Refactored WFG Algorithm

Test Report (B)

Christopher Jones
Software Quality & Testing

This documentation provides a clear and comprehensive record of all the test cases and faults, as well as a detailed defects report using multiple testing techniques. Presenting an informative critique on the testing approach.

Contents

1 Testing Strategy

1.1 Critique

With a total of 22 functional and structural tests received from the test plan, it became more evident that many of the cases derived were deemed obsolete. Of these cases, most of the structural tests were unable to be executed due to its obscurity and complication to complete the test strongly. It also became clear that acquiring a full understanding of the system internal structures, and meeting the required deadlines proved to be a problematic exercise (*Rongala, 2015*). The execution of a minimal number of structural tests were only possible, therefore the use of this technique became an unnecessary investment in both time and effort.

It was recognised that some of the functional test cases defined nebulous and comparable information, some of which have been modified to give a more concrete approach of each technique. E.g. Addition of duplicate/obsolete test case table and refactoring of the test results table. These informative additions provide a more detailed test/defects record that proves beneficial for developers in future use.

The undertaking of discovering the efficiency of all the tests is a challenging endeavour, as it's not possible to know how many defects there currently are before beginning the test phase. Therefore, the efficiency of the tests will be measured by found defects alone. Since the initial test case, there has been a 77% increase in the number of test cases explored.

After further analysis in *section 1.2.1* and *1.2.2*, it shows Error Guessing (EG) being the most efficient with a defect detection percentage of 71%. EG became reliable with identifying flaws that are of high liability. Exploratory Testing received an impressive defect detection percentage of 94%. 43% of the tests being high severity defects, showing its effectiveness with identifying faults in the system. Finally, EP & BVA became the least efficient technique used with 80% of the defects being of low severity, while finding none of high priority. Considering the total defect percentage and ratio, all techniques have a particularly good reach and was a generous improvement to the functional tests derived in part A.

Defect records were built in conjunction with the test results, providing defect-severity percentages shown in *section 1.2.3*. This table became advantageous with providing thorough recognition of which defects are more critical than others.

In terms of improving the quality of the test, one method I would suggest is deploying automated testing throughout the project. The development team can then contribute with maintaining and executing tests. With the inclusion of an advanced automated process, it could reduce the time spent on manually labour while improving the code coverage (*Altexsoft.com, 2018*).

Word Count: 431 *(Including references)*

1.2 Test Case Review

1.2.1 Output Conditions

Equivalence Partitioning & Boundary Value Analysis

Passed Tests	Partially Passed Tests	Failed Tests
1	10	2

Figure 1: *Percentage and number of fully passed, partially passed and failed test Equivalence Partitioning tests.*

Error Guessing

Passed Tests	Partially Passed Tests	Failed Tests
0	0	7

Figure 2: *Percentage and number of fully passed, partially passed and failed test Error Guessing tests.*

Exploratory Testing

Passed Tests	Partially Passed Tests	Failed Tests
0	8	11

Figure 3: *Percentage and number of fully passed, partially passed and failed test Exploratory tests.*

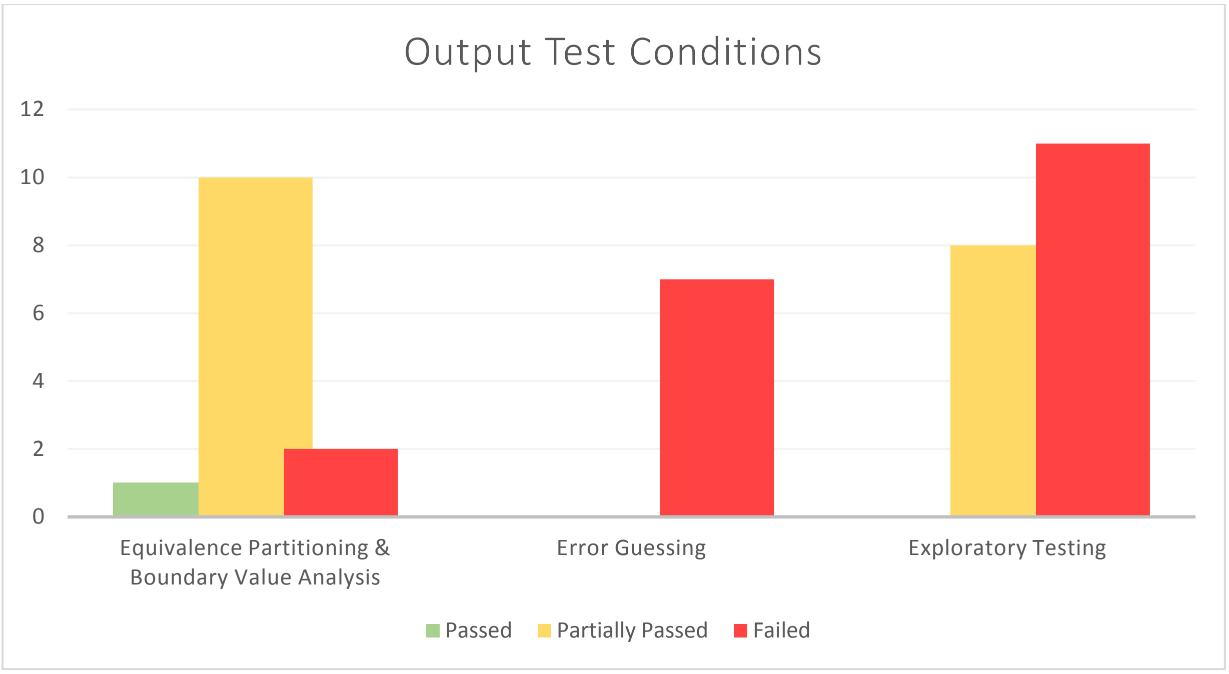

Figure 4: *Bar chart showing the output conditions by testing techniques.*

1.2.2 Test Technique Review

	Technique	**Test Cases**	**Defects Found**	**New Defects per Test**	**Detection of Defects %**	**Detection of New Defect %**
	Equivalence Partitioning	13	5	0.38	92%	38%
	Error Guessing	7	5	0.71	100%	71%
	Exploratory Testing	19	8	0.26	94%	26%
Total	-	**39**	**18**	**0.46**	**95%**	**46%**

Figure 5: *Table containing the percentage of new defects getting detected and the test per defect ratio organised per technique.*

1.2.3 Defect Severity and Priority Review

	Technique	Severity	Defects	Defect-Severity %
	Equivalence Partitioning & Boundary Value Analysis	Low	4	80%
		Medium	0	0%
		High	1	20%
	Error Guessing	Low	1	20%
		Medium	1	20%
		High	3	60%
	Exploratory Testing	Low	4	44%
		Medium	2	22%
		High	3	34%
Total		**Low**	**9**	**47%**
		Medium	**3**	**16%**
		High	**7**	**37%**

Figure 6: *Table containing the defect severity percentage for each testing technique. Quality of life issues are included.*

1.2.4 Duplicate and Obsolete Test Review

Test Case	Test Technique	Rationale for Test Removal	Solution
One of the data values is 1-dimensional	BVA	Removed due to **Test 01** deeming it obsolete. One-dimensional data points have already been tested fully.	Use Test 01 to find output when inputting data value with one 1-dimensional value.
None of the Data/ Reference Points are the same dimensions.	BVA	Removed due to **Test 05** deeming it out-of-date. Creating a test where the data points and reference points have contrasting dimensions has already been analysed.	Use Test 05 to gain similar test reporting outcome.
All the data values are 2-dimensional or more.	BVA	Removed due to multiple tests causing it to be obsolete. Test 02, 03 and 04 have all analysed this tested this situation in detail.	Execute Test 02, 03 or 04

Removing the Front.txt file during while the program is running	EG	Removed due to the complication to execute the test.	Execute tests on large Front.txt fronts and when the Front.txt file is unavailable in separate occasions.
Negative Reference Point and Data Points – 2 Dimensional	BVA	Test case has become obsolete due to other tests analysing similar credentials	Execute test cases 12 and 21 to gather similar test results.
Negative Reference Point and Data Points – 3 Dimensional	BVA	Test case has become obsolete due to other tests analysing similar credentials	Execute test cases 13 and 21 to gather similar test results
EP Class ID: 5 Fronts separated correctly with hashes	EG	Tests 16 and 17 cover the syntax of the Front.txt file, therefore deeming this test obsolete.	Executing Test 16 and 17
Incorrect Front.txt syntax. – Empty line in the Front.txt file.	EG	Removed due to Test 14 and 15 which analyse the outcome when whitespaces are contained in the Front.txt file.	Execute Test 14 and 15

Figure 7: *Table containing the list of test cases that were removed due to becoming obsolete or creating a duplicate test input.*

1.2.5 Assumptions

NO.	**DESCRIPTION**	**ASSUMPTION FUNDAMENTALS**	**ACTUAL ASSUMPTION**
01	Front.txt **syntax**	Specification provided indicates that each dimension is **separated by a new line**. Information is not available while running the program or in the source code provided.	Assume that the **specification is correct**.

02	***cygwin1.dll*** file	**Specification doesn't describe** that the *cygwin1.dll* file is a requirement for the program to work.	Assume that the **program requires the file** for functional testing to work as expected
03	Input **syntax**	Specification doesn't describe the inputs that the user needs to make for the program to work.	Specification is **vague** and assume that the program needs to input the Front.txt file and each dimension in the Command Prompt.
04	**Two-Dimensional** Outcomes	Specification provided indicates that **only two-dimensional outcomes** are expected to be verified.	Assume that the program shouldn't work for **one-dimensional** outcomes.
05	Refactored program **name change**	Specification provided doesn't indicate that the name of the application (*wfg*) has been changed (*wfgRefactoredSQT*).	For each test, the application has been **renamed** from "*wfgRefactoredSQT*" back to "*wfg*"
06	Text file name	Requirements brief doesn't indicate what the name of the text file should to run the program.	The text file that is being used to test the program will be called "Front.txt"
07	Output showing each Front	It is not specified whether having the program output the fronts is a required task.	Not include the front outputs in expected and actual outcomes throughout each test

Figure 8: *Table showing the assumptions that were made during each test.*

1.2.6 Legends

Defect Outcome	Definition
P	**Passed Test** – Test produced the correct output and containing no known defects.
PP	**Partially Passed –** Tests that produce the correct Hypervolume output while containing minor defects that are quality concerns. *E.g. Misspelt error messages*
F	**Failed Test** – Test fault occurred which caused an incorrect output to be incorrect. *E.g. Output Hypervolume was expected to be 5.0000000000 but the Actual Hypervolume ended up being 25.000000000*

Figure 9: *Small legend describing the potential defect outcomes for each test case.*

Severity/Defect Scale	Definition
High **(H)**	Affects the system in a large scale. Defect causing the user to get insufficient results or information explaining the error. Top priority.
Medium **(M)**	Defect that affects the result for users but not a frequent defect to discover.
Low **(L)**	Does not affect the results of the system. Very unlikely for typical users to find the defect. Not a huge priority.

Figure 10: *Legend describing the severity and defect scale with descriptions.*

2 Test Case Records

2.1 Equivalence Partitioning and Boundary Value Analysis Test Results

No.	Test Case	Justification	Input	Expected Output	Actual Output	Condition	Comments
01	Testing **1-Dimensional** Hypervolume Partition	**As stated in the Assignment Brief –** *"...only two-dimensional outcomes are expected to be verified."* Analysing this partition to see if an outcome appears for one-dimensional	Front.txt file: *#* *1* *2* *3* *#* Input command – *"wfg Front.txt 5"*	Output contains: Error Message notifying the user that there needs to be >= 2 Reference Points.	Output contains: Error Message *"There should be more than one dimension in the reference point*".	**P**	Detailed error message
02	Testing **2-Dimensional** Hypervolume Partition	**As stated in the Assignment Brief –** *"...only two-dimensional outcomes are expected to be verified."* **–** Analysing this partition will test whether the output is affected when inputting 2D data.	Front.txt file: *#* *1 1* *2 2* *3 2* *#* Input command – *"wfg Front.txt 5 5"*	Output contains: Hypervolume: *16.000000000* Front time: 0.00000 (s) Total time: 0.00000 (s)	Output contains: Hypervolume: *16.000000000* Contains message *"Why bother????"* Total time: 0.00000 (s)	**PP**	Displaying unnecessary message instead of the front time.

03	Testing **3-Dimensional** Hypervolume Partition __________ EP Class ID: 3 All Data Points **<** Reference Point	**As stated in the Assignment Brief –** *“The functionality of more than two dimensions can be tested.”* **–** This test will verify that the test is following the credentials described in the brief. **–** Will also test the boundaries of all the data points and reference point.	Front.txt file: *#* *1 1 1* *2 2 2* *3 2 2* *#* Input command – *“wfg Front.txt 5 5 5”*	Output contains: Hypervolume: *64.000000000* Display front time: 0.00000 (s) Displays total time: 0.00000 (s)	Output contains: Hypervolume: *64.000000000* Contains message – *“Why bother????”* Contains message *“IF the user sees this, they should check themselves out!!!”* Displays total time: 0.00000 (s)	**PP**	Displaying the correct Hypervolume twice.
04	Testing **4-Dimensional** Hypervolume Partition	**As stated in the Assignment Brief –** *“The functionality of more than two dimensions can be tested.”* Testing if the program meets the specifications.	Front.txt file: *#* *1 1 1 1* *2 2 2 2* *3 2 2 3* *#* Input command – *“wfg Front.txt 5 5 5 5”*	Output contains: Hypervolume: 256.00000000 Display front time: 0.00000 (s) Displays total time: 0.00000 (s)	Output contains: Hypervolume: 256.00000000 Contains message *“Why bother????”* Displays total time: 0.00000 (s)	**PP**	In terms of the Hypervolume output, functioning ordinarily.

05	EP Class ID: 4 All Data Points **>** Reference Point	**Stated in the Assignment Brief** – *"…calculates the area created over data points relative to the reference point."* Confirming if the statement is still true and whether the boundaries between each point affect the output.	Front.txt file: *#* *2 2* *2 3* *3 4* *#* Input command – *"wfg Front.txt 1 1"*	Output contains: Hypervolume: *6.0000000000* Front time: 0.00000 (s) Total time: 0.00000 (s)	Output contains: Hypervolume: *6.0000000000* Contains message *"Why bother????"* Total time: 0.00000 (s)	**PP**	
06	EP Class ID: 4 All Data Points **=** Reference Point	**As stated in the Assignment Brief –** *"…calculates the area created over data points relative to the reference point."* **–** Making sure that the area is calculated regardless of what the data values and reference point is.	Front.txt file: *#* *2 2* *2 2* *2 2* *#* Input command – *"wfg Front.txt 2 2"*	Output contains: Hypervolume: 0.00000000 Front time: 0.00000 (s) Total time: 0.00000 (s)	Output contains: Hypervolume: 0.*0000000000* Contains message *"Why bother????"* Total time: 0.00000 (s)	**PP**	

07	EP Class ID: 4 Data Points that are **above and below** the Reference Point	Data Values shouldn't dominate the Reference Point.	Front.txt file: *#* *5 5* *2 2* *3 4* *#* Input command– *"wfg Front.txt 3 3"*	Output contains: Hypervolume: 4.0000000000 Front time: 0.00000 (s) Total time: *0.00000 (s)*	Output contains: Hypervolume: *4.0000000000* Contains message *"Why bother????"* Total time: *0.00000 (s)*	**PP**	
08	Program handling **two fronts** in a single process	Validate that the program still functions when inputting two fronts in one process.	Front.txt file: *#* *5 5* *2 2* *3 4* *#* *2 3* *1 2* *4 4* *#* Input command– *"wfg Front.txt 5 5"*	Output contains: Hypervolume: 9.0000000000 12.000000000 Front time: 0.00000 (s) Total time: 0.00000 (s)	Output contains: Hypervolume: 9.0000000000 12.000000000 Contains message *"Two fronts here we come:"* Contains message *"Why bother????"*	**PP**	Unique message which affects software quality.
09	Program handling **three fronts** in a single process	Validate that the program still functions when inputting three fronts in one process.	Front.txt file: *#* *5 5* *2 2* *3 4* *#* *2 3* *1 2* *4 4* *#* *1 1* *2 2* *3 3* *#* Input command– *"wfg Front.txt 5 5"*	Output contains: Hypervolume: 9.0000000000 12.000000000 16.000000000 Front time: 0.00000 (s) Total time: 0.00000 (s)	Output contains: Hypervolume: 9.0000000000 12.000000000 16.000000000 Contains message *"Clearly a greedy user!"* Contains message *"Why bother????"* Total time: 0.00000 (s)	**PP**	Unique message which affects software quality.

10	Program handling **four fronts** in a single process	Validate that the program still functions when inputting four fronts in one process.	Front.txt file: *#* *5 5* *2 2* *3 4* *#* *2 3* *1 2* *4 4* *#* *1 1* *2 2* *3 3* *#* *4 5* *3 4* *6 5* *#* Input command– *"wfg Front.txt 8 8"*	Output contains: Hypervolume: *36.0000000000* *42.000000000* *49.000000000* *20.000000000* Front time: *0.00000 (s)* Total time: *0.00000 (s)*	Output contains: Hypervolume: *36.0000000000* *42.000000000* *49.000000000* *20.000000000* Contains message *"Clearly a greedy user!"* Contains message *"Why bother????"* Total time: *0.00000 (s)*	**PP**	Same output as the Test 09.
11	Reference Point is zero	Validating the boundary between a positive and negative reference point.	Front.txt file: *#* *5 5* *2 2* *3 4* *#* Input command– *"wfg Front.txt 0 0"*	Output contains: Hypervolume: *25.0000000000* Front time: *0.00000 (s)* Total time: *0.00000 (s)*	Output contains: Hypervolume: *%\%* Contains message *"Why bother????"*	**F**	Invalid Hypervolume output

12	Negative Reference Point – 2 Dimensions	Validating the boundary between a positive and negative 2D reference points.	Front.txt file: *#* *5 5* *2 2* *3 4* *#* Input command– *"wfg Front.txt -1 -1"*	Output contains: Hypervolume: *36.000000000* Front time: *0.00000 (s)* Total time: *0.00000 (s)*	Output contains: Hypervolume: %\%\ Contains message *"Why bother????"*	**F**	
13	Negative Reference Point – 3 Dimensions	Validating the boundary between a positive and negative 3D reference points.	Front.txt file: *#* *5 5 5* *2 2 2* *3 4 3* *#* Input command– *"wfg Front.txt -1 -1 -1"*	Output contains: Hypervolume: *216.000000000* Front time: *0.00000 (s)* Total time: *0.00000 (s)*	Output contains: Hypervolume: *216.000000000* *216.000000000* Contains message *"IF the user sees that, they should check themselves out!!!"* Contains message *"Why bother????"* Total time: *0.00000 (s)*	**PP**	Duplicate Hypervolume output Unique message which affects software quality.

2.2 Error Guessing Test Results

No.	Test Case	Justification	Input	Expected Output	Actual Output	Condition	Comments
14	A **space after a hash** in the Front.txt file	Test to see if spaces affect the Front.txt file in a **negative** way. Error message should appear based on experience.	Front.txt file: *#[SPACE]* *1 1* *2 2* *#* Input command *"wfg Front.txt 5 5"*	Output contains: **Error Message** *"Whitespacing in Front.txt. Edit file and try again".* Unsuccessful Hypervolume	Output contains: No Output or Error Message.	**F**	Program not functioning
15	A **space before the hash** in the Front.txt file	Test to see if spaces affect the Front.txt file in a **negative way**. Error message should appear based on experience.	Front.txt file: *[SPACE]#* *1 1* *2 2* *#* Input command *"wfg Front.txt 5 5"*	Output contains: **Error Message** *"Whitespacing in Front.txt. Edit file and try again".* Runs the Hypervolume unsuccessfully	Output contains: No Output or Error Message Displayed.	**F**	
16	All data in the Front.txt file to **appear in the first line** of the text file	Test to see if the program is meeting the requirements provided. Each data point should be in a separate line.	Front.txt file: *#1 1 2 2#* Input command *"wfg Front.txt 5 5"*	Output contains: **Error Message** *"Invalid Data Point Inputs modify Front.txt and try again".* Unsuccessful Hypervolume	Output contains: No Output or Error Message Displayed.	**F**	

17	Non-numerical symbol (£, $, %, &, *, ~) instead of the hash symbols	Checking if the requirements are met as "#" should be in between each front. Alternative non-numerical symbols can affect the syntax of a program.	Front.txt file: *%* *1 1* *2 2* *%* Input command *"wfg Front.txt 5 5"*	Output contains: **Error Message** *"Invalid Front(s). Modify Front.txt and try again".* Runs the Hypervolume unsuccessfully	Output contains: No Output or Error Message Displayed.	**F**	
18	**No Reference Point** is provided	Testing to see the outcome when the user forgets to input a reference point.	Front.txt file: *#* *1 2* *2 3* *#* Input command *"wfg Front.txt"*	Output contains: **Contains Message** *"No reference point provided: using the origin"* Hypervolume Output: *6.0000000000* Front time: *0.00000 (s)* Total time: *0.00000 (s)*	Output contains: **Error Message** appears *"No reference point provided. Please provide a reference point".*	**F**	Expectation was for the program to use the origin.
19	Program executing a large front	Testing how the program handles large data processes	Front.txt file: *#* *1 2* *2 3* *2 2* *3 2* *3 4* *5 5* *4 5* *6 6* *5 6* *8 8* *6 7* *(x20)* *#*	Output contains: Hypervolume Output: *114.000000000* Front time: *0.00000 (s)* Total time: *0.00000 (s)*	Output contains: Displays Hypervolume: *57.000000000* Contains message: *"Why bother????"* Total time: *0.00000 (s)*	**F**	Incorrect Hypervolume

			Input command *“wfg Front.txt 15 15“*				
20	Program executing five or more fronts	Testing how the program handles many fronts simultaneously	Front.txt file: *#* *1 2* *2 3* *#* *2 2* *3 2* *#* *3 4* *5 5* *#* *4 5* *6 6* *#* *5 6* *8 8* *#* *9 7* *6 7* *#* Input command *“wfg Front.txt 15 15“*	Output contains: **Displays all Hypervolume**: *182.000000000* *169.000000000* *132.000000000* *110.000000000* *90.0000000000* *72.0000000000* Front time: *0.00000 (s)* Total time: *0.00000 (s)*	Output contains: **Displays all Hypervolume**: *91.000000000* *84.000000000* *66.000000000* *55.000000000* *45.000000000* *36.000000000* Contains message *“Clearly a greedy user!”* and *“Why bother????”* Total time: *0.00000 (s)*	**F**	Unique message which affects software quality. Incorrect Hypervolume

2.3 Exploratory Test Results

No.	Test Case	Justification	Input	Expected Output	Actual Output	Condition	Comments
21	Negative Data Values in the Front.txt file.	Testing the positive and negative boundary for all data values in the Front file.	Front.txt file: *#* *-1 -2* *-2 -3* *-1 -4* *#* Input command: *“wfg Front.txt 5 5”*	Output contains: Hypervolume: *62.000000000* Total time: *0.00000 (s)*	Output contains: Hypervolume: *62.000000000* Contains message: *“Why bother????”* Total time: *0.00000 (s)*	PP	Front time replaced with unique message to user affecting software quality
22	Hypervolume with only one Data Point	Testing how the program functions when only one data point is provided. Should provide valid Hypervolume	Front.txt file: *#* *2 3* *#* Input command: *“wfg Front.txt 5 5”*	Output contains: Display Hypervolume: *6.0000000000* Total time: *0.00000(s)*	Output contains: Displays Hypervolume: *6.000000000* Contains message: *“Why bother????”* Total time: *0.00000 (s)*	PP	
23	Reference Points that contain 0 values.	Checking how the program deals with zero values in the reference point	Front.txt file: *#* *2 0* *3 0* *4 0* *#* Input command: *“wfg Front.txt 5 5”*	Output contains: Display Hypervolume: *15.000000000* Total time: *0.00000(s)*	Output contains: Displays Hypervolume: 15.000000000 Contains message: *“Why bother????”* Total time: *0.00000(s)*	PP	

24	No Front.txt or Reference Point	Monitoring the process of running the program when the user hasn't provided with the Front.txt file and the reference point.	Front.txt file: *#* *1 2* *2 3* *3 4* *#* Input command: *"wfg"*	Output contains: **Error Message** *"File could not be opened. Please try again".* **Fails** to run the Hypervolume.	Output contains: *"1209381294124 1kjhkjdhksj hkfsjddhf238382 3431413213123 01228777998hfdh gsdhgfshgf987 9799797lfj gljdslfkgkdjfgw9rjg 4jt4392ithjbv49425 y846888vjr8v8b8w r8t888%#*"*	F	Output never changes when repeated
25	Removal of *cygwin1.dll* file.	cygwin1.dll file is **necessary** for the program to work. Testing program without that file.	Front.txt file: # 1 2 2 3 3 4 # Input command: "wfg Front.txt 5 5"	Output contains: **Error Message** *"The code execution cannot proceed because cygwin1.dll was not found."* **Fails** to run the program.	Output contains: **Error Message** *"File 5 could not be opened".*	F	Error message needs to be more specific
26	**No Data Points**	Still contains the **Front.txt** file and "**#**". How the program reacts when no data is inside the front file.	Front.txt file: *#* *#* Input command *"wfg Front.txt 5 5"*	Output contains: **Error Message** *"No Data Points in Front.txt. Modify Front.txt and try again".* Runs the Hypervolume **unsuccessfully**	Output contains: **Error Message** *"Your Reference Point should have 13107387 values".*	F	

27	EP Class ID: 8 **Empty Front in Front.txt**	Still contains the Front.txt file, but not the "#". **Empty .txt file.**	Empty Front.txt file Input command *"wfg Front.txt 5 5"*	**Output contains:** **Error Message** *"No Data Points in Front.txt. Modify Front.txt and try again".* Runs the Hypervolume **unsuccessfully**	**Output contains:** **Error Message** "Your reference pint should have 0 values".	**F**	Quality of life error. Incorrect error message
28	EP Class ID: 9 **No Front.txt file** in the folder.	How the program reacts when the Front.txt file is missing in the folder.	*No Front.txt file* Input command *"wfg Front.txt 5 5"*	Output contains: **Shows Error Message** – *"Front.txt file cannot be found."* Notifying the lack of Front.txt file. Runs the Hypervolume **unsuccessfully**	Output contains: Outputs Error Message – "File Front.txt could note be opened"	**PP**	Misspelt error message
29	EP Class ID: 10 **Invalid Data Points Inputs**	Test for **invalid characters** entered in the commands.	Front.txt file: *#* *1 2* *2 B* *#* Input command *"wfg Front.txt 5 5"*	Output contains: **Shows Error Message** – *"Invalid Data Point Inputs modify Front.txt and try again".* Runs the Hypervolume **unsuccessfully**	Output contains: Displays Hypervolume: 18.000000000 Contains message: *"Why bother????"* Total time: *0.00000 (s)*	**F**	

30	All Data Points are decimals	Testing for all data points that may contain decimal values.	Front.txt file: # 0.5 0.6 0.4 0.3 0.6 0.2 # Input command "wfg Front.txt 2 2"	Output contains: Hypervolume: 2.8600000000 Time: 0.00000 (s) Total time: *0.00000 (s)*	Output contains: Hypervolume: 2.860000000000 Contains message: *"Why bother????"* Total time: *0.00000 (s)*	**PP**	
31	Decimal Reference Point	Test for reference points that may contain decimal values	Front.txt file: # 4 3 5 2 3 1 # Input command "wfg Front.txt 5.5 5.5"	Output contains: Hypervolume: 11.250000000 Time: 0.00000 (s) Total time: *0.00000 (s)*	Output contains: Displays Hypervolume: 11.25000000000 Contains message: *"Why bother????"* Total time: *0.00000 (s)*	**PP**	
32	Valid and Invalid Data Points in a Front	Test the Front.txt file when the first front is not a valid input, but the second front is.	Front.txt file: # # 3 4 2 2 # Input command "wfg Front.txt 5 5"	Output contains: Error Message that notifies the user the problem.	Output contains: Contains message *"Two fronts here we come:"* No Output or Error Message Displayed.	**F**	Program freezes command line and never continues
33	Inputting Front.txt incorrectly in the command	Checking how the output is modified when the user inputs the Front.txt incorrectly in the command prompt.	Front.txt file: # 1 2 2 3 4 2 # Input command "wfg Front 5 5"	Output contains: Hypervolume: 12.0000000000 Time: *0.00000 (s)* Total time: *0.00000 (s)*	Output contains: Hypervolume: 12.0000000000 Contains message: *"Why bother????"* Total time: *0.00000 (s)*	**PP**	Minor quality concerns

34	Inputting three empty fronts with 2-dimensional reference point	Similar scenario to Test 27 but concentrating on multiple fronts.	Front.txt file: # # # # Input command "wfg Front.txt 5 5"	Output contains: Error Message: *"Your reference should have 0 values"*	Output contains: Contains message *"Clearly a greedy user!"* No Output or Error Message Displayed.	**F**	Infinite loop error
35	Inputting three empty fronts with no reference point	Granted that in Test 34, the expected outcome is the program asking the user for no reference point. This test will edit the input to meet that demand.	Front.txt file: # # # # Input command "wfg Front.txt"	Output contains: Error Message that helps the user identify the problem	Output contains: Contains message *"Clearly a greedy user!"* No Output or Error Message Displayed.	**F**	Infinite loop error
36	Inputting two empty Fronts all on the same line	Checking how much the structure of the Front.txt affects the result of the outcome	Front.txt file: #### Input command "wfg Front.txt 5 5"	Output contains: Error Message that helps the user identify the issue.	Output contains: No output or error message	**F**	
37	Front with no hashes between the data points	Testing the Front file that contains data values without the hash separating each front.	Front.txt file: 2 3 3 2 Input command "wfg Front.txt 5 5"	Output contains: Error Message that helps the user identify the issue.	Output contains: No output or error message	**F**	

38	Data value is 3-dimensional, but Reference Point is 2-dimensional	Testing how the program functions for 3-dimensional data values when the reference point is 2-dimensional	Front.txt file: # 2 3 4 3 2 2 4 3 3 # Input command "wfg Front.txt 5 5"	Output contains: Error Message: *"Your reference should have 3 values"*	Output contains: Error Message *"Your reference point should have -2 values"*	**F**	Providing user incorrect information
39	Data value is 4-dimensional, but Reference Point is 2-dimensional	Testing to see how it compares to Test 36's outcome.	Front.txt file: # 2 3 3 2 4 3 # Input command "wfg Front.txt 5 5 5 5"	Output contains: Contains message: *"Your reference point should have 2 values"*	Output contains: Contains message: *"Your reference pint should have 2 values"* Total time: *0.00000 (s)*	**PP**	
40	Containing an empty front and no reference point	Testing to see if an error message appears for both not having any data points or a reference point.	Front.txt file: # # Input command "wfg Front.txt	Output contains: Error Message that helps the user identify the issue.	Output contains: Error Message *"No referrence point provided. Please provide a reference point"*	**PP**	Misspelt error message

3 Defect Records

A record of the defects that occurred in each of the tests is displayed with additional description and solutions to solve the defects. This report includes quality of life (QOL) defects that affect the user experience as oppose to the performance of the software outputs.

Initially discovered by test 02

Defect No	01	**Operating System**	Microsoft Windows 10 Pro x64
Defect Type	QOL	**Testing Tool**	Command Prompt
Tests ID affected by Defect	02, 03, 05, 06, 07, 08, 09, 10, 11, 12, 13, 19, 20, 21, 22, 23, 29, 30, 31, 33, 39	**Severity**	Low
Program Title	wfg.exe	**Priority**	Medium
Tester	Christopher Jones		

Defect Description
When the program has successfully executed a Hypervolume, it then produces an unexpected message to the user.

Expected Output	**Actual Output**
Hypervolume output received with no external messages towards the user.	Hypervolume produced + receiving unnecessary message. *"Why bother????"*

Reproducing Steps

1. Edit Front.txt file, making sure it is a valid syntax, containing one front
2. Write the input commands in Command Prompt, where the data points and reference point have the same dimensions. Links with the Front.txt file. E.g. *"wfg Frontr.txt 5 5"*
3. Run the program
4. Program outputs error: *"Why bother????"*

Severity and Priority Reasoning

Severity:
Low as the defect mentioned contains minimal effect to the output of each Hypervolume. Message only effects the QOL of the system.

Priority:
Medium as the defect is visible to all users when outputting successful Hypervolumes.
Decreases the user experience.

Additional Testing

N/A

Initially discovered by test 03

Defect No.	02	**Operating System**	Microsoft Windows 10 Pro x64
Defect Type	QOL	**Testing Tool**	Command Prompt
Tests ID affected by Defect	03, 13	**Severity**	Low
Program Title	wfg.exe	**Priority**	Medium
Tester	Christopher Jones		

Defect Description

When the program has successfully executed a three-dimensional hypervolume, it outputs the hypervolume twice.

Expected Output	**Actual Output**
System outputting the correct hypervolume only **once**.	System outputs the correct hypervolume **twice**

Reproducing Steps

1. Editing the Front.txt file so each data point has three-dimensional only.
2. Write the input commands in Command Prompt, where the reference point has three-dimensional. Also links correctly with the Front.txt file. E.g. *"wfg Frontr.txt 5 5 5"*
3. Run the program
4. Program outputs error

Severity and Priority Reasoning

Severity:
Low as it causes minimal effect to software performance. Doesn't affect Hypervolume result at all. Message only effects the QOL of the system.

Priority:
Medium as the defect is easily detected by users when attempting to get three-dimensional hypervolumes, but not visible in all outputs with different dimensions.

Additional Testing

Test 13: Testing three-dimensional hypervolumes with different data points and reference points (Negative numbers).

Initially discovered by test 08

Defect No.	03	**Operating System**	Microsoft Windows 10 Pro x64
Defect Type	QOL	**Testing Tool**	Command Prompt
Tests ID affected by Defect	08, 32	**Severity**	Low
Program Title	wfg.exe	**Priority**	Medium
Tester	Christopher Jones		

Defect Description

When the system has successfully executed **two** hypervolumes in the same process (two fronts), it outputs a message that's targeted towards the user.

Expected Output	**Actual Output**
System outputting two hypervolumes, with no external messaging towards users.	System outputs hypervolume, as well as adding message *"Two fronts here we come:"* to the users.

Reproducing Steps

1. Editing the Front.txt file so each data point has two fronts only.
2. Write input commands in Command Prompt, where the reference point has the same dimensions as all the data points. *E.g. Reference point and data point all being two-dimensional values.* Also links correctly with the Front.txt file.
3. Run the program
4. Program outputs error

Severity and Priority Reasoning

Severity:
Minimal effect to software performance. Doesn't affect Hypervolume result at all. Message only effects the QOL of the system.

Priority:
Clear visible detection for users when attempting 2 fronts. Likelihood for users to input multiple fronts in the system is much smaller compared to producing one 2-dimensional front.

Additional Testing

Test 09 and 10: Testing multiple fronts different data points and reference points.

Initially discovered by test 09

Defect No.	04	**Operating System**	Microsoft Windows 10 Pro x64
Defect Type	QOL	**Testing Tool**	Command Prompt
Tests ID affected by Defect	09, 10, 20, 34, 35	**Severity**	Low
Program Title	wfg.exe	**Priority**	Medium
Tester	Christopher Jones		

Defect Description

When the system has successfully executed **three** hypervolumes in the same process (three fronts), it outputs a message that's targeted towards the user.

Expected Output	**Actual Output**
System outputting three hypervolumes, with no external messaging towards users.	System outputs hypervolume, as well as adding message *"Clearly a greedy user!"* to the users.

Reproducing Steps

1. Editing the Front.txt file so each data point has three fronts only.
2. Write input commands in Command Prompt, where the reference point has the same dimensions as all the data points. *E.g. Reference point and data point all being three-dimensional values.* Also links correctly with the Front.txt file.
3. Run the program
4. Program outputs error

Severity and Priority Reasoning

Severity:
Minimal effect to software performance. Doesn't affect Hypervolume result at all. Message only effects the QOL of the system.

Priority:
Clear visible detection for users when attempting three fronts. Likelihood for users to input multiple fronts in the system is much smaller compared to producing one two-dimensional front.

Additional Testing

N/A

Initially discovered by test 11

Defect No.	05	**Operating System**	Microsoft Windows 10 Pro x64
Defect Type	Functionality	**Testing Tool**	Command Prompt
Tests ID affected by Defect	11, 12	**Severity**	High
Program Title	wfg.exe	**Priority**	High
Tester	Christopher Jones		

Defect Description

Invalid Hypervolume when the Reference Points are zero.

Expected Output	**Actual Output**
System outputs the correct Hypervolume to the user	System outputs invalid Hypervolume: %\%\ to the user

Reproducing Steps

1. Editing the Front.txt file so each data point is a valid data point, all the same dimensions.
2. Write input commands in Command Prompt, where the reference point values are all less than or equal to 0, whilst having the same dimensions as all the data points. Also links correctly with the Front.txt file.
3. Run the program
4. Program outputs invalid Hypervolume: %\%\

Severity and Priority Reasoning

Severity:
High as it provides an incorrect error message, affecting the system performance.

Priority:
Defect has a drastic effect to UX. Provides incorrect value to the user when inputting common Reference Point values. Giving user false messaging that may confuse them. Will affect the userbase long term.

Additional Testing

Test 12: Testing negative reference point values. If the defect is still occurring, then it proves to be only appearing when the reference point is less than or equal to zero.

Initially discovered by test 03

Defect No.	06	**Operating System**	Microsoft Windows 10 Pro x64
Defect Type	QOL	**Testing Tool**	Command Prompt
Tests ID affected by Defect	03, 13	**Severity**	Low
Program Title	wfg.exe	**Priority**	Medium
Tester	Christopher Jones		

Defect Description

When the program has successfully executed a 3-Dimensional Hypervolume,
it then produces a message targeted towards the user.

Expected Output	**Actual Output**
System outputting three-dimensional hypervolume, with no external messaging towards users.	System outputs hypervolume, as well as outputting message *"IF the user sees that, they should check themselves out!!!"* to the users.

Reproducing Steps

1. Editing the Front.txt file so each data point has three-dimensional only.
2. Write the input commands in Command Prompt, where the reference point has three-dimensional. Also links correctly with the Front.txt file.
3. Run the program
4. Program outputs defect message

Severity and Priority Reasoning

Severity:
Error contains minimal effect to the output of each Hypervolume.

Priority:
Defect visible to all users when outputting successful Hypervolumes. Decreases the user experience.

Additional Testing

Test 13: Testing negative three-dimensional hypervolumes as well. Showing whether the defect appears for both positive and negative values.

Initially discovered in test 14

Defect ID	07	**Operating System**	Microsoft Windows 10 Pro x64
Defect Type	Functionality	**Testing Tool**	Command Prompt
Test ID affected by Defect	14, 15	**Severity**	High
Program Title	wfg.exe	**Priority**	High
Tester	Christopher Jones		

Defect Description

Whitespacing appearing in the Front.txt file,
causing no Hypervolume to be produced.

Expected Output	**Actual Output**
System to throw an error message revealing the issue to the user.	Program doesn’t output anything to the user.

Reproducing Steps

1. Editing the Front.txt file where it contains whitespace before or after the first hash.
2. Write the input commands in Command Prompt, where the reference point has the same dimensions as all the data points. Also links correctly with the Front.txt file.
3. Run the program
4. Program outputs error

Severity and Priority Reasoning

Severity:
High as it provides an no Hypervolume output or error message notifying the user what the problem is. Affecting the system performance.

Priority:
High as the error is making the program display more unprofessionally as Stakeholders expect more.

Additional Testing

N/A

Initially discovered in test 16

Defect ID	08	**Operating System**	Microsoft Windows 10 Pro x64
Defect Type	Functionality	**Testing Tool**	Command Prompt
Test ID affected by Defect	16, 17, 36, 37	**Severity**	High
Program Title	wfg.exe	**Priority**	Medium
Tester	Christopher Jones		

Defect Description
No Hypervolume output when the structure of the Front.txt file has changed. Similar defect to 07 but not involving Whitespacing.

Expected Output	**Actual Output**
System to throw an error message revealing the issue to the user.	Program doesn't output anything to the user.

Reproducing Steps
1. Editing the Front.txt file where all the data points and hashes are contained on the first line of the file. **2.** Write the input commands in Command Prompt, where the reference point has the same dimensions as all the data points. Also links correctly with the Front.txt file. **3.** Run the program **4.** Program outputs error

Severity and Priority Reasoning
Severity: High as it provides an no Hypervolume output or error message notifying the user what the problem is. Affecting the system performance. **Priority:** Priority not as high as defect 07 as it's less common for users to get the syntax of the Hypervolume to appear the same as this test. The defect still makes the program display more unprofessionally which will ultimately affect user experience.

Additional Testing
Test 36 and 37: Modifying the structure of the Front.txt file to see if the same outcome occurs.

Initially discovered in test 18

Defect ID	09	**Operating System**	Microsoft Windows 10 Pro x64
Defect Type	Functionality	**Testing Tool**	Command Prompt
Test ID affected by Defect	18	**Severity**	Low
Program Title	wfg.exe	**Priority**	Low
Tester	Christopher Jones		

Defect Description

When providing no reference point, the program doesn't use the origin as the default reference point.

Expected Output	**Actual Output**
System to use the origin as the reference point when program can't find one. Produces successful hypervolume.	System throws error message noting that the program requires a reference point. Program terminates.

Reproducing Steps

1. Edit Front.txt file so it is a valid data points and syntax.
2. Write the input commands in Command Prompt, containing no reference point.
3. Run the program
4. Program outputs error

Severity and Priority Reasoning

Severity:
Program doesn't follow what is expected, but still provides a detailed error message with the issue in hand for the user. Doesn't affect system performance as much as previous defects described.

Priority:
Common user error to face is missing certain credentials**.** The error message helps increase the maintainability of the system. Low priority as it doesn't affect the experience of the user as much as other defects found.

Additional Testing

N/A

Initially discovered in test 19

Defect ID	10	**Operating System**	Microsoft Windows 10 Pro x64
Defect Type		**Testing Tool**	Command Prompt
Test ID affected by Defect	19, 20	**Severity**	High
Program Title	wfg.exe	**Priority**	Medium
Tester	Christopher Jones		

Defect Description

Incorrect Hypervolume when running large front or multiple fronts.

Expected Output	**Actual Output**
System outputs the correct Hypervolume to the user	System outputs an incorrect Hypervolume to the user

Reproducing Steps

See tests 19 and 20 for more details

1. Editing the Front.txt file so it contains large fronts/5 or more fronts in the system.
2. Write input commands in Command Prompt, where the reference point has the same dimensions as all the data points. *E.g. Reference point and data point all being two-dimensional values.* Also links correctly with the Front.txt file.
3. Run the program
4. Program outputs error

Severity and Priority Reasoning

Severity:
High as it provides an incorrect Hypervolume output as well as no error messages notifying the user what the problem is. Affecting the system performance in a large scale

Priority:
Giving users no insight as to if there is an error in the system. Causes a lack of trust in the system and ultimately will cause a reduction in the userbase.

Additional Testing

N/A

Initially discovered in test 24

Defect ID	11	**Operating System**	Microsoft Windows 10 Pro x64
Defect Type	Functionality	**Testing Tool**	Command Prompt
Test ID affected by Defect	24	**Severity**	Medium
Program Title	wfg.exe	**Priority**	Medium
Tester	Christopher Jones		

Defect Description
Invalid output when both reference point and front.txt is not included in the input commands

Expected Output	**Actual Output**
System to throw an error message revealing the that the reference point and front.txt could not be found.	System outputs long string of characters and integers.

Reproducing Steps
1. Write the input commands in Command Prompt, but not include any reference points or a front.txt. **2.** Run the program **3.** Program outputs error

Severity and Priority Reasoning
Severity: High as it makes the program look unprofessional and affects the user satisfaction with the product. **Priority:** Defect has a drastic effect to UX for some.

Additional Testing
N/A

Initially discovered in test 25

Defect ID	12	**Operating System**	Microsoft Windows 10 Pro x64
Defect Type	Functionality	**Testing Tool**	Command Prompt
Test ID affected by Defect	25	**Severity**	High
Program Title	wfg.exe	**Priority**	High
Tester	Christopher Jones		

Defect Description
Invalid output when the user doesn't have a *cygwin1.dll* file

Expected Output	**Actual Output**
System to throw an error message revealing the that the .dll file was not found.	System throws an error message revealing that *"File 5 could not be opened".*

Reproducing Steps
1. Remove the .dll file from the directory. **2.** Write the input commands in Command Prompt, where the reference point has the same dimensions as all the data points. Also links correctly with the Front.txt file **3.** Run the program **4.** Program outputs error

Severity and Priority Reasoning
Severity: Due to how necessary the .dll file is for this program to function properly, it should be expected to provide error messages when that file is missing. Affects user performance. **Priority:** High chance of this defect appearing for users. Top priority.

Additional Testing
N/A

Initially discovered in test 27

Defect ID	13	**Operating System**	Microsoft Windows 10 Pro x64
Defect Type	Functionality	**Testing Tool**	Command Prompt
Test ID affected by Defect	27	**Severity**	Low
Program Title	wfg.exe	**Priority**	Medium
Tester	Christopher Jones		

Defect Description
When the program attempts to execute when there is an empty Front.txt file, it then produces a message targeted towards the user that has a misspelt word.

Expected Output	**Actual Output**
Output message "Reference point"	Output message "Reference Pint"

Reproducing Steps
1. Editing the Front.txt file so there are no data points in the file, but still contains a front. **2.** Write input commands in Command Prompt, where the reference point is a valid two-dimensional. Also links correctly with the Front.txt file. **3.** Run the program **4.** Program outputs invalid error message

Severity and Priority Reasoning
Severity: Error contains minimal effect to the output of each Hypervolume. **Priority:** Defect visible to all users when outputting successful Hypervolumes. Decreases the user experience.

Additional Testing
N/A

Initially discovered in test 27

Defect ID	14	**Operating System**	Microsoft Windows 10 Pro x64
Defect Type	Functionality	**Testing Tool**	Command Prompt
Test ID affected by Defect	27	**Severity**	Medium
Program Title	wfg.exe	**Priority**	High
Tester	Christopher Jones		

Defect Description	
When the program attempts to execute when there is an empty Front.txt file, it then notifies the user the wrong information. Showing the wrong number of values that the reference should contain.	
Expected Output	**Actual Output**
Output message "Invalid data values, please try again with a dimension of 2"	"Your reference pint should have 0 values".
Reproducing Steps	
1. Editing the Front.txt file so there are no data points in the file, but still contains a front. **2.** Write input commands in Command Prompt, where the reference point is a valid two-dimensional. Also links correctly with the Front.txt file. **3.** Run the program **4.** Program outputs invalid error message	
Severity and Priority Reasoning	
Severity: High as it makes the program look improper and affects the user satisfaction with the product. **Priority:** Defect has a drastic effect to UX for some users	
Additional Testing	
N/A	

Initially discovered in test 29

Defect ID	15	**Operating System**	Microsoft Windows 10 Pro x64
Defect Type	Functionality	**Testing Tool**	Command Prompt
Test ID affected by Defect	29	**Severity**	High
Program Title	wfg.exe	**Priority**	High
Tester	Christopher Jones		

Defect Description

When the program executes when there are invalid data values in the Front.txt file (letters instead of numbers), the system then outputs an incorrect Hypervolume instead of throwing an error message.

Expected Output	**Actual Output**
System to throw an error message revealing the issue to the user.	Produce incorrect Hypervolume

Reproducing Steps

1. Editing the Front.txt file where it contains some invalid data points (letters instead of numbers), all data points are the same dimensions.
2. Write input commands in Command Prompt, where the reference point have the same dimensions as all the data points. Also links correctly with the Front.txt file.
3. Run the program
4. Program outputs invalid Hypervolume

Severity and Priority Reasoning

Severity:
High as it makes the program look unprofessional and affects the user satisfaction with the product.

Priority:
Defect has a drastic effect to UX for some.

Additional Testing

N/A

Initially discovered in test 32

Defect ID	16	**Operating System**	Microsoft Windows 10 Pro x64
Defect Type	Functionality	**Testing Tool**	Command Prompt
Test ID affected by Defect	32, 34, 35	**Severity**	High
Program Title	wfg.exe	**Priority**	High
Tester	Christopher Jones		

Defect Description

Program is executed when there are multiple fronts in the Front.txt file, some of them containing no data values whatsoever.

Expected Output	**Actual Output**
System to throw an error message revealing the that some of the fronts are invalid	System enters an infinite loop and never terminates or reveals an error message to the user.

Reproducing Steps

1. Editing the Front.txt file so it contains multiple fronts, some of them containing no data values.
2. Write input commands in Command Prompt, where the reference point has the same dimensions as all the data points. *E.g. Reference point and data point all being two-dimensional values.* Also links correctly with the Front.txt file.
3. Run the program
4. Program outputs error

Severity and Priority Reasoning

Severity:
Critical defect that breaks the program. Defect causes the user to terminate the program manually to get out of the infinite loop

Priority:
Defect affects the functionality and reliability of the system. Makes the program look very unprofessional. Defect needs to be removed ASAP.

Additional Testing

Test 35: Tests multiple fronts where all of them don't contain data values. This can show that the user providing some valid fronts in the system when there are already invalid fronts in there makes no difference.

Initially discovered in test 38

Defect ID	17	**Operating System**	Microsoft Windows 10 Pro x64
Defect Type	Functionality	**Testing Tool**	Command Prompt
Test ID affected by Defect	38	**Severity**	Medium
Program Title	wfg.exe	**Priority**	Medium
Tester	Christopher Jones		

Defect Description	
Program is producing an invalid error message, occurs when the data values are 3-dimensional, but the reference point is only 2-dimensional.	
Expected Output	**Actual Output**
System to throw an error message revealing the issue to the user.	System outputs error message containing incorrect information to the user
Reproducing Steps	
1. Editing the Front.txt file so it one 3-dimensional front. **2.** Write input commands in Command Prompt, where the reference point is only 2-dimensional and links correctly with the Front.txt file. **3.** Run the program **4.** Program outputs error	
Severity and Priority Reasoning	
Severity: High as it is providing users with invalid error messages, causing confusion. Affecting the usability and functionality of the system **Priority:** High chance of this defect appearing for users. Defect has a drastic effect to UX for some.	
Additional Testing	
Test 39: Testing when the data values are 4-dimensional, but Reference Point is 2-dimensional. Checking how it compares with Test 38	

Initially discovered in test 40

Defect ID	18	**Operating System**	Microsoft Windows 10 Pro x64
Defect Type	QOL	**Testing Tool**	Command Prompt
Test ID affected by Defect	40	**Severity**	Low
Program Title	wfg.exe	**Priority**	Medium
Tester	Christopher Jones		

Defect Description

When the program attempts to execute when there is an empty Front.txt file or reference point, it provides a misspelt error message.

Expected Output	**Actual Output**
System to throw an error message revealing the that some of the fronts are invalid	System to throw an error message revealing the that some of the fronts are invalid but contains misspellings. *"No referrence point provided"*

Reproducing Steps

1. Editing the Front.txt file so it contains an empty front
2. Write input commands in Command Prompt, where it contains no reference point. Link the Front.txt file successfully.
3. Run the program
4. Program outputs error

Severity and Priority Reasoning

Severity:
Error contains minimal effect to the output of each Hypervolume.

Priority:
Defect visible to all users when outputting successful Hypervolumes. Decreases the user experience.

Additional Testing

N/A

Initially discovered in test 28

Defect ID	19	**Operating System**	Microsoft Windows 10 Pro x64
Defect Type	QOL	**Testing Tool**	Command Prompt
Test ID affected by Defect	28	**Severity**	Low
Program Title	wfg.exe	**Priority**	Medium
Tester	Christopher Jones		

Defect Description
When the program attempts to execute when no Front.txt file is found, an error message appears which contains a misspelt word.

Expected Output	**Actual Output**
System to throw an error message revealing the that some of the fronts are invalid, with no spelling or grammar errors.	System to throw an error message revealing the that some of the fronts are invalid but contains misspellings. *"File Front.txt could note be opened"*

Reproducing Steps
1. Editing the Front.txt file so it contains an no front file in the directory **2.** Write input commands in Command Prompt, where it contains reference points (2 dimension or higher). Link the Front.txt file successfully. **3.** Run the program **4.** Program outputs error

Severity and Priority Reasoning
Severity: Error contains minimal effect to the output of each Hypervolume. Affecting the usability and functionality of the system **Priority:** Defect visible to all users when outputting successful Hypervolumes. Decreases the user experience.

Additional Testing
N/A

4 Defect List

1. *"Why bother????" message that appears when a hypervolume has been produced.*
2. System outputting the same hypervolume twice for three-dimensional outputs.
3. "Two fronts here we come:" message that appears when two fronts are attempting to be produced.
4. *"Clearly a greedy user!" output message that appears when three fronts are attempting to be produced*
5. Hypervolume: *%\%* appearing when the reference points are less than or equal to zero
6. *"IF the user sees that, they should check themselves out!!!" output message that appears when three-dimensional hypervolumes are attempting to be produced.*
7. No output when Whitespacing has been added to the Front.txt file
8. No Hypervolume outputting when the structure of the Front.txt file has been altered that doesn't contain Whitespacing.
9. System throwing an error message when the reference point is missing, instead of creating a Hypervolume from the origin.
10. Incorrect Hypervolumes when running many data points or 5 or more fronts in a single process.
11. Invalid output when both reference point and Front file is not found.
12. Invalid output when the cygwin.dll file is not included with the system
13. "Reference Pint" misspelt output message.
14. Invalid error message when there is an empty Front.txt file
15. Infinite loop error when one of the fronts contains no data values while executing multiple fronts in one process.
16. Produce incorrect Hypervolumes when the user inputs invalid data values (letters instead of numbers)
17. Outputting the incorrect error message when the reference point and data values have different dimensions.
18. "No referrence point provided" misspelt output message
19. "File Front.txt could note be opened" misspelt output message

5 References

Rongala, A. (2015). What is White Box Software Testing: Advantages and Disadvantages - Invensis Technologies. [online] Invensis Technologies. Available at: https://www.invensis.net/blog/it/white-box-software-testing-advantages-disadvantages/.

Altexsoft.com. (2018). 6 Ways to Improve Software Testing. [online] Available at: https://www.altexsoft.com/blog/engineering/6-ways-to-improve-software-testing-through-planning-work-environment-automated-testing-and-reporting/ [Accessed 29 Nov. 2018].