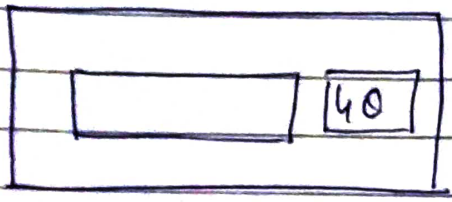
8/4/2021

Software Testing

Assignment - I

An oracle is a mechanism for determining whether the program has passed or failed test case.

Go button



For Go we can have following test cases -

-> ~~Test~~ Input field is empty.

-> item which is present in database
-> item which is not present in database
-> ~~Foo~~ Input is given as digit.
-> Alphanumeric I/p

-> The generator would provide predicted or expected result for each test case & i.e. if item is present then display result otherwise display not found
-> Comparator will compare predicted & ~~output~~ obtained results.

-> Evaluator will determine whether the comparision result are sufficiently close. If they are close for all the above cases then it will be determined that Go button is working fine.

When the search result are displayed previous & next buttons will be checked.

By cases –

- If you are on first page then previous button should be disabled as there is no previous page.
- If you are on some other page so by clicking on previous button it should go to previous page.

→ If you are on last page next button should be disabled otherwise should go to next page.

Oracle will check for all the cases & if the expected output is almost same as predicted output then we can infer that previous & next are implemented correctly.

→ Any of the oracle capabilities may be automated. We might generate predictions for test case from previous test result on this program, from the previous release of this program or competitor's program or from standard function or custom model.

Q2 When we test individual component we only focus on that particular component without considering the other components. When these individually tested components are combined errors might be generated. So to make sure that these components are working properly when they are combined together, integration testing is required. as -

→ Each module is designed by individual software developer whose programming logic may be different from developers of other modules, so Software testing is necessary to make sure that these modules are working properly.

→ If the two modules are incompatible, it will generate errors.

→ Hardware's compatibility must be checked with software.

→ If exception handling is inadequate between the modules, it can create bugs.

→ Requirements can be changed or enhanced at the time of module development. These new requirements may not be tested at the level of unit testing hence integration testing is mandatory.

Integration strategies :-

→ Incremental Integration Testing
  → Top down Integration Testing
  → Bottom up integration testing

→ Non Incremental Integration testing.

Incremental Approach -

- There is strong relationship between dependent modules.
- We take 2 or more modules & verify data flow between them. If it is working fine, then we add more modules.

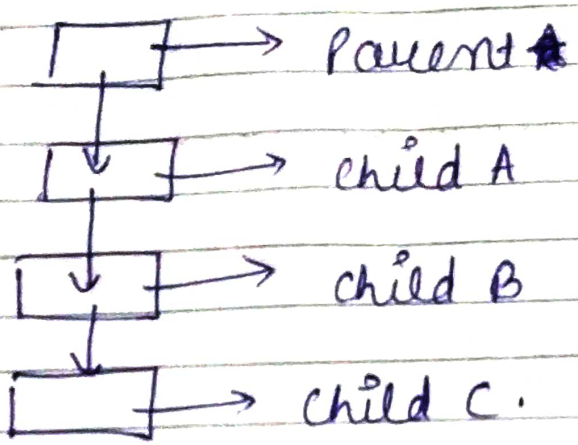
Example - suppose we have online food ordering app like Zomato. We will perform.
Incremental testing ~~method~~ & flow of application would be :
Zomato → Log in → Search Restaurant → Add to cart → Payment → Log out.

Incremental Testing methods -

(1) Top down approach - In this, higher level modules are tested with lower level modules until successful completion of testing of all the modules.
Modules are added one by one & check data flow in the same order.

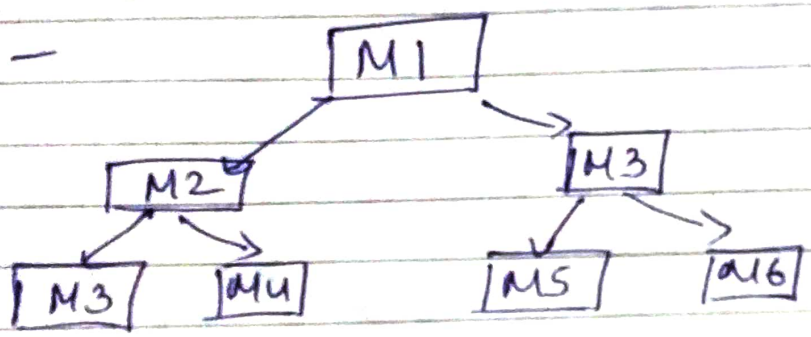
→ start from top most module & gradually progress towards lower modules.

→



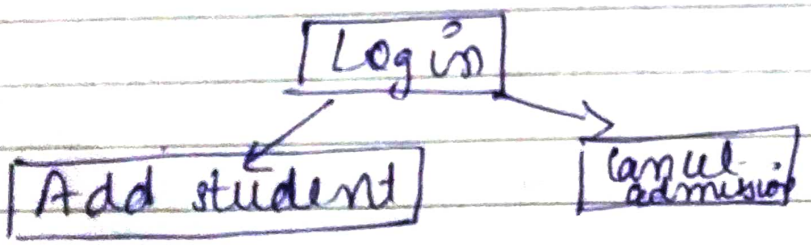
→ Module which is being added should be child of previous one like B is child of A & so on.

EX –



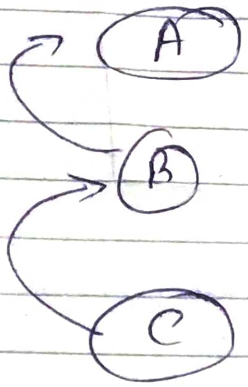
Testing starts from M1. then we will integrate M2, then M3, M4, M5, M6.

EX- Suppose some application has modules Login, Add student, cancel admission.

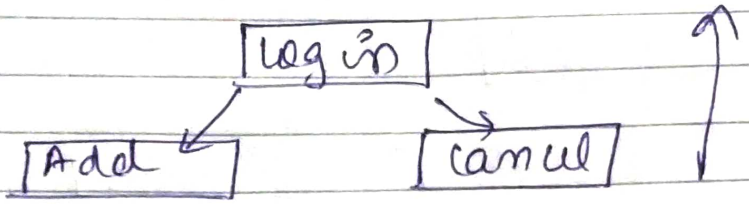


Testing starts from login, then Add student, then cancel admission.

Bottom up approach - lower level modules are tested with higher level modules. until successful completion of testing of all the modules.



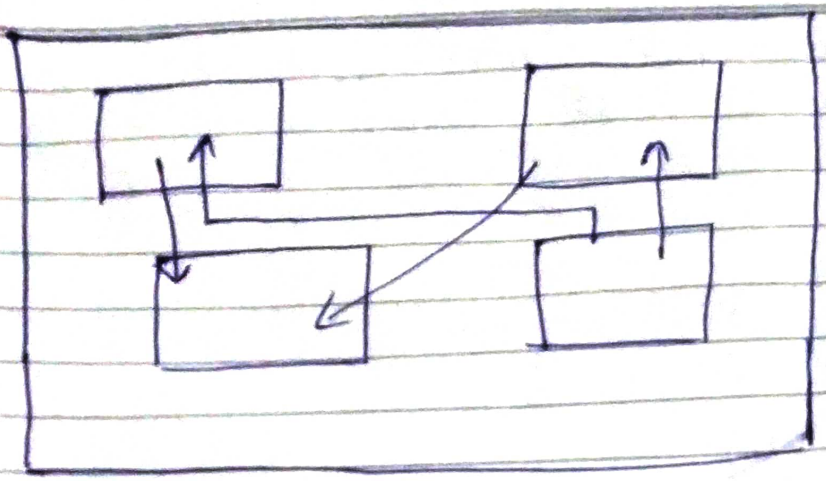
Example - for same example.



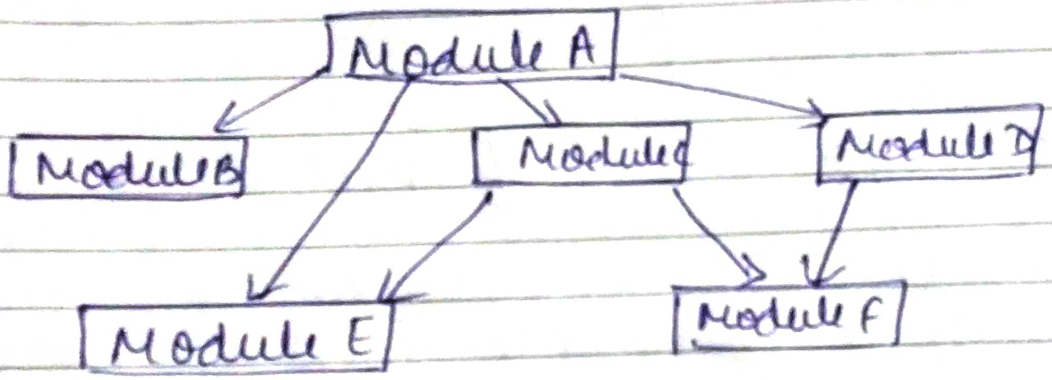
Testing starts from add & cancel & then the topmost module login is tested at last.

Non Incremental integration Testing

→ when data flow is very complex & when it is difficult to find who is parent & who is child.
→ We create data in any module b ang on all other existing modules & check if data is present.
Hence it is known as Big Bang method.

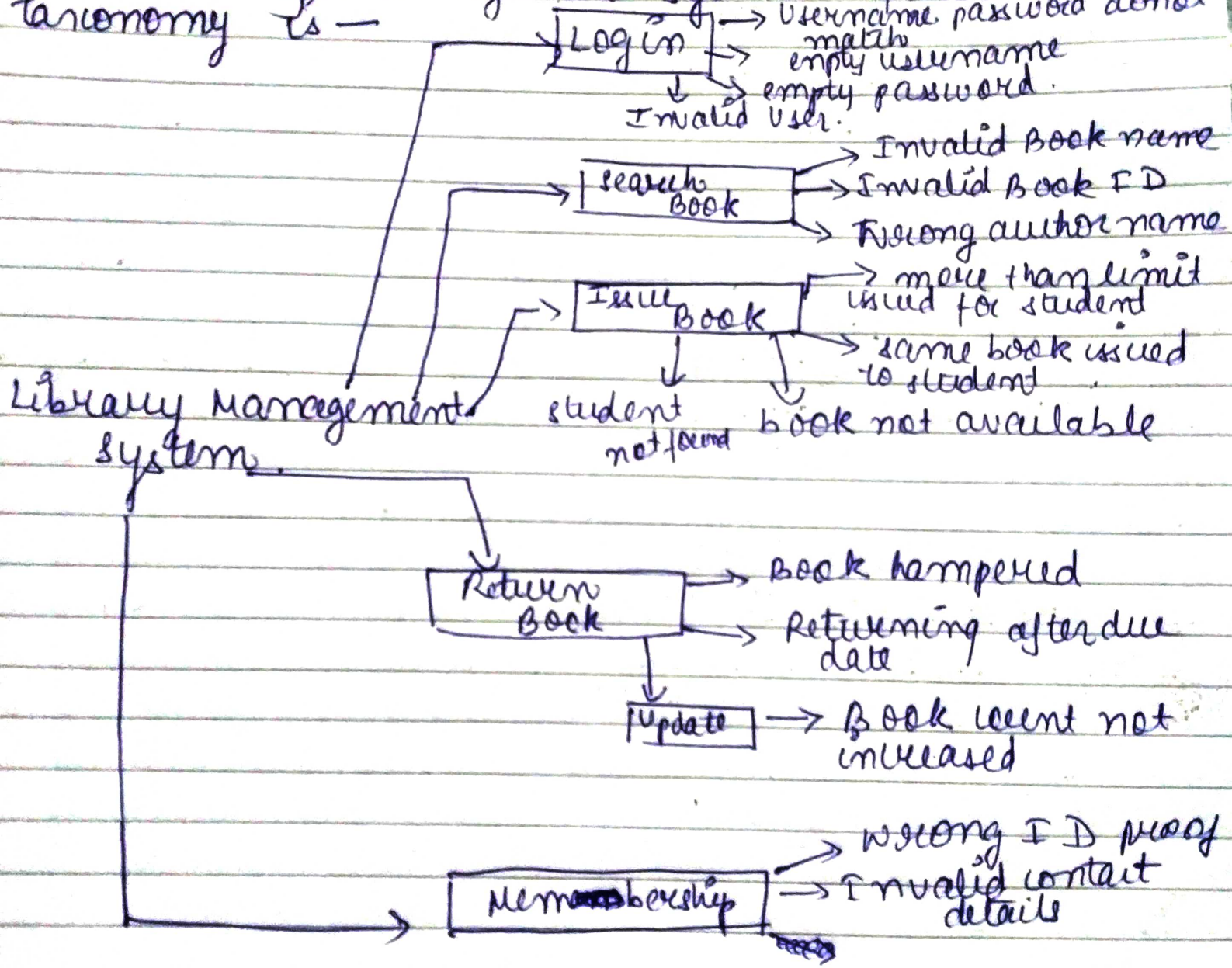


* All software components are combined at once & make compliated systems.
→ Integration testing will not be executed until all components are completed.



Q3 Taxonomy is classification of things into ordered groups or categories that indicate natural / hierarchial relationship.

For library management system fault taxonomy is —



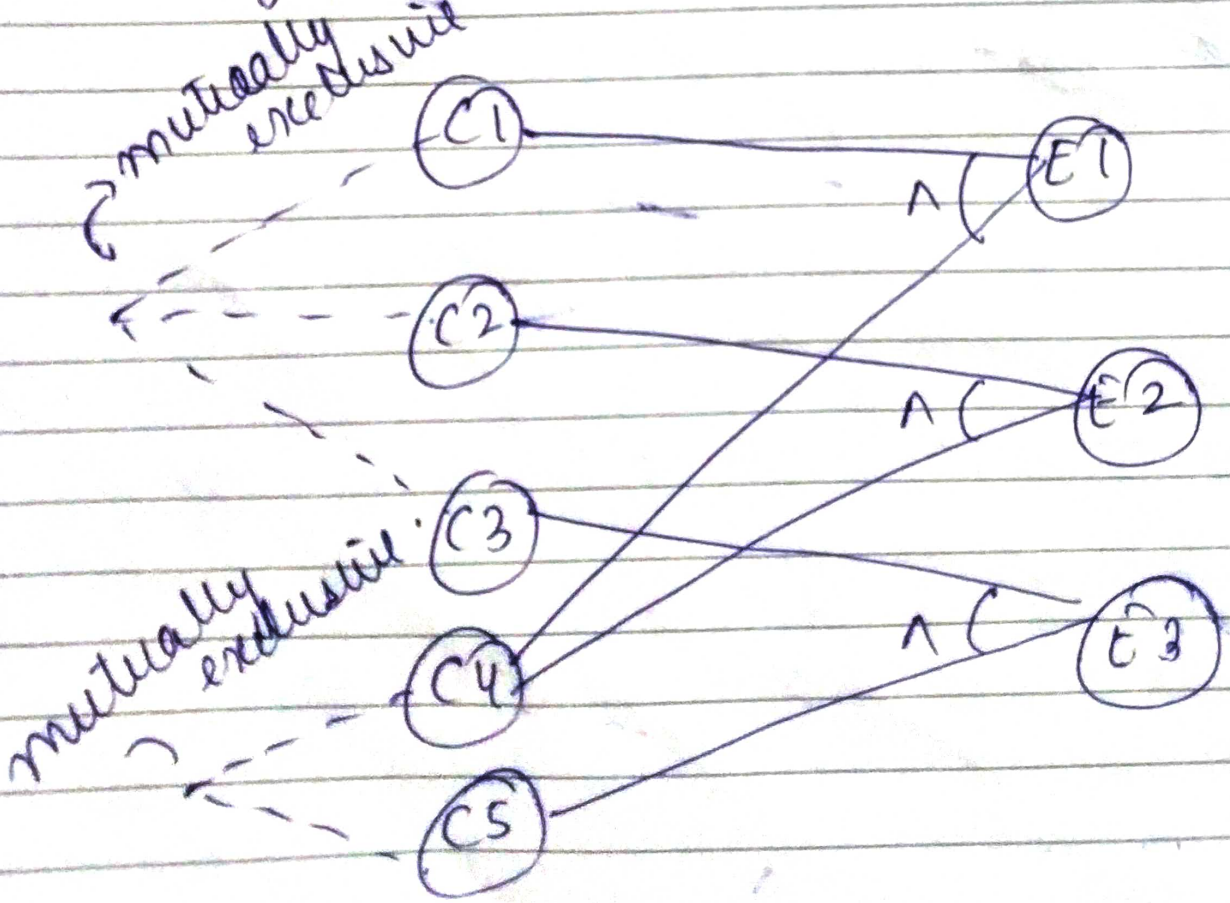
Q4 Causes allocated by C are as follows -

C1 : a is 100 meters
C2 : a is 120 meters
C3 : a is 80 meters
C4 : b is 10 mm.
C5 : b is 20 mm.

Effects allocated by E are as follows -

E1 : Level is safe.
E2 : Level is too high
E3 : Level is too low.

Cause Effect Graph -



For E1 to be true, C1 & C4 must be true
For E2 to be true, C2 & C4 must be true
For E3 to be true, C3 & C5 must be true.

Test Case	1	2	3
causes:			
C1: a is 100 metres	1	0	0
C2: a is 120 metres	0	1	0
C3: a is 80 meters	0	0	1
C4: b is 10 mm	1	1	0
C5 b is 20 mm.	0	0	1
Effects:			
E1: level is safe	1	0	0
E2: level is too high	0	1	0
E3: level is too low	0	0	1

Test Case #	Input (causes)		Expected output. (Effects)
	a	b	
1	100	10	level is safe
2	120	10	level is too high
3	80	20	level is too low.