Efficient Test Suite Management cont…

Dr. Durga Prasad Mohapatra
Professor
Dept. of CSE, NIT Rourkela

Data Flow Based Test Case Prioritization

- This is a white box testing technique used to detect improper use of data values due to coding errors.
- Data usage for a variable affects the white box testing and thereby the regression testing.
- If the prioritization of regression test suite is based on this concept, the rate of detection of faults will be high and critical bugs can be detected earlier.

Data Flow Based Test Case Prioritization cont…

J.Rummel et al. proposed an approach to regression test prioritization that leverages the all-du's (definition-use) test adequacy criterion that focuses on the definition and use of variables within the program under test.

Data Flow Based Test Case Prioritization cont…

- Kumar et al. proposed an approach for test case prioritization using *du* path as well as definition clear (dc) paths.

  Idea: *du* paths which may not be *dc*, may be very problematic as non-dc paths may be subtle source of errors.

Data Flow Based Test Case Prioritization cont…

- In another work, it was indicated that for testing modified programs, the prioritized test suite for original program may not work because after modification, new *du* paths may get introduced and some of these *du* paths may also not be definition clear (dc).
- These paths may also be more prone to errors.

Data Flow Based Test Case Prioritization cont…

- So, a list of all such newly introduced non-dc paths is prepared and test cases corresponding to these paths at the highest priority are placed in the test suite for a modified program. This set of test cases is referred to as ‘Set-1’.
- Because of modification in the program, some existing *dc* paths may become *non-dc*. The set of test cases corresponding to these paths is taken at the next priority and this set of test cases may be referred as ‘Set-2’.

Data Flow Based Test Case Prioritization cont…

- Further, there may be some *non-dc* paths of the original program, which are still *non-dc* after modification of the program. The set of test cases for such paths may be referred to as 'Set-3'.
- Finally, there are some test cases which do not cover non-dc paths, i.e., these test cases cover dc-paths. This may be referred to as 'Set-4'.
- Using these 4 concepts (sets), an algorithm has been designed for prioritizing the test suite.

Algorithm for prioritizing the test suite

Step 1:

If a test case covers more number of paths in Set-1

Then place it at the highest priority.

Step 2:

If 2 test cases (in Step1)cover an equal number of paths in Set-1

Then the test case which covers more number of paths in Set-2 in the modified program is placed at the next priority.

Step 3:

If 2 test cases (in Step 2) cover equal number of paths in Set-2 in the modified program,

Then the test case which covers maximum number of lines of code in the modified program is placed at the next priority.

Module Coupling Slice-Based Test Case Prioritization

- This technique is based on dependence and coupling between the modules, and proceeds in 2 steps.

Steps for Module Coupling Slice-Based Test Case Prioritization

Step 1: Find the highly affected module whenever there is a change in a module.

Step 2: Prioritize the test cases of the affected module identified in Step 1.

- In both steps, coupling information between modules is considered.

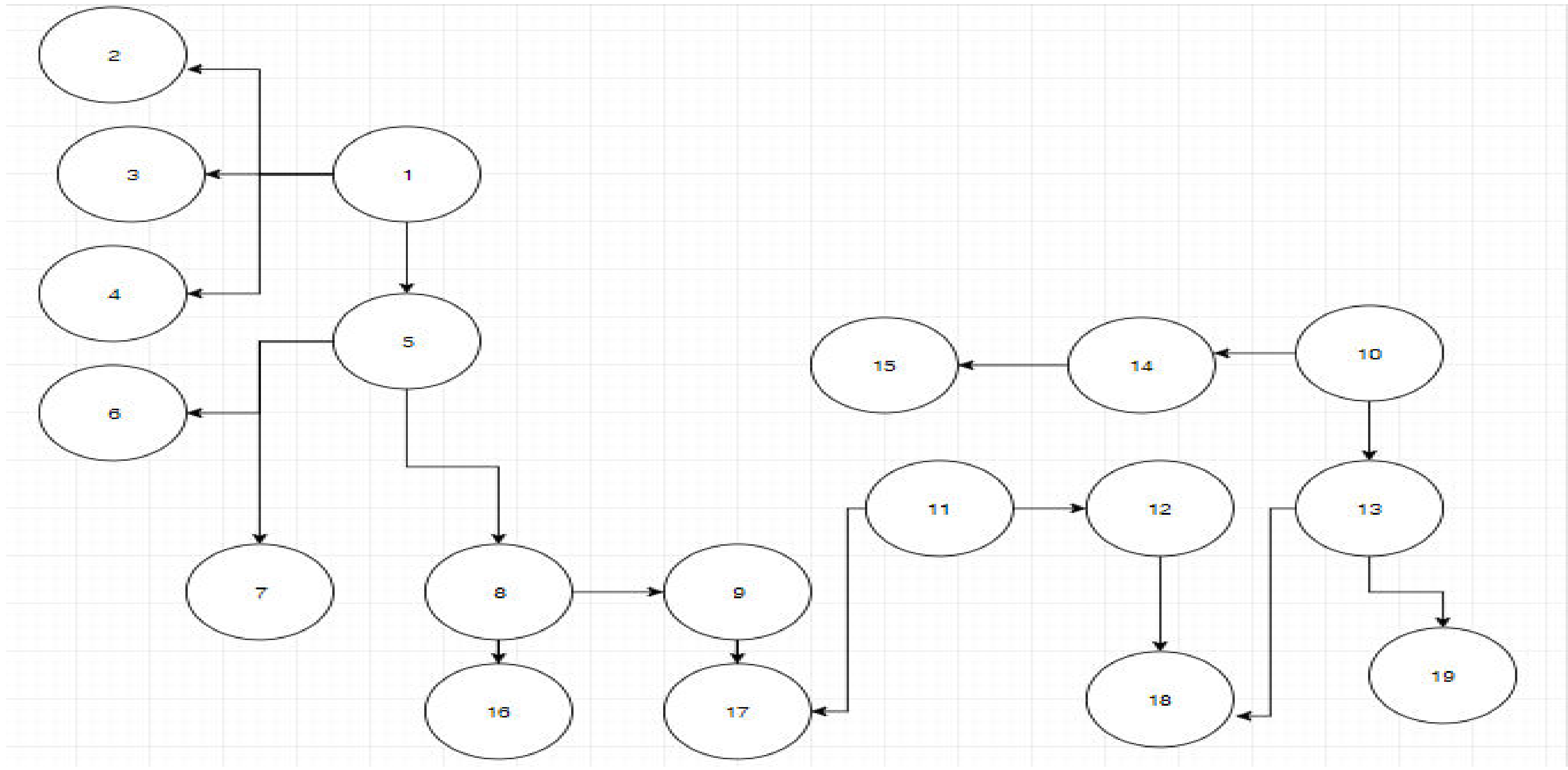
Module Coupling Slice-Based Test Case Prioritization cont…

- When there is a change in a module, there will be some effect on other modules which are coupled with this module.
- Based on the coupling information between the modules, the highly affected module can be identified.

Module-Coupling Effect

- The effect is worse if there is high coupling between the modules causing high probability of errors. This is called as *Module-Coupling Effect*.
- If regression test case prioritization is done based on this module coupling effect, there will be high percentage of detecting **critical errors** that have been propagated to other modules due to any change in a module.

Example Call Graph



Module-Coupling Effect cont…

- Modules 17 and 18 are being called by multiple modules.
- If there is any change in these modules 17 & 18, then modules 9,11and 12,13 will be affected respectively.
- If there is no prioritization, then according to regression testing, all test cases of all the affected modules will be executed, thus increasing the testing time and effort.
- If the coupling type between modules is known, then a prioritization scheme can be developed based on this coupling information.

Module Coupling Slice-Based Test Case Prioritization cont…

- Modules having worst type of coupling will be prioritized over other modules and their test cases.
- Module dependence can be identified by coupling and cohesion.
- A quantitative measure of the dependence of modules will be useful to find out the stability of the design.
- The work is based on the premise that different values can be assigned for various types of module coupling and cohesion.

Module Coupling Slice-Based Test Case Prioritization cont…

Table 1: Coupling types and their values

Coupling Types	Value
Content	0.95
Common	0.70
External	0.60
Control	0.50
Stamp	0.35
Data	0.20

Module Coupling Slice-Based Test Case Prioritization cont…

Table 2: Cohesion types and their values

Cohesion Types	Value
Coincidental	0.95
Logical	0.40
Temporal	0.60
Procedural	0.40
Communicational	0.25
Sequential	0.20
Functional	0.20

Module Coupling Slice-Based Test Case Prioritization cont…

- A matrix can be obtained by using the 2 tables, which gives the dependence among all the modules in a program.
- This dependence matrix describes the probability of having to change module i, given that module j has been changed.
- Module Dependence Matrix is derived using the following 3 steps:

Steps for Module Dependence Matrix

Step 1:

(i) Determine the coupling among all the modules in the program.

(ii) Construct an mxm coupling matrix, where m is the number of modules. Using Table 1, fill the elements of matrix. Element $\mathbf{C_{ij}}$ represents coupling between modules i and j.

(iii) This matrix is symmetric, i.e. $\mathbf{C_{ij} = C_{ji}}$ for all i and j.

(iv) Elements on the diagonal are all 1 (C_{ij} = 1for all i).

Steps for Module Dependence Matrix cont…

- **Step 2:**

(i) Determine strength of each module in the program.

(ii) Using Table 2, record the corresponding numerical values of cohesion in module cohesion matrix (**S**).

Steps for Module Dependence Matrix cont…

- **Step 3:**

(i) Construct the Module Dependence Matrix **D** using the following equation:

D_{ij}=0.15 (S_i + S_j)+ 0.7 C_{ij} , where C_{ij} is not equal to 0,

= 0, where C_{ij} = 0 D_{ii} = 1 for all i.

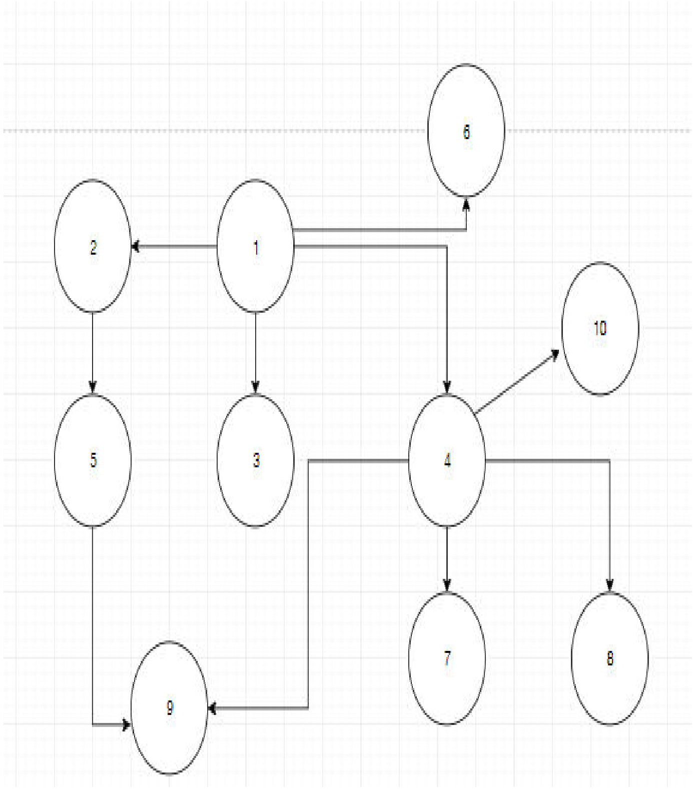
Prioritization of modules can de done by comparing non zero entries of matrix **D**.

Steps for Module Dependence Matrix cont…

- For example, if module i has been modified, find all the existing parent modules(j,k,l,…) of that changed module(i) and after that compare first-order dependence matrix entries for particular links, like i-j, i-k,i-l and so on.
- The link having the highest module dependence matrix value will get the highest priority and the link with the lowest module dependence matrix value will get the lowest priority.

Example

A Software consists of 10 modules. The coupling and cohesion information of these modules are given in tables shown in next slide. Let us find out the badly affected module in this software.



Call Graph

Example cont…

Coupling information

Type of Coupling	No. of Modules in Relation	Examples
Data Coupling	3	1-2,1-4,1-6
Stamp Coupling	1	1-3
Control Coupling	4	4-7,4-8, 4-9,4-10
Common Coupling	2	2-5,5-9
Message (Content) Coupling	0	-

Cohesion information

Module Number	Cohesion Type
1	Coincidental
2	Functional
3	Communicational
4	Logical
5	Procedural
6	Functional
7	Functional
8	Functional
9	Functional
10	Functional

Example cont…

By using coupling values among different modules, a module coupling matrix is given below:

1.0	0.2	0.35	0.2	0.0	0.2	0.0	0.0	0.0	0.0
0.2	1.0	0.0	0.0	0.70	0.0	0.0	0.0	0.0	0.0
0.35	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.2	0.0	0.0	1.0	0.0	0.0	0.50	0.50	0.50	0.50
0.0	0.2	0.0	0.0	1.0	0.0	0.0	0.0	0.70	0.0
0.2	0.2	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
0.0	0.0	0.2	0.50	0.0	0.0	1.0	0.0	0.0	0.0
0.0	0.0	0.0	0.50	0.0	0.0	0.0	1.0	0.0	0.0
0.0	0.0	0.0	0.50	0.70	0.0	0.0	0.0	1.0	0.0
0.0	0.0	0.0	0.50	0.0	0.0	0.0	0.0	0.0	1.0

Example cont…

By using the value of cohesion among different modules a Cohesion Matrix (**S**) is designed as shown below:

0.95	0.2	0.25	0.4	0.4	0.2	0.2	0.2	0.2	0.2

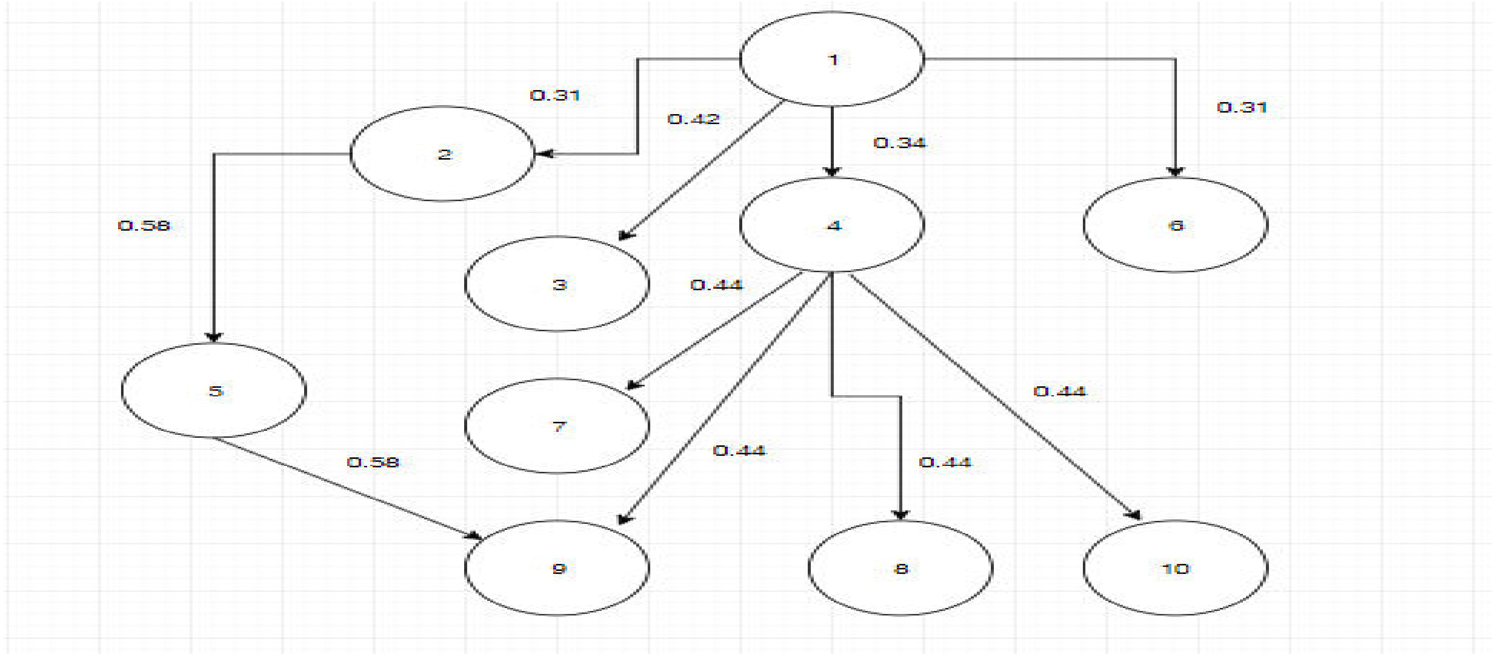
Example cont…

A Module Dependence Matrix (**D**)is designed, (using the previous equation).

1.0	0.31	0.42	0.34	0.0	0.31	0.0	0.0	0.0	0.0
0.31	1.0	0.0	0.0	0.58	0.0	0.0	0.0	0.0	0.0
0.42	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.34	0.0	0.0	1.0	0.0	0.0	0.44	0.44	0.44	0.44
0.0	0.58	0.0	0.0	1.0	0.0	0.0	0.0	0.58	0.0
0.31	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.44	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.44	0.0	0.0	0.0	1.0	0.0	0.0
0.0	0.0	0.0	0.44	0.58	0.0	0.0	0.0	1.0	0.0
0.0	0.0	0.0	0.44	0.0	0.0	0.0	0.0	0.0	1.0

Example cont…

From the module dependence values obtained from call graph as shown below and from Module Dependence matrix, we conclude that change in module 4 propagates to modules 7,8,9 and 10.



Example cont…

- Modules 7,8,9 and 10 have the same module dependence value(0.44), so the order of prioritization of test cases for these modules is the same.
- Similarly, the change in module 1 propagates to modules 2,3,4 and 6.
- The module dependence values for these modules show that module 3 is the more affected module when compared to modules 2,4 and 6.
- So, the test cases for module 3 have to be prioritized first as compared to modules 2,4 and 6.

Module Coupling Slice-Based Test Case Prioritization cont…

- After identifying the highly affected module due to a change in a module, there is a need to execute the test cases corresponding to this affected module.
- However, there may be a large number of test cases in this module.
- Therefore, there is a need to prioritize these test cases so that critical bugs can be found easily.
- Based on the coupling information between the changed module and the affected module, a **coupling slice** can be prepared that helps in prioritizing the test cases.

Module Coupling Slice-Based Test Case Prioritization cont…

- All the statement numbers present in the execution history of a program comprise the execution slice of a program.
- Coupling information can be helpful to decide which variables are affected in the caller module.
- Depending upon this information the statement numbers of affected variables can be found. This may be called a **coupling slice**.

Module Coupling Slice-Based Test Case Prioritization cont…

- At last, these statement numbers are matched in the execution slice.
- The test cases for these statements will be given high priority and executed first.

THANK YOU