example of how the code tests looks can be seen in the Appendix. All tests were done comparing the recovery times of the two scenarios. The expected results were summarized in Table 1 the inputs where modified and results can be seen, that of the 7 tests conducted, all 7 where passed.

# Procedure of Tests

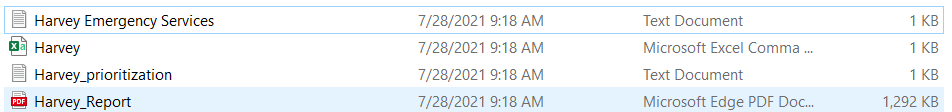
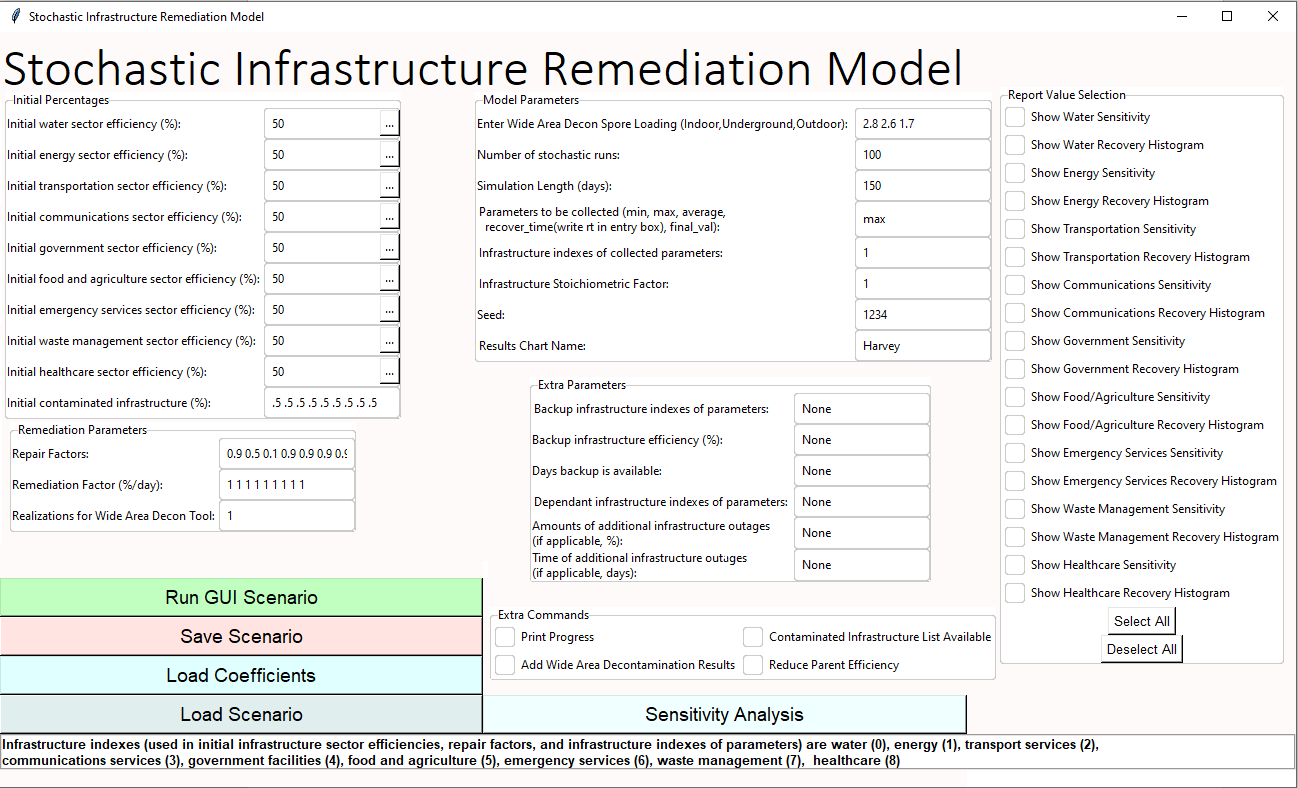
All tests that have been conducted where comparing the recovery days, which were calculated by the python tool. The procedure of the tests was as follows values were chosen as baseline parameters to be compared to an increase or decrease in the value. If a feature was to be tested, arbitrary baseline parameters were chosen, or the feature was toggled on or off. Then, the python GUI tool was run, and a Comma Separated value file or .csv file is outputted and saved for the test to be used in the python script to test the values. An example of the output of the python GUI tool is shown in Figure 1. 

Figure 1 Example of GUI tool output

The Python Script was written with the module unit test to help automate the unit testing. This module will tell the user if the proper result is achieved either asserting a pass or fail. A true or false value was assigned to see if values where larger or smaller, then each other and finally the test results were given. An example of how the tests look will be in the appendix section.

# Reduce Parent Efficiency

The first test conducted was on the Reduce Parent Efficiency functionality. This simulates some sectors using supplies from other sectors that are dependent on each other. An expected result when Reducing Parent Efficiency is that the days of recovery should be higher when reducing Parent Efficiency. This is because when you take supplies from the dependent sectors, it will take longer to recover. A screenshot of the scenario used is shown below.

Figure 2 Example GUI 50% (Parent Reduction off)

The scenario was run with and without the parent efficiency and the days of recovery are shown in the tables below.

Table 2 Parent Reduction On, Recovery Times in Days



Table 3 Parent Reduction Off, Recovery Times in Days



As shown in the charts, changing the Parent Reduction will, in fact, change lengthen the recovery time. The unit test module also returned the expected result of a passed test. Therefore, it can be concluded that the functionality of the parent reduction is working as intended.

# Efficiency Percentages High vs Low

The next test that was run was the initial percentages were changed. The Initial Sector Efficiency represents the operating efficiency of the sector. For this test, it is expected that at higher Efficiencies the days of recovery should be lower, while at lower efficiency percentages they should be higher. The results of the two runs are shown in tables 4 and 5, comparing the recovery times.

Table 4 80% Efficiency Days of Recovery, Recovery Times in Days



Table 5 50% Efficiency Days of Recovery, Recovery Times in Days



Comparing the recovery times in Tables 4 and 5, the test would given our initial hypothesis a passed test would be expected, and this is exactly what happened. The unit test module returned a passed test which was as expected. Therefore, the Efficiency functionality works as intended.

# Remediation Factor

The remediation factor is defined as the linear percentage per day by which contamination is reduced. The results that would be expected are that the lower the remediation factor is, the higher the days of recovery would be this is because less contaminant is reduced per day. Conversely, a higher remediation factor would produce smaller days of recovery. The same 50% efficiency was used as the baseline for both tests and the remediation factor was the only thing that was changed. The remediation factor was lowered to .01%/day for each and then changed to 1%/day.

Table 6 Recovery Times in Days



Table 7 Recovery Times in Days



Comparing the results from Tables 6 and 7, the remediation factor when decreased to .01% per day is shown to increase the days of recovery compared to the Table 7. The Unit Test written had confirmed this and returned a result of pass, thus it can be concluded that the Remediation factor does have an impact works properly.

# Repair Factors

Repair factors are defined at the rate of at which society works to restore the infrastructure services. It is reasonable to speculate that the higher the repair factor the lower the reparation days. The repair factor was run at .1 and then sent to .9 with all the other inputs as figure 2.

Table 8 Higher Repair Factor Recovery Times in Days



Table 9 Lower Repair Factor Recovery Times in Days



Due to the Unit Test module giving a passed test and by inspection it can be concluded that the repair factor functionality is working properly, and no were bugs detected.

# Stoichiometric Factor

The stoichiometric factor is the value by which each stoichiometric coefficient in the reaction equations is multiplied by. This varies the level of stochastic behavior, as well as the stability of the data. With higher stoichiometric factors the days of recovery decrease due to the behavior of the reactions. The tests were conducted with figure 2 inputs while changing the stoichiometric factor from 1 to 400.

Table 10 Stoichiometric Factor of 1, Recovery Times in Days



Table 11 Stoichiometric Factor of 400, Recovery Times in Days



As expected, the stoichiometric factor variation produced the expected results and the unit test module confirmed what figures 10 and 11 show that when varying the stoichiometric factor, the days of recovery decrease. It can be concluded that the stoichiometric factor works as intended and no bug detected for the function.

# Backup Percentages

In the table below there is a description of the backup percentages inputs that need to be entered simultaneously.

Table Description of Backup Percentages



When testing the percentages, the days of recovery for the chosen infrastructures should go down slightly. The decrease should be related to the reported connection that the two infrastructures have. In the test case, the Backup Infrastructure index was 1, (energy), and the dependent infrastructures was 6, (emergency medical services). In the test a decrease was seen in the days of recovery in the emergency medical services indicating that there was no issue with the function and no bug was found.

# Outages

The additional outages are percentages and days of additional outages seen during the event. The percentage is the efficiency of the outage, and the days are how many days are in the outage. When testing the additional outages functionality, the user expects larger days of recovery as compared to no outages.

Table No Additional Outages

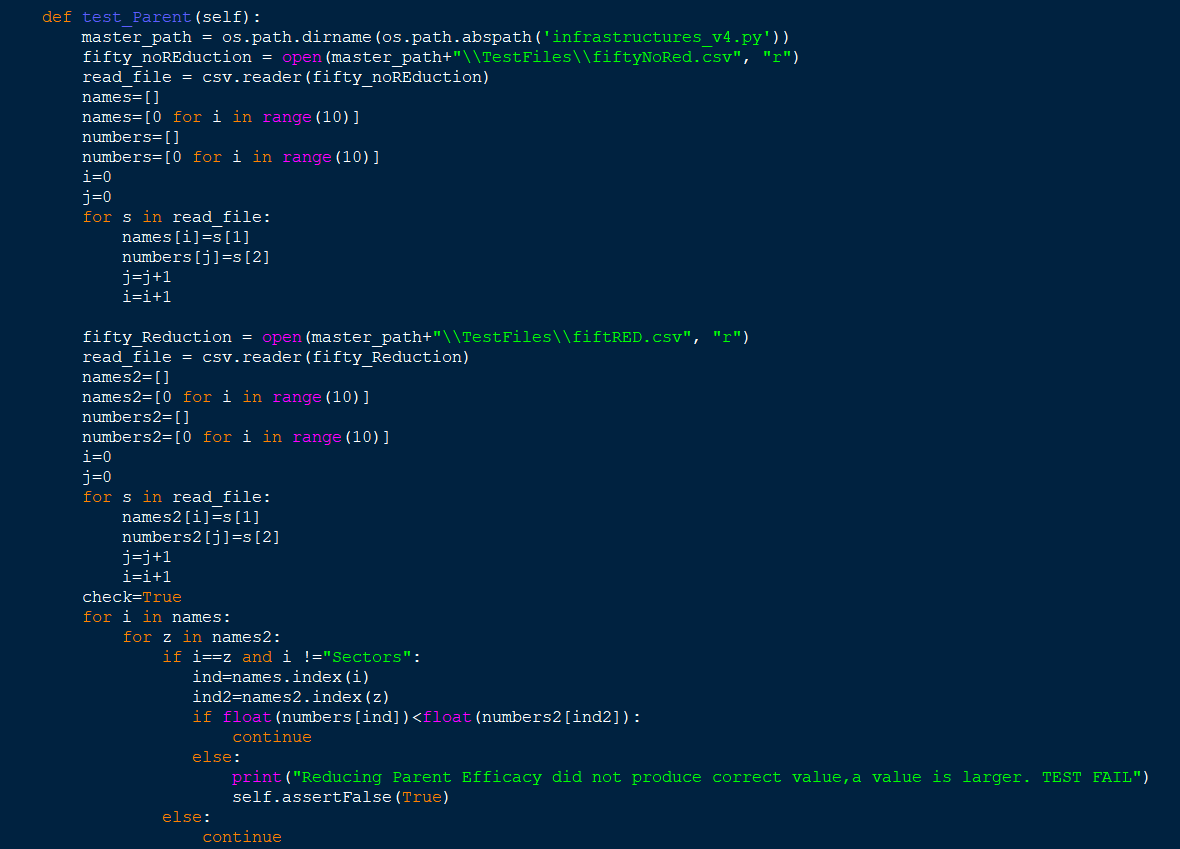


Table Additional Outages



As expected, the Additional Outages variation produced a result and the unit test modal confirmed what Tables 13 and 14 shows. It can be concluded that the additional outages functionality is working properly, and no bug detected.

# Appendix

Figure 3 Code Example