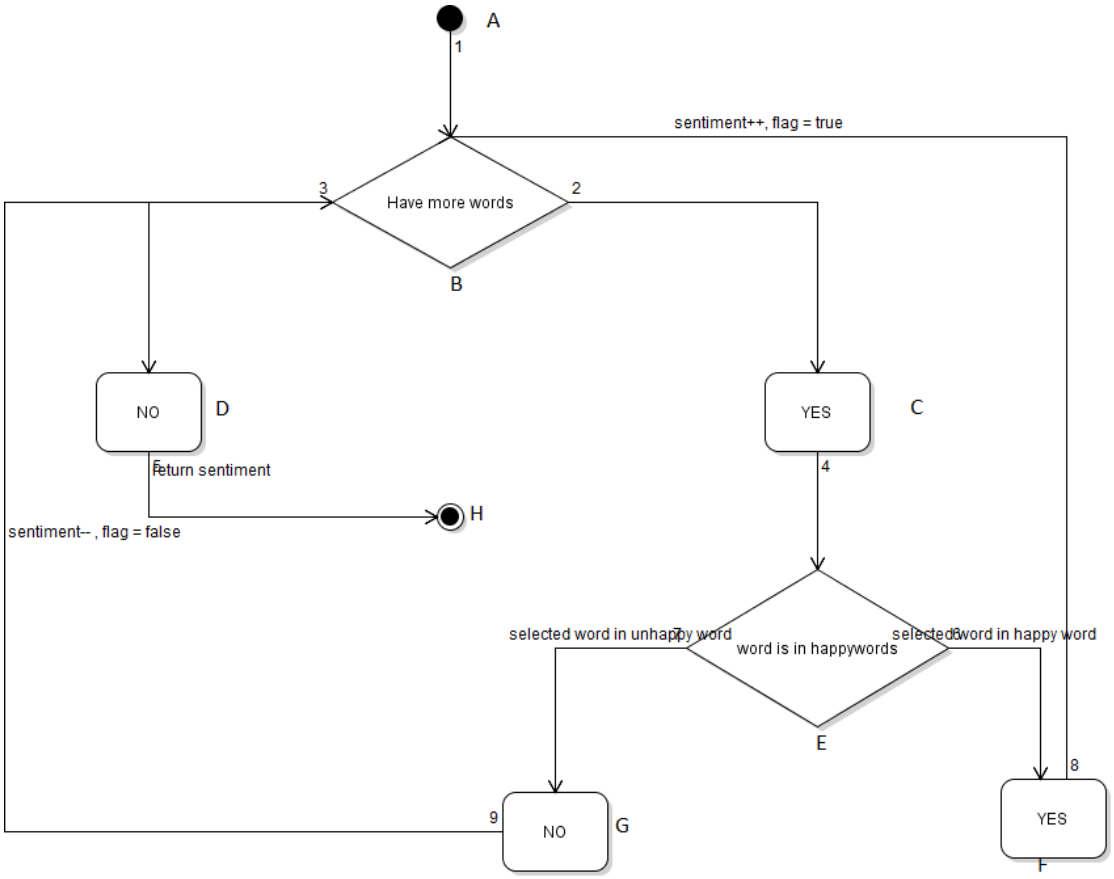
Model Based Testing – Part 3:



2.

In the above diagram, control flow graph of code is depicted. The first case involves traversal through the "Yes "decision and covered edges 1, 2, 4, 6 and 8 but edges 3,5,7 and 9 are not covered in this path. Second case involves traversal through the "No" decision, and covered edges 1, 3 and 5 but edges 2,4,6, 7,8 and 9 are not covered in this path. Third case involves traversal through the "No" decision and covered edges is 1, 2, 4, 7 and 9 but edges 3,5 and 6 are not covered in this path.

Path 1 - A1-B2-C4-E6-F8

Path 2 - A1-B3-D5

Path 3 - A1-B2-C4-E7-G9

Branch Coverage (BC) = Number of paths

=3

3.

Scenario : text = “happy unfortunate”

We can see that after first step we get two words in our word array. Then as first word is happy and second word is unhappy, first iteration satisfy first condition and sentiment is increased by 1. In second iteration, the second condition is satisfied and sentiment decreases by 1. In third iteration it goes out of the while loop. So, it covers all the statements of the code. Last line of the code which prints Sentiment analysis complete is never executed as the sentiment is returned before that line.

4.

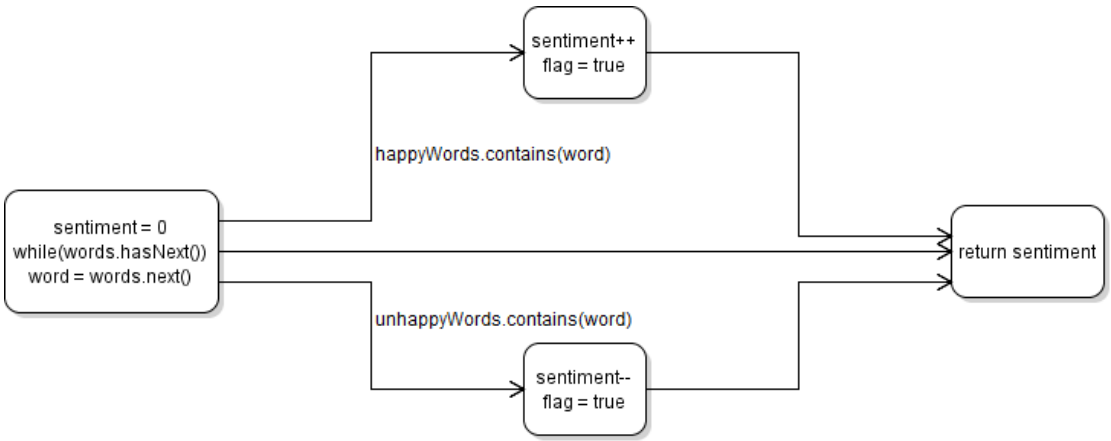
Path 1 - A1-B2-C4-E6-F8

Path 2 - A1-B3-D5

Path 3 - A1-B2-C4-E7-G9

In the above paths, the first path follows the scenario where the text has one happy word. Second path follows the scenario where the text contains no word which could be analyzed for sentiment. Third path follows the scenario in which text contains one unhappy word and decreases the sentiment by 1.

5.



6.

To calculate Statement Coverage, find out the shortest number of paths following which all the nodes will be covered. By traversing through path A1-B2-C4-E6-F8 all the nodes are covered. Therefore, by traveling through only one path, the nodes 1,2,4,6 and 8 are covered, so nodes 3, 5 and 7 not covered. We need a test case which contains at least one happy word and at least one unhappy word to cover all statements.

7.

Static code analysis is a method of debugging by examining source code before a program is run. It's done by analyzing a set of code against a set (or multiple sets) of coding rules. Static code analysis and static analysis are often used interchangeably, along with source code analysis. This type of analysis addresses weaknesses in source code that might lead to vulnerabilities. Of course, this may also be achieved through manual code reviews. But using automated tools is much more effective.

8.

The Boolean flag which is declared in line 3 is initiated as false, if we have one word in argument words then the flag will be true irrespective of word being a happy word or unhappy word. We can just remove the statement which is written twice and add it at the end of the while loop. Furthermore, the last line of the code which prints Sentiment analysis complete is never executed.