**CMPE 287 – Software Quality Assurance and Testing**

**Homework #3**

**First Name:** Harish

**Last Name:** Marepalli

**SJSU ID:** 016707314

**Professor:** Dr. Jerry Gao

**Question #1: Software Integration Testing (25%)**

1. (20%) Questions about software integration based on a program structure shown in Figure 1:
2. Explain the idea of the top-down integration approach. (3%)
3. List an integration sequence using the top-down approach and its required test stubs (or drivers) using the breath-first order. (5%)
4. List the required total number of test stubs. (5%)
5. Explain the idea of the bottom-up integration approach. (2%)
6. List an integration sequence using the bottom-up approach and its required test drivers. (5%)
7. List the required total number of test drivers. (5%)

Hints: Please writes your integration sequence as follows:

A

1. Integrated system (IS) = module A and module B (required test stubs for C,D and E)

E

1. Integrated system (IS) = IS and module D

C

B

……….

**Figure 1.**

M1

M3

M2

M4

M5

M7

M10

M9

M6

M12

M11

M8

M13

D

**Answer:**

**1.a)** Top-down Integration Approach:

The top-down integration approach starts with the high-level control modules and works downwards to integrate lower-level components of the system. In this approach, the main control module is developed and tested first. Then the next level modules that are invoked by the main control module are integrated and tested. This continues downwards until all the system components are integrated from top to bottom.

Some key points about top-down integration:

* It begins with the main control module that drives the rest of the system. This allows early testing of critical system functionality.
* High-level modules are developed and tested before detailed lower-level modules. This follows natural abstraction layers in the system architecture.
* Requires stubbing/mocking of components not yet integrated. Stubs emulate lower-level modules that are not yet complete.
* Testing focuses on integration between modules. Allows divide-and-conquer testing of complex systems.
* Promotes early detection of system-level architectural or design flaws. High-level issues can be found before detailed lower-level development.
* Dependency issues are discovered earlier. Missing or conflicting interfaces between modules become evident.
* Can be less flexible for incremental or iterative development. Full high-level design needs to be completed upfront.

**1.b)** Integration sequence using top-down approach:

Integration Order: Breadth -First (Left Order)

IS: Integrated System Mi’: Software stub for module Mi.

Step #1: IS = M1 and M2 (need: M3’, M13’, M4’, and M5’)

Step #2: IS = IS + M3 (need: M13’, M4’, M5’, and M6’)

Step #3: IS = IS + M13 (need: M4’, M5’, M6’, and M7’)

Step #4: IS = IS + M4 (need: M5’, M6’, M7’, and M8’)

Step #5: IS = IS + M5 (need: M6’, M7’, M8’, and M9’)

Step #6: IS = IS + M6 (need: M7’, M8’, M9’, and M10’)

Step #7: IS = IS + M7 (need: M8’, M9’, M10’, M11’, and M12’)

Step #8: IS = IS + M8 (need: M9’, M10’, M11’, and M12’)

Step #9: IS = IS + M9 (need: M10’, M11’, and M12’)

Step #10: IS = IS + M10 (need: M11’, and M12’)

Step #11: IS = IS + M11 (need: M12’)

Step #12: IS = IS + M12

**1.c)** Total number of stubs:

The list of total number of stubs is:

1. M3’
2. M4’
3. M5’
4. M6’
5. M7’
6. M8’
7. M9’
8. M10’
9. M11’
10. M12’
11. M13’

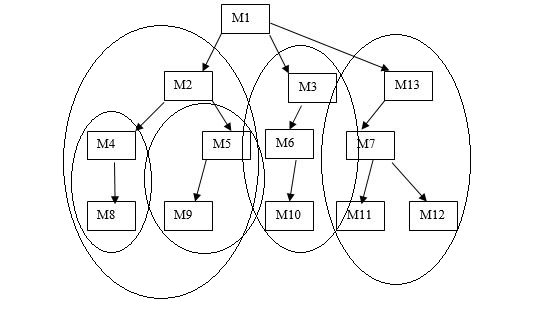
**1.d)** Bottom-up Integration Approach:

The bottom-up integration approach starts with the low-level components and works upwards to integrate higher-level modules of the system. In this approach, the lowest level components like utilities or drivers are developed and tested first. Then progressively higher-level components that use these lower modules are integrated and tested. This continues upwards until the complete system is integrated from bottom to top.

Some key points about bottom-up integration:

* It begins with detailed low-level modules like device drivers and libraries. These components have limited dependencies.
* Lower-level components are integrated before high-level components that depend on them. Follows dependency hierarchy.
* Allows early development of critical low-level functions like hardware interaction.
* Testing focuses on components progressively, simplifying troubleshooting.
* Allows incremental development and testing of subsystem functionality.
* High-level components may require stubs/mocks representing lower-level modules.
* Architectural issues detected later in development lifecycle after detailed design.
* Less big-picture visibility till high-level modules integrated and tested.

**1.e)** Integration sequence using bottom-up approach:



Integration Order: Breadth-First (Left Order)  
  
IS: Integrated System Mi”: software driver for Module Mi.

Step #1: IS1 = M8 + M4 (need: M2”)

Step #2: IS2 = M9 + M5 (need: M2”)

Step #3: IS3 = IS1 + M2 (need: M1”)

Step #4: IS3 = IS3 + IS2 (need: M1”)

Step #5: IS4 = M10 + M6 (need: M3”)

Step #6: IS4 = IS4 + M3 (need: M1”)

Step #7: IS5 = M11 + M7 (need: M13”)

Step #8: IS5 = IS5 + M12 (need: M13”)

Step #9: IS5 = IS5 + M13 (need: M1”)

Step #10: IS = IS3 + M1 (need: M3” and M13”)

Step #11: IS = IS + IS4 (need: M13”)

Step #12: IS = IS + IS5

**1.f)** Total number of test drivers:

The list of total number of test drivers is:

1. M1”
2. M2”
3. M3”
4. M13”

The list of total number of stubs is:

1. M4’
2. M5’
3. M6’
4. M7’
5. M8’
6. M9’
7. M10’
8. M11’
9. M12’

**Question #2: Questions about Software Regression Testing (40%)**

Figure 2 shows the class relation structure among the classes of a program written in C++. “I” indicates an Inheritance relation between two classes. “AG” represents an aggregation relation between two class objects, “AS” indicates an association relation between two classes. Assume Class 7 and Class 1 are changed. Please follow the class firewall’s concept to identify a class firewall to enclose all affected classes and re-integration links after changing Class 7 and Class 1. (Note: pleased Please present this class firewall by answering the following questions:

1. (10%) What is the class test order based on the class diagram in Figure 2.
2. (10%) What are the class firewalls for the two changed classes, and please present the two firewalls separately using a diagram.
3. (10%) Which classes are not directly affected by each changed class, but must be re-integrated?
4. (10%) Please list the class re-test order for each detected class firewall.

You may need to read the published papers below:

Paper 1: Class Firewall, Test Order, and Regression Testing of Object-Oriented Programs, Journal of Object-Oriented Programming, 1993.

by David C. Kung, Jerry Gao , Pei Hsia , Jeremy Lin , Yasufumi Toyoshima

URL: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.53.6284>

Paper 2: A Test Strategy for Object-Oriented Programs

[By D. Kung](http://ieeexplore.ieee.org/search/searchresult.jsp?searchWithin=%22Authors%22:.QT.D.%20Kung.QT.&newsearch=true); [J. Gao](http://ieeexplore.ieee.org/search/searchresult.jsp?searchWithin=%22Authors%22:.QT.J.%20Gao.QT.&newsearch=true); [Pei Hsia](http://ieeexplore.ieee.org/search/searchresult.jsp?searchWithin=%22Authors%22:.QT.%20Pei%20Hsia.QT.&newsearch=true); [Y. Toyoshima](http://ieeexplore.ieee.org/search/searchresult.jsp?searchWithin=%22Authors%22:.QT.Y.%20Toyoshima.QT.&newsearch=true) ; [C. Chen](http://ieeexplore.ieee.org/search/searchresult.jsp?searchWithin=%22Authors%22:.QT.C.%20Chen.QT.&newsearch=true),

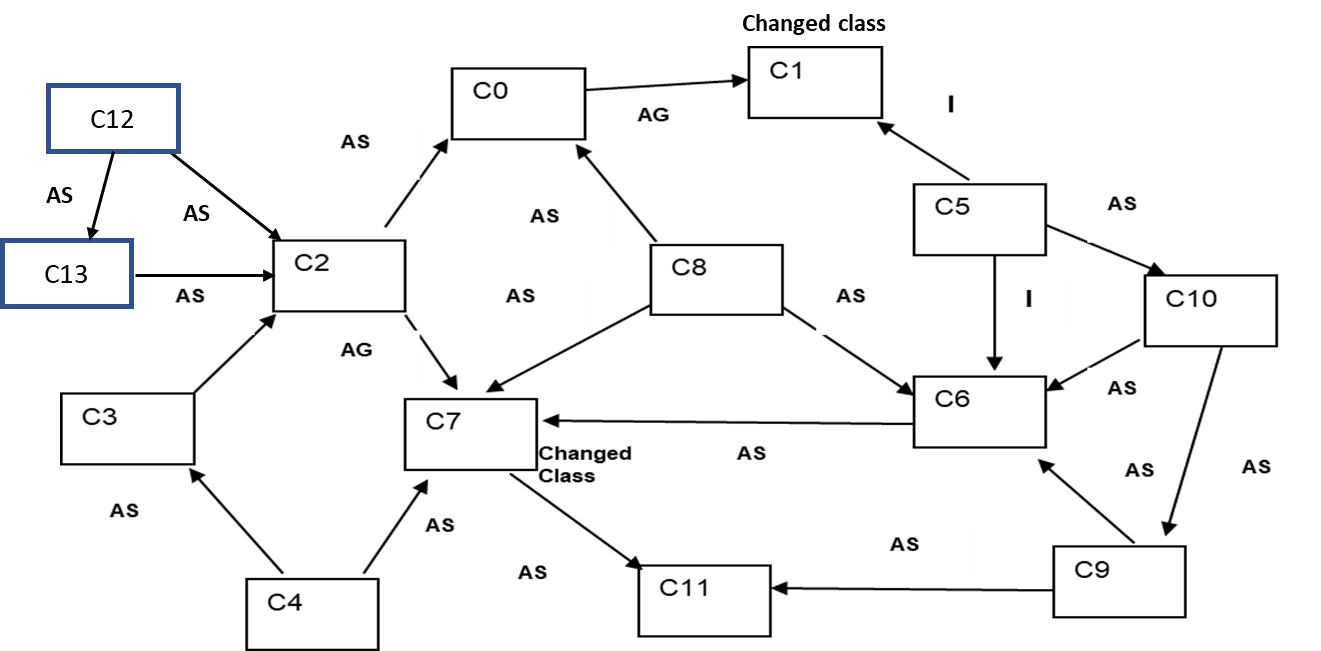
Proceedings., Nineteenth Annual International conference on Computer Software and Applications, 1995. (COMPSAC 95)

URL: <http://ieeexplore.ieee.org/abstract/document/524786/>

Paper 3: Change Impact Identification in Object-Oriented Software Maintenance

By D. Kung, J. Gao, P. Hsia, F. Wen, Y. Toyoshima, Ray Chen

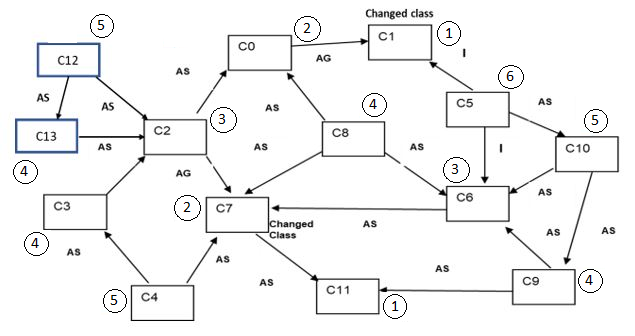
URL: http://ieeexplore.ieee.org/abstract/document/336774/

****

**Figure 2.**

**Answer:**

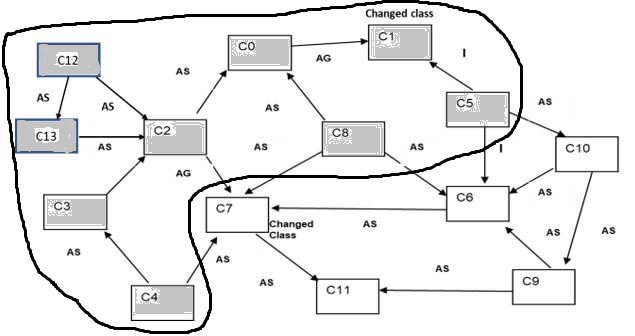
**2.a)** The class test order is as follows:



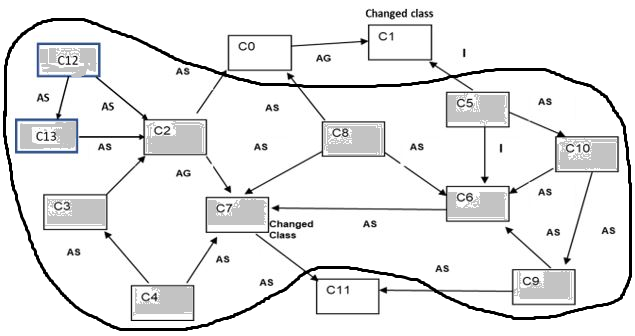
1. C1, C11
2. C0, C7
3. C2, C6
4. C3, C8, C9, C13
5. C4, C10, C12
6. C5

**2.b)** Class firewalls:

When C1 is changed, the class firewalls are C0, C1, C2, C3, C4, C5, C8, C12, C13.



When C7 is changed, the class firewalls are C2, C3, C4, C5, C6, C7, C8, C9, C10, C12, C13.



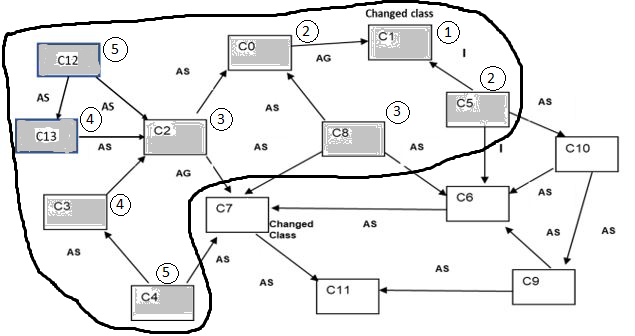
**2.c)** List of classes not directly affected by each changed class.

By the change of C1, the classes C6, C7, C9, C10, and C11 are not directly affected. Among these, only C6, C7, and C10 classes must be re-integrated as they directly depend on the class firewall.

By the change of C7, the classes C0, C1, and C11 are not directly affected. All three of these classes must be re-integrated as they directly depend on the class firewall.

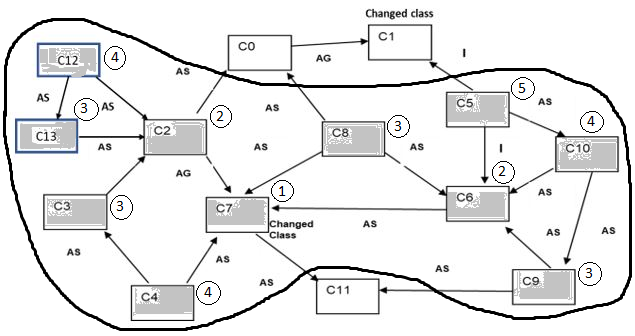
**2.d)** List of class re-test order for each detected class firewall.

The class re-test order for the change of C1 is as follows:



1. C1
2. C0, C5
3. C2, C8
4. C3, C13
5. C4, C12

The class re-test order for the change of C7 is as follows:



1. C7
2. C2, C6
3. C3, C13, C8, C9
4. C12, C4, C10
5. C5

**Question #3: Testing a selected AI Mobile App (35%)**

For your selected intelligent function/feature of your mobile app (or online app), please focus on one of your selected AI-powered function and conduct your 3D AI test modeling based on our lecture and given 3D AI test tool.

Please answer the following questions:

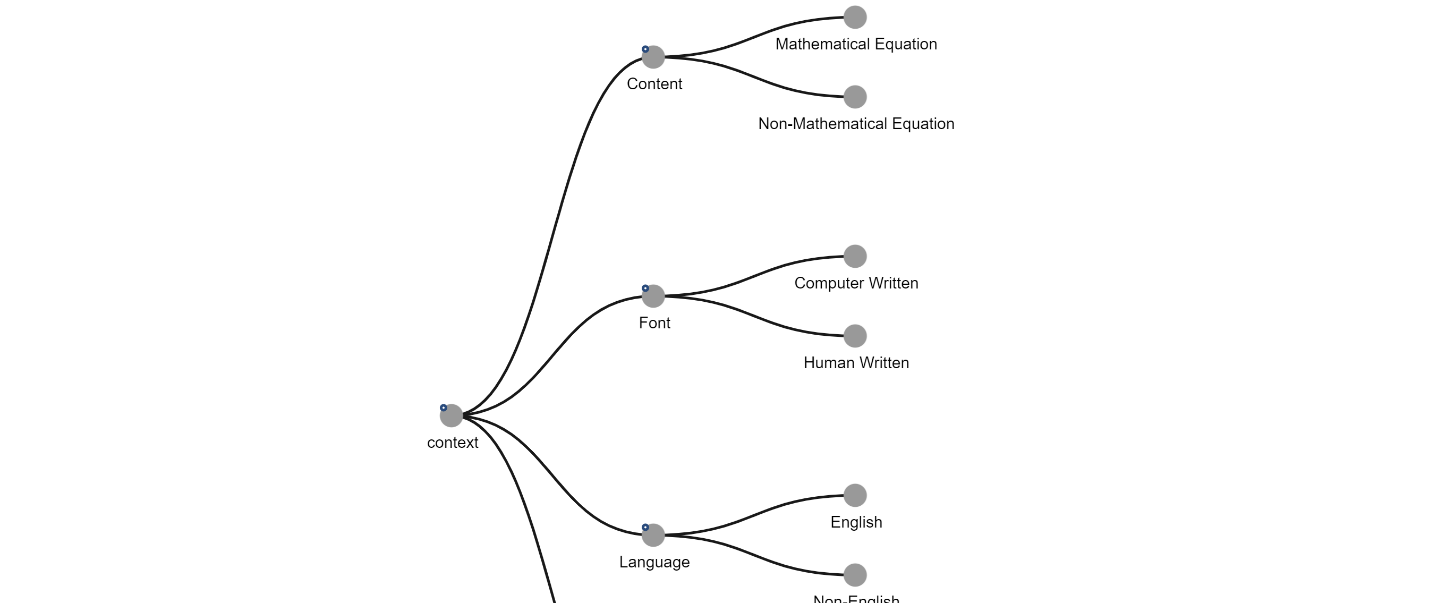
1. Your 3D AI test modeling test model: (20%)
   1. Context Model (Context Modeling Tree) (5%)
   2. Input Classification Model (Input Classification Tree) (5%)
   3. Output Classification Model (Output Classification Tree) (5%)
   4. 3D Decision Table (10%)
2. Your test complexity based on your 3D AI decision table (5%)
3. Present three your selected bugs you detected. (5%)

Please present your bugs in a well-defined format.

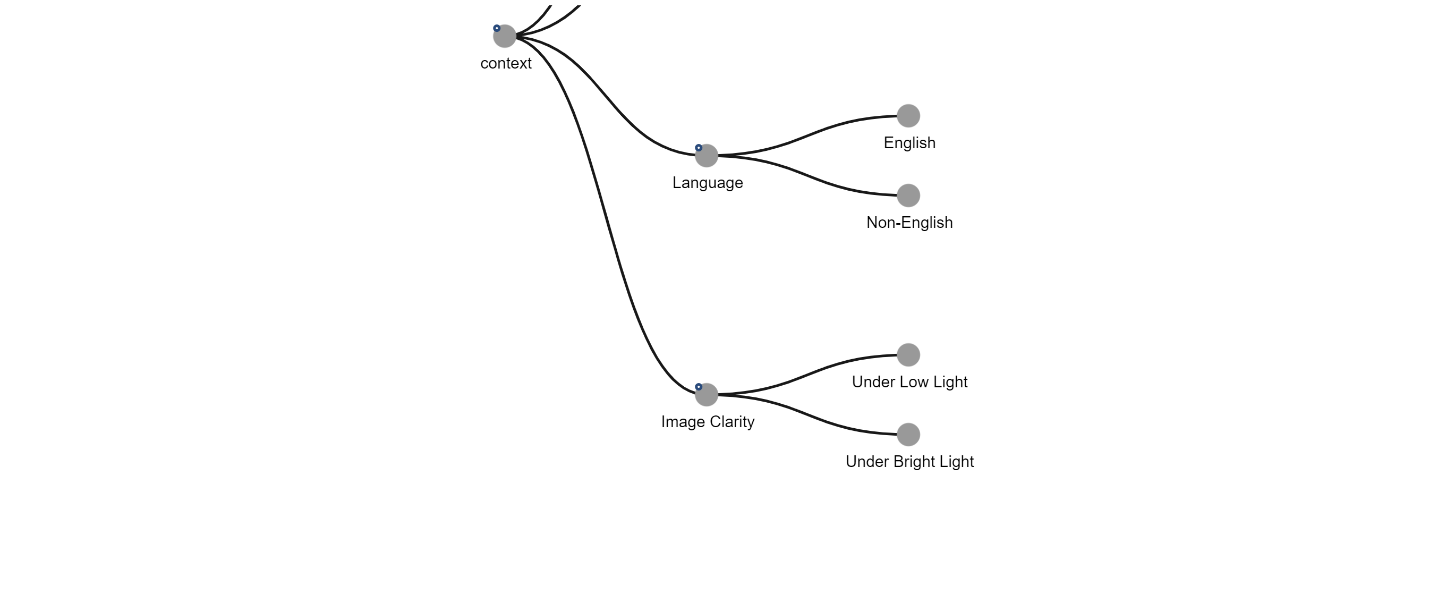
**Answer:**

**3.a)** 3D AI test modeling test model:

**a.** Context Model (Context Modeling Tree):

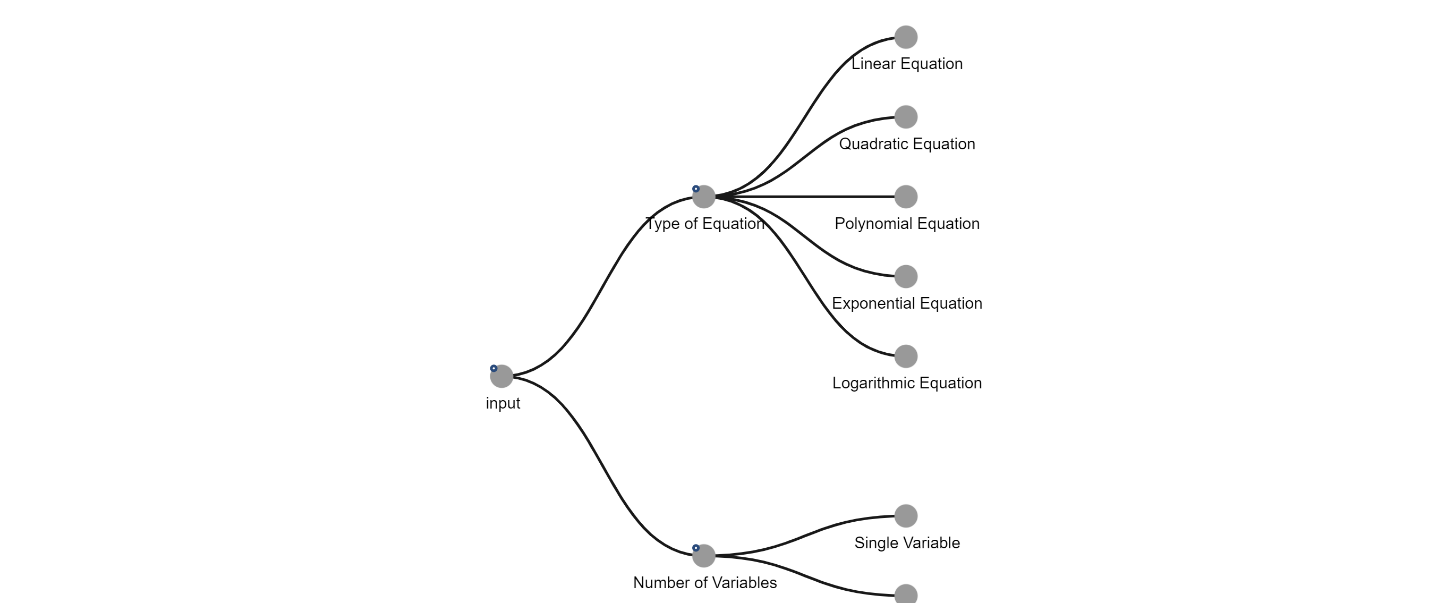


***Context Tree Model***

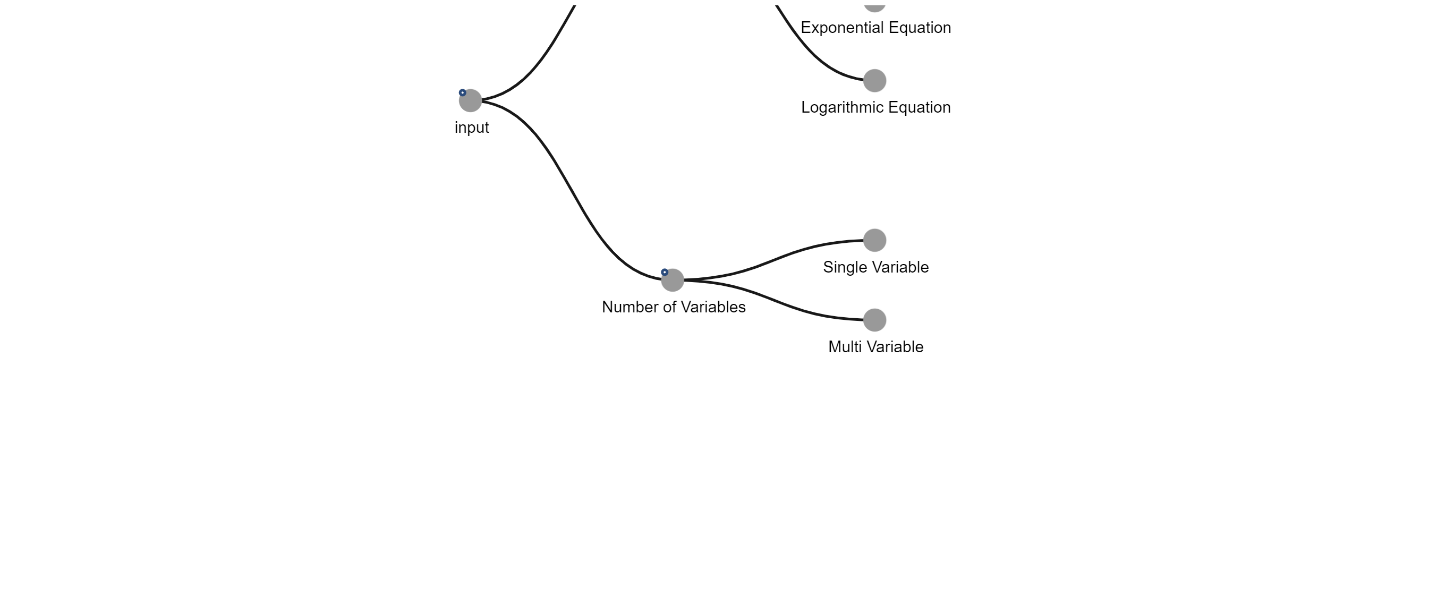


***Context Tree Model***

**b.** Input Classification Model (Input Classification Tree):

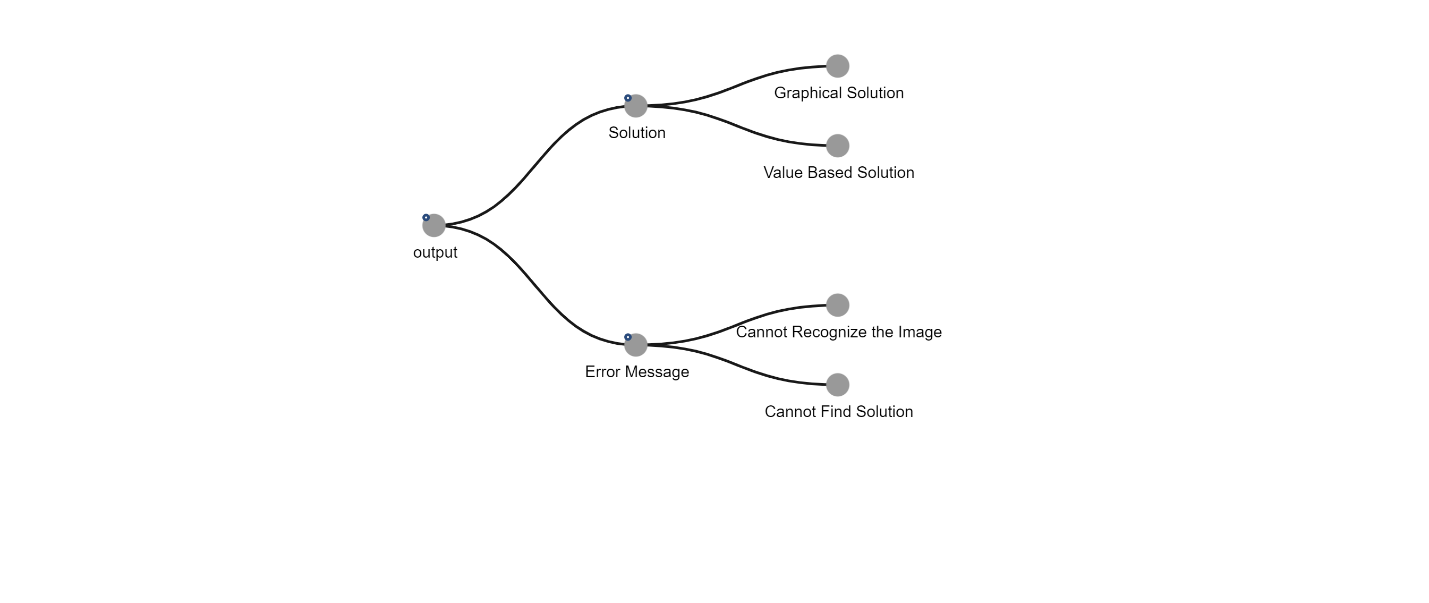


***Input Tree Model***



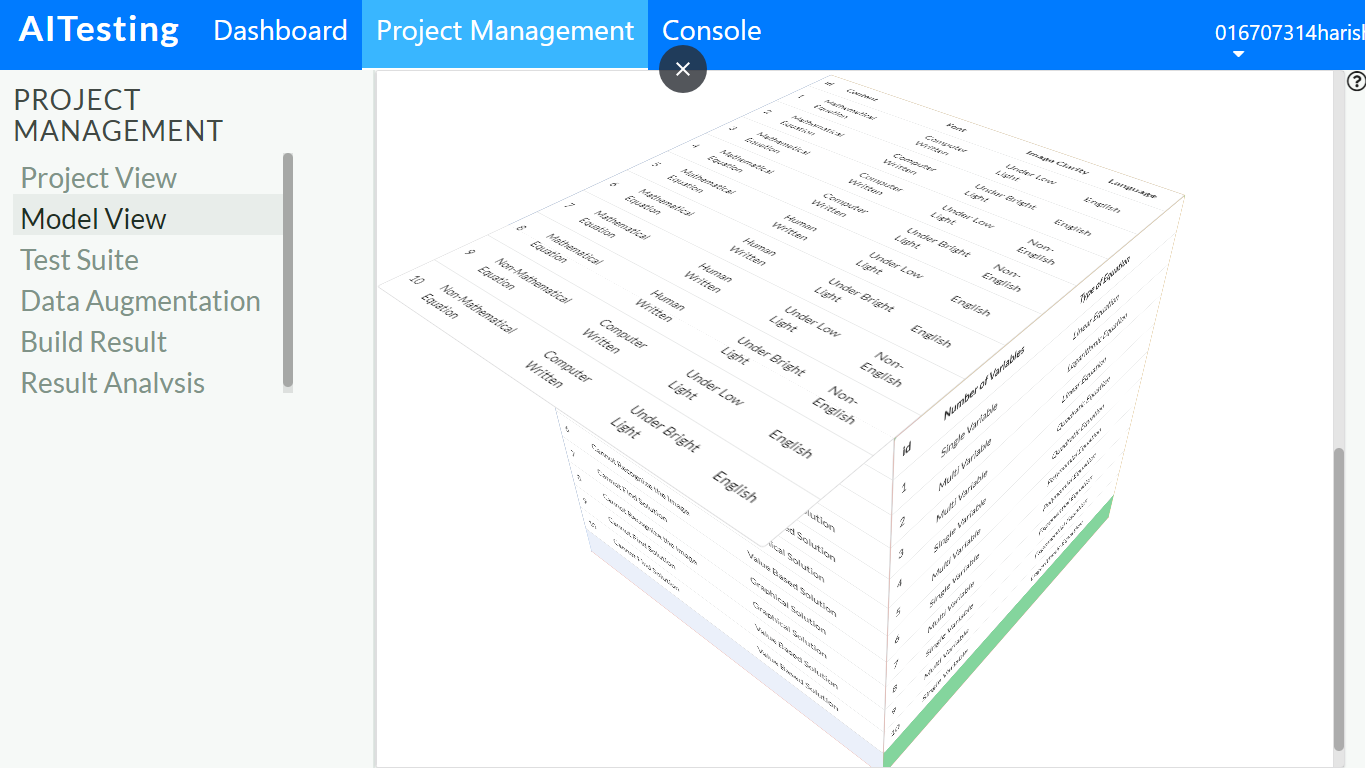
***Input Tree Model***

**c.** Output Classification Model (Output Classification Tree):

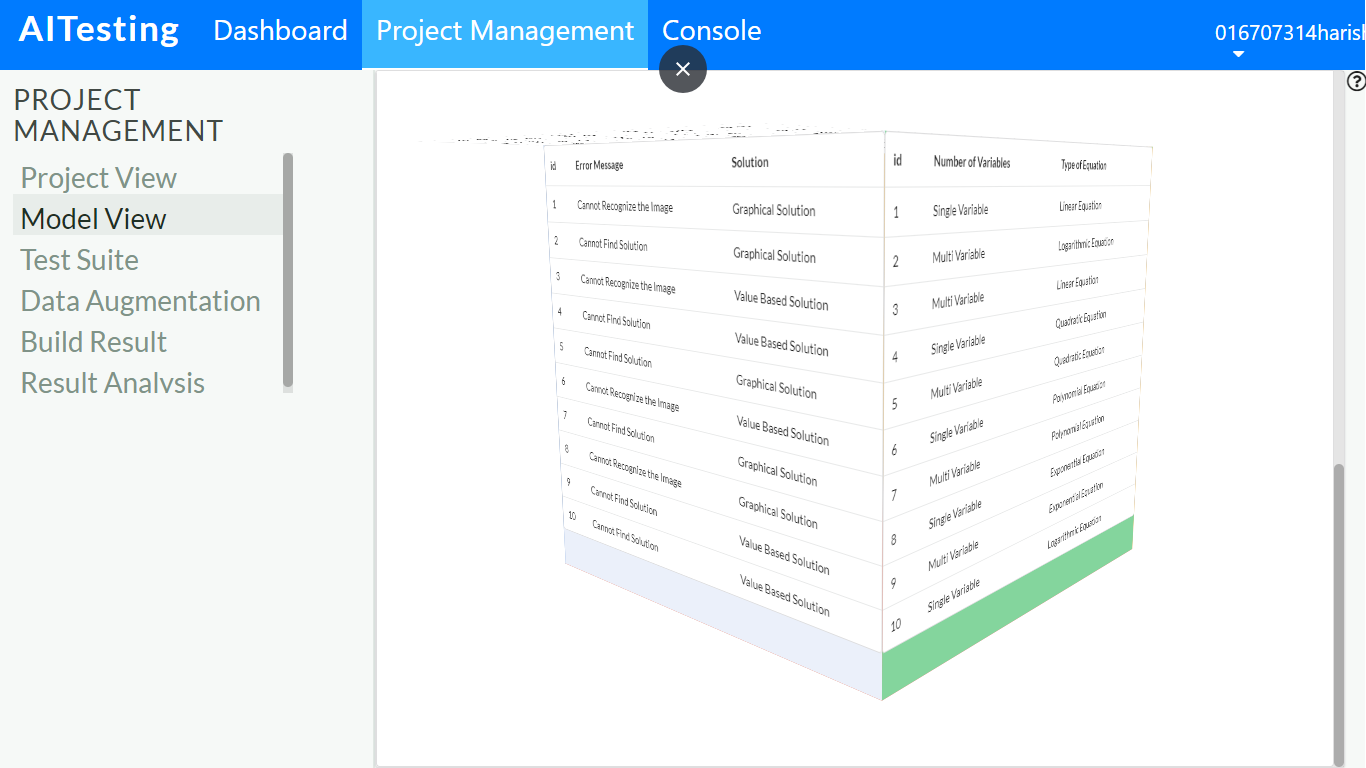


***Output Tree Model***

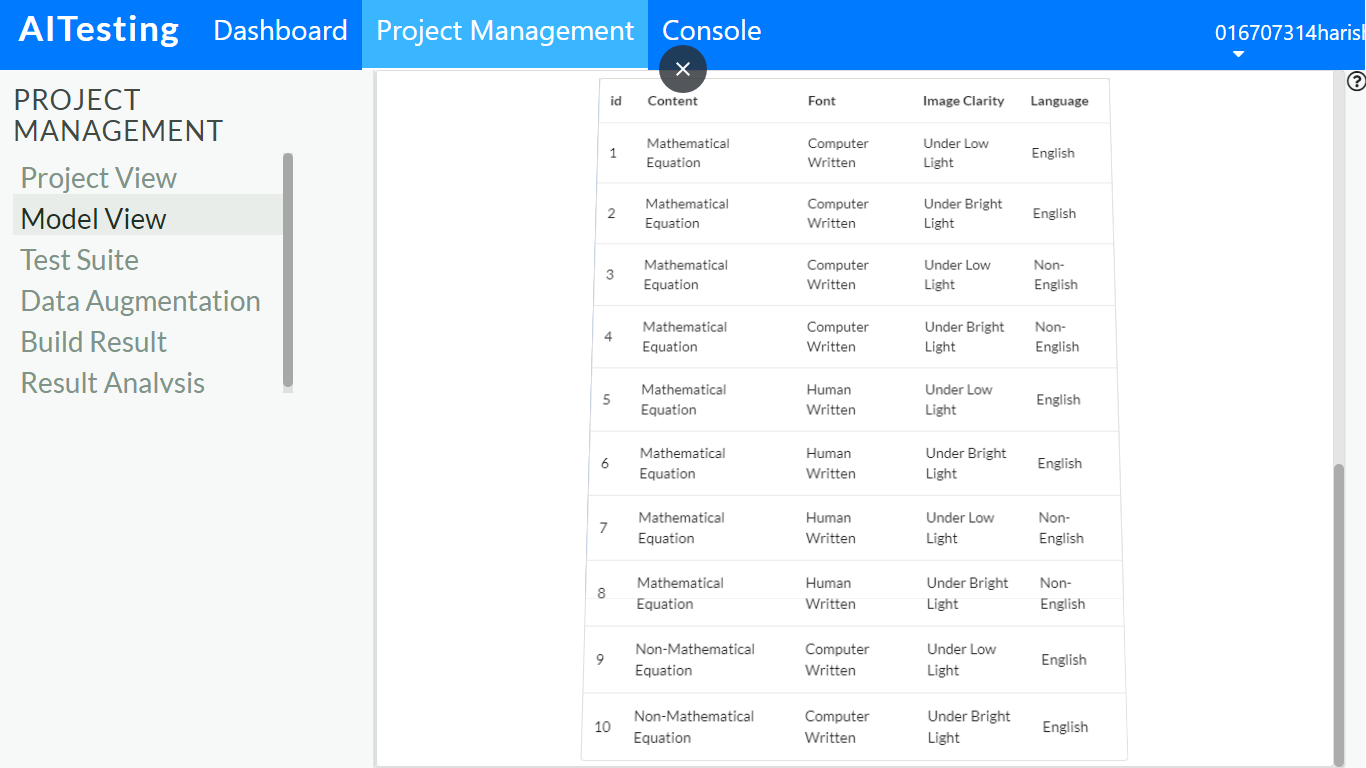
**d.** 3D Decision Table:



***3D Decision Table***



***3D Decision Table***



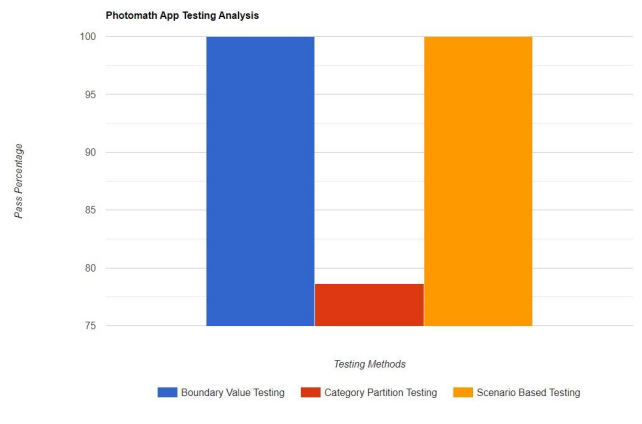
***3D Decision Table***

**3.b)** Test complexity based on 3D AI decision table:

We used black box testing techniques like Boundary value testing, category-based testing, scenario-based testing and equivalence partitioning testing for conventional testing. In regard of AI testing we used an AI tool for generating various test cases based on various context features.

We have 160 test cases generated by AI testing tool and for conventional testing we had a total of 72 test cases.

|  |  |
| --- | --- |
| Test Method | Photomath |
| Boundary Value Testing | 100% |
| Category Partition Testing | 78.62% |
| Scenario Testing | 100% |



1. Different model-based text complexity

* Test complexity for context model:

No. of branches = 4

No. of nodes = 8

Test complexity = (1\*2)\*(1\*2)\*(1\*2)\*(1\*2) = 16

* Test complexity for input model:

No. of branches = 2

No. of nodes = 7

Test complexity = (1\*5)\*(1\*2) = 10

* Test complexity for output model:

No. of branches = 2

No. of nodes = 4

Test complexity = (1\*2)\*(1\*2) = 4

1. 3D model-based test complexity = 8\*7 = 56

**3.c)** Three of the selected bugs detected are as follows:

1.

|  |  |
| --- | --- |
| **Test Case ID** | **1** |
| Test Topic | Algebra (Polynomial Equations) |
| Test Description | Word Problem |
| AI Context Type | Non-Maths Equation |
| AI Input Type | Single variable |
| Test Case Input | If John has 25 cars, then what will be the new total number of cars if he sells out one from it? |
| Performed By | Harish Marepalli |
| Execution Date | 20th November 2023 |
| AI Output Type | Value-based |
| Expected Result | 24 |
| Actual Result | Error (Unable to read the word problem) |
| Test Case Result | Fail |

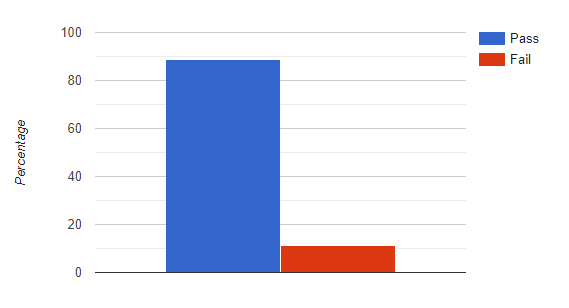
2.

|  |  |
| --- | --- |
| **Test Case ID** | **2** |
| Test Topic | Algebra (Polynomial Equations) |
| Test Description | Word Problem |
| AI Context Type | English |
| AI Input Type | Single variable |
| Test Case Input | If the value of x2 is 25, what is the value of x2 - 1? |
| Performed By | Harish Marepalli |
| Execution Date | 20th November 2023 |
| AI Output Type | Value-based |
| Expected Result | x = 24 |
| Actual Result | Error (Unable to read the word problem) |
| Test Case Result | Fail |

3.

|  |  |
| --- | --- |
| **Test Case ID** | **3** |
| Test Topic | Algebra (Polynomial Equations) |
| Test Description | Word problem |
| AI Context Type | Non-English |
| AI Input Type | Multi variable |
| Test Case Input | x విలువ 25 మరియు y 10 అయితే, xy విలువ ఎంత? |
| Performed By | Harish Marepalli |
| Execution Date | 20th November 2023 |
| AI Output Type | Value-based |
| Expected Result | Cannot recognize the problem |
| Actual Result |  |
| Test Case Result | Fail |

AI Function Bug Analysis:



With the AI tool generated test cases, we can see above that the pass percentage is good but needs a little improvement by getting rid of the bugs.