Complexity Metrics. Stage of Planning

- Quality of software: output per unit of measure
- Output: Profits (losses) of the firm related to the project
- Unit of measure: Lines of code

Complexity Metrics. Stage of Planning

- Quality of software: output per unit of measure
- Output: Profits (losses) of the firm related to the project
- Unit of measure: Lines of code

Complexity Metrics. Stage of Planning

- Quality of software: output per unit of measure
- Output: Profits (losses) of the firm related to the project
- Unit of measure: Lines of code

Lines of code

- Easy to measure BUT
- Language dependent
- Penalized for a careful design
- Preliminary estimates of the quality is performed at an early stage when the length of the code is difficult to estimate
- More universal (abstract) function points can be used

Computing function points

			Weighting factor			
Measurement parameter	**Count**		**Simple**	**Average**	**Complex**	
Number of user inputs		×	3	4	6	=
Number of user outputs		×	4	5	7	=
Number of user inquiries		×	3	4	6	=
Number of files		×	7	10	15	=
Number of external interfaces		×	5	7	10	=
Count total	→					

- **Number of user inputs**. Each user input that provides distinct application-oriented data to the software is counted. Inputs should be distinguished from inquiries, which are counted separately.
- **Number of user outputs**. Each user output that provides application-oriented information to the user is counted. In this context output refers to reports, screens, error messages, etc. Individual data items within a report are not counted separately.
- **Number of user inquiries**. An inquiry is defined as an on-line input that results in the generation of some immediate software response in the form of an on-line output. Each distinct inquiry is counted
- **Number of files**. Each logical master file (i.e., a logical grouping of data that may be one part of a large database or a separate file) is counted.
- **Number of external interfaces**. All machine readable interfaces (e.g., data files on storage media) that are used to transmit information to another system are counted.

$$\text{FP} = \text{count total} \times \left(0.65 + 0.01 \times \sum(F_j)\right)$$

where F_j, $j = 1, \ldots, 14$ are number from 0 to 5 obtained as the answers to the following 14 questions:

1. Does the system require reliable backup and recovery?
2. Are data communications required?
3. Are there distributed processing functions?
4. Is performance critical?
5. Will the system run in an existing, heavily utilized operational environment?
6. Does the system require on-line data entry?
7. Does the on-line data entry require the input transaction to be built over multiple screens or operations?
8. Are the master files updated on-line?
9. Are the inputs, outputs, files, or inquiries complex?
10. Is the internal processing complex?
11. Is the code designed to be reusable?
12. Are conversion and installation included in the design?
13. Is the system designed for multiple installations in different organizations?
14. Is the application designed to facilitate change and ease of use by the user?

Example of computation

Measurement Parameter	Count	Weighting Factor
Number of user inputs	8	Simple
Number of user outputs	2	Simple
Number of user inquires	91	Average
Number of files	120	Simple
Number of external interfaces	2	Complex

The sum of complexity adjustment values is 8. Find FP.

Example of computation

Measurement Parameter	Count	Weighting Factor
Number of user inputs	8	Simple, 3
Number of user outputs	2	Simple, 4
Number of user inquires	91	Average, 4
Number of files	120	Simple, 7
Number of external interfaces	2	Complex, 10

The sum of complexity adjustment values is 8. Find FP.

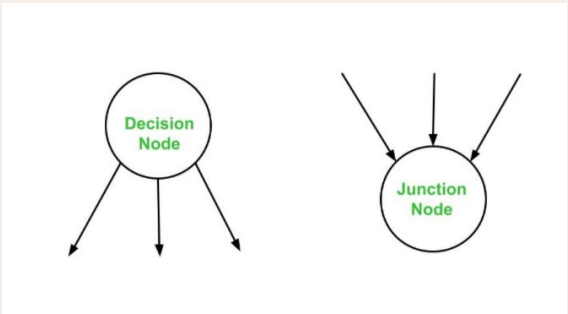
Aposterior Estimates: Codes and Classes

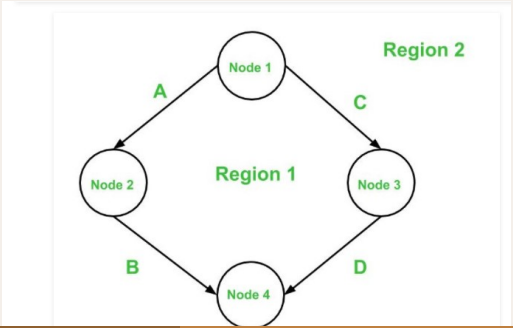
Control Flow Graph

A control flow graph (or simply, flow graph) is a directed graph which represents the control structure of a program or module. A control flow graph (V, E) has V number of nodes/vertices and E number of edges in it. A control graph can also have :

- **Junction Node** is a node with more than one arrow entering it.
- **Decision Node** is a node with more then one arrow leaving it.
- **Region** is the area bounded by edges and nodes (area outside the graph is also counted as a region.)

Junction and decision nodes: example





Cyclomatic Complexity

Definition

$$V = E - N + 2P,$$

where

- E is the number of edges
- N is the number of vertices
- P is the number of the connected components

Alternative

$$V = \text{Number of Regions},$$

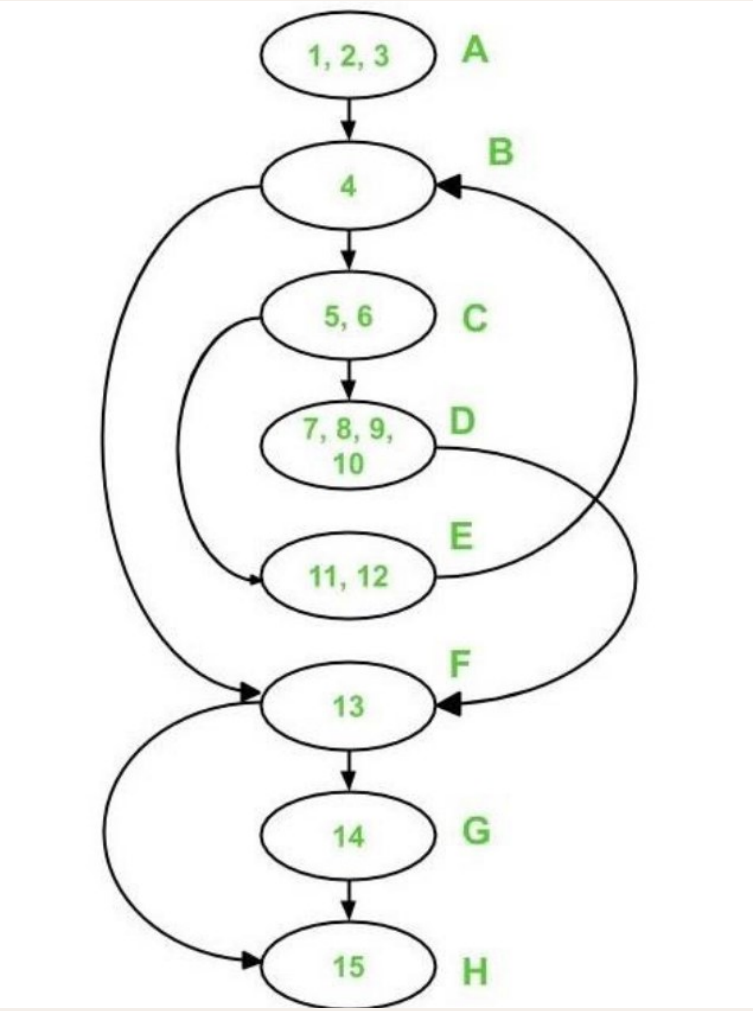
Example of computation: Code

```
int main (){
    int n, index;
    cout << "Enter a number: " <> n;
    index = 2;
    while (index <= n - 1){
      if (n % index == 0){
        cout << "It is not a prime number" << endl;
        break;
     }
     index++;
    }
     if (index == n)
       cout << "It is a prime number" << endl;
} // end main
```

Example: Computation

1. After declaring the variables, begin numbering the statements (if no variables have been initialized in that statement). If a variable is initialized and declared on the same line, however, the numbering should begin on that line. It is done.
2. Combine all of the sequential statements into one node. Statements 1, 2, and 3 are, for example, all consecutive statements that should be concatenated into a single node. For the rest of the statements, we'll use the notations described here. **Note.** To keep things simple, number nodes in alphabetical order.

Cyclomatic Complexity



Example: Computation

Recall

$$V = E - N + 2P,$$

$E = 10,\ N = 8,\ P = 1$

$$V = 10 - 8 + 2 * 1 = 4.$$

Example: Computation

Recall

$$V = E - N + 2P,$$

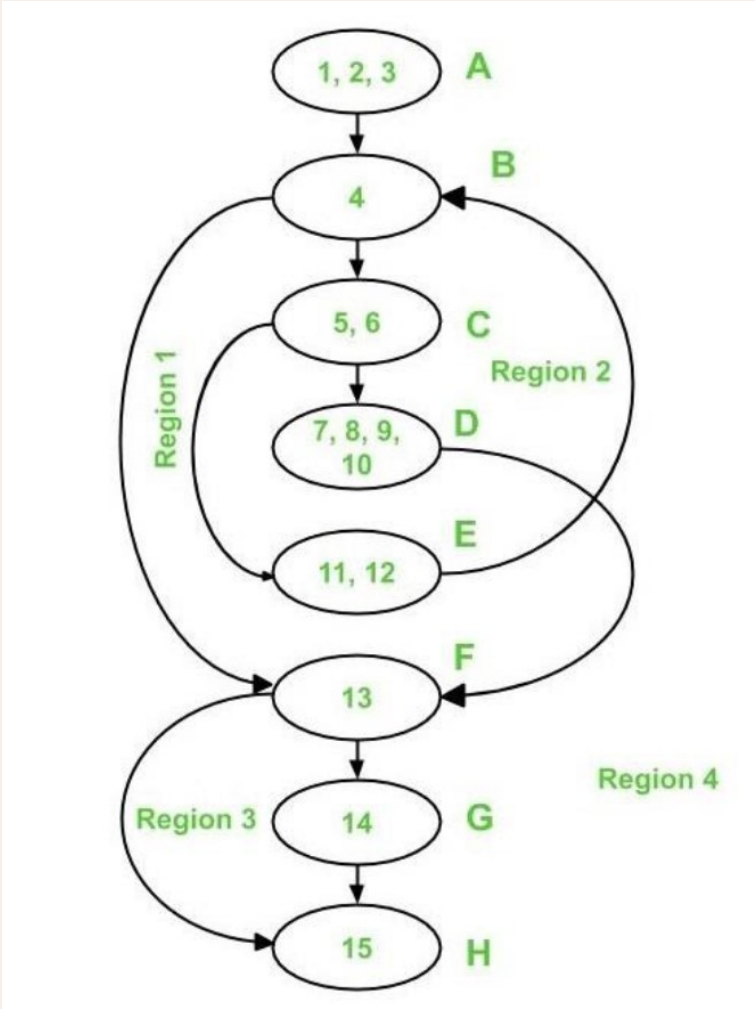
- This translates to the number of decisions + 1.
- Binary decisions such as `if` and `while` statements add 1 to complexity.
- Boolean operators can add either one or nothing to complexity. For instance, one may be added if a Boolean operator is found within a conditional statement.

Example: Computation

Recall: Alternative

$$V = \text{Number of Regions},$$

Cyclomatic Complexity



Example: Computation

Recall: Alternative

$$V = \text{Number of Regions} = 4$$

Cyclomatic Complexity: Excercise

Find cyclomatic complexity

```
void foo(void){
    if (a && b)
        x=1;
    else
        x=2;
}
```

Cyclomatic Complexity: Excercise

Find cyclomatic complexity

```
void foo(void){
    if (a && b)
        x=1;
    else
        x=2;
}
```

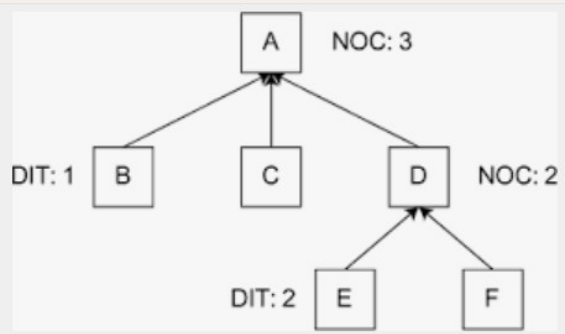
Answer: 3

From Separate Classes to the Whole Software

Weighted Methods Per Class

- The sum of the complexity values for all methods of a given class
- The higher the value, the more effort required
- WMC should be kept low

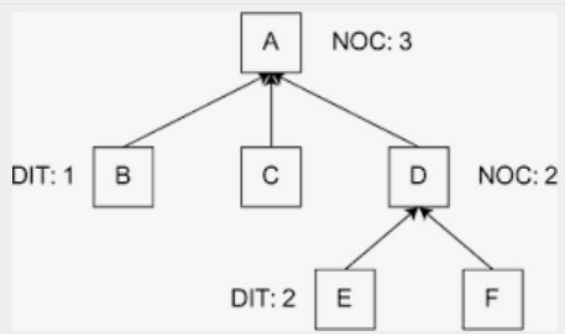
Depth of Inheritance Tree (DIT)



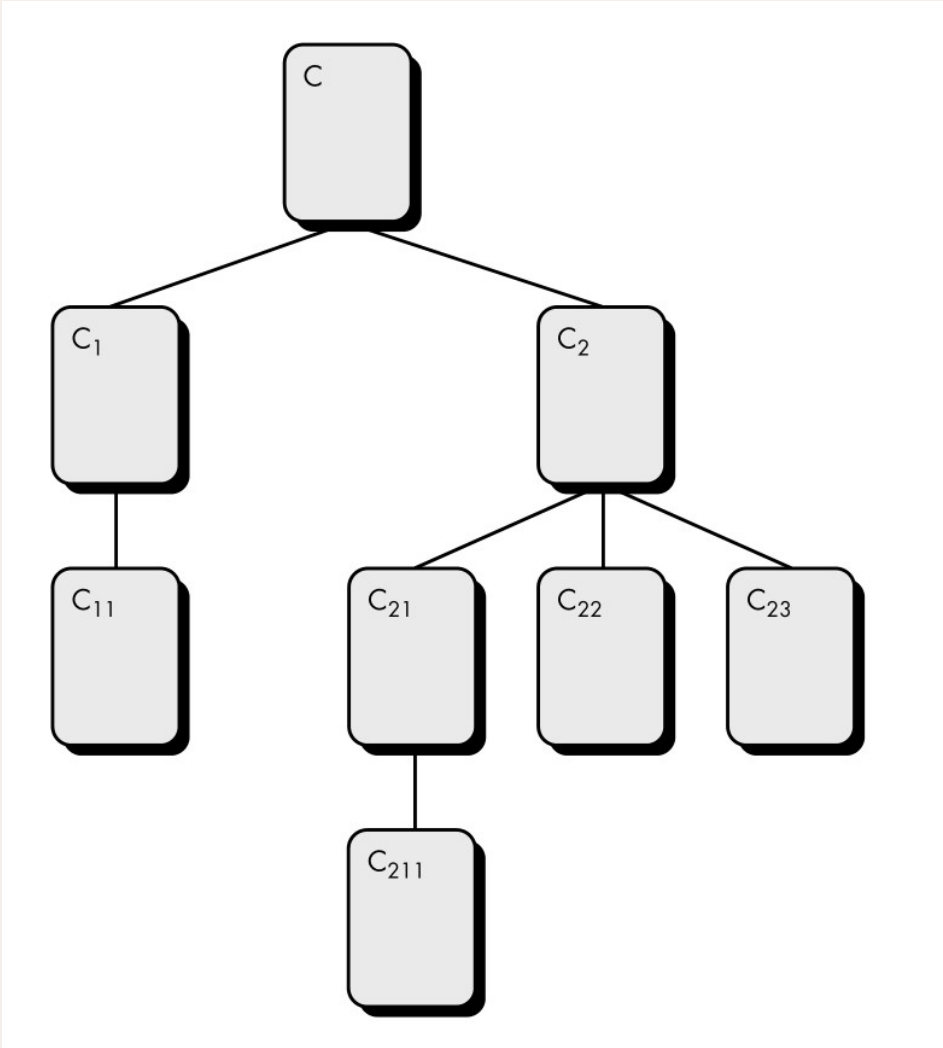
DIT metric is the length of the maximum path from the node to the root of the tree. So this metric calculates how far down a class is declared in the inheritance hierarchy. The following figure shows the value of DIT for a simple class hierarchy. DIT represents the complexity of the behaviour of a class, the complexity of design of a class and potential reuse.

Number of Children (NOC)

This metric measures how many sub-classes are going to inherit the methods of the parent class

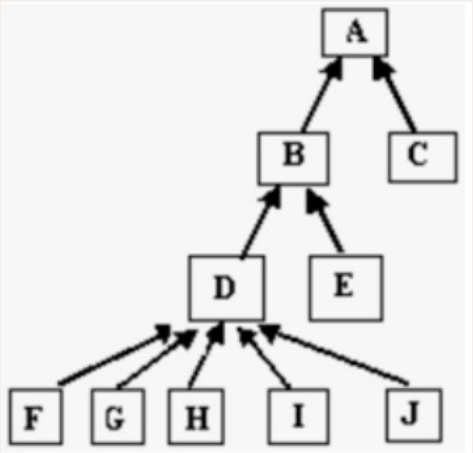


DIT & NOC: Excersice



What are DIT and NOC here?

DIT & NOC: Another Excersice



What are DIT and NOC here?

Response for a class

- The number of methods that can potentially be executed in response to a message received by an object of a given class
- As the RFC value increases, testing effort and design complexity increase
- RFC should be kept low

Lack of cohesion in methods (LCOM)

- The number of methods in a given class that access one or more of the same instance variables
- The higher the LCOM value, the lower the cohesion of methods, and greater the coupling
- A high LCOM value could indicate the need to break the class apart into multiple classes

Class Size (CS)

- The total number of methods plus the total number of attributes encapsulated by a given class
- Inherited members should be weighted more heavily than local members
- Large values of CS could indicate that the class is too large; that is it encapsulates too much behavior, structure, and responsibility
- High CS values may also indicate lower reusability

Number of operations overridden by a subclass (NOO)

- A count of the methods in subclasses that have been redefined
- Large NOO values could indicate a design problem, since the model of the class seems to be violated

Number of operations added by a subclass (NOA)

- A count of the new methods appearing in subclasses
- A large NOA value could indicate a design abstraction violation

Specialization index (SI)

The specialization index provides a rough indication of the degree of specialization for each of the subclasses in an OO system. Specialization can be achieved by adding or deleting operations or by overriding

$$SI = \frac{NOO \times level}{M_{\text{total}}}$$

where *level* is the level in the class hierarchy at which the class resides and M_{total} is he total number of methods for the class. The higher is the value of SI, the more likely the class hierarchy has classes that do not conform to the superclass abstraction.

Method inheritance factor (MIF)

The degree to which the class architecture of an OO system makes use of inheritance for both methods (operations) and attributes is defined as

$$MIF = \frac{\sum_{i=1}^{TC} M_i(C_i)}{\sum_{i=1}^{TC} M_a(C_i)},$$

TC is the total number of classes in the architecture, C_i is a class within the architecture, and

$$M_a(C_i) = M_d(C_i) + M_i(C_i)$$

$M_a(C_i)$ is the number of methods that can be invoked in association with C_i,
$M_d(C_i)$ is the number of methods declared in the class C_i,
$M_i(C_i)$ is the number of methods inherited (and not overridden) in C_i.

Method inheritance factor (MIF)

- If the number of the methods declared in a class and inherited (and not overridden) is approximately the same (for each class), what is MIF?
- $\frac{\sum_{i=1}^{TC} M_i(C_i)}{\sum_{i=1}^{TC} M_a(C_i)} \approx \frac{\sum_{i=1}^{TC} M_i(C_i)}{\sum_{i=1}^{TC} M_i(C_i)+M_i(C_i)} = 0.5$
- if the number of local methods is approximately 10 times larger?

Method inheritance factor (MIF)

- If the number of the methods declared in a class and inherited (and not overridden) is approximately the same (for each class), what is MIF?
- $\frac{\sum_{i=1}^{TC} M_i(C_i)}{\sum_{i=1}^{TC} M_a(C_i)} \approx \frac{\sum_{i=1}^{TC} M_i(C_i)}{\sum_{i=1}^{TC} M_i(C_i)+M_i(C_i)} = 0.5$
- if the number of local methods is approximately 10 times larger?

Method inheritance factor (MIF)

- If the number of the methods declared in a class and inherited (and not overridden) is approximately the same (for each class), what is MIF?
- $\frac{\sum_{i=1}^{TC} M_i(C_i)}{\sum_{i=1}^{TC} M_a(C_i)} \approx \frac{\sum_{i=1}^{TC} M_i(C_i)}{\sum_{i=1}^{TC} M_i(C_i)+M_i(C_i)} = 0.5$
- if the number of local methods is approximately 10 times larger?

Coupling between classes

- The amount of collaboration and interaction between a given class and the other classes in the system
- As the CBO value increases, reusability decreases
- a high CBO indicates potential diffiulty in modifying the class and the Subsequent testing of the modifications
- CBO should be kept low

Coupling between classes

$$CF = \sum_i \sum_j \frac{\delta_{ij}}{TC^2 - TC},$$

$\delta_{ij} = 1$ if a relationship exists between the classes C_i and C_j; $\delta_{ij} = 0$ otherwise. $\delta_{ii} = 0$ by definition.

Comutational Complexity

- Time required for the code to run
- Polynomial time is affordable
- in contrast to the exponential time

Examples

- Sorting: the algorithm requires $O(n^2)$ operation; what can you say about its quality?
- Algorithms placed to the library has $O(n \log(n))$ computational time
- What is the idea of such algorithms?

Examples

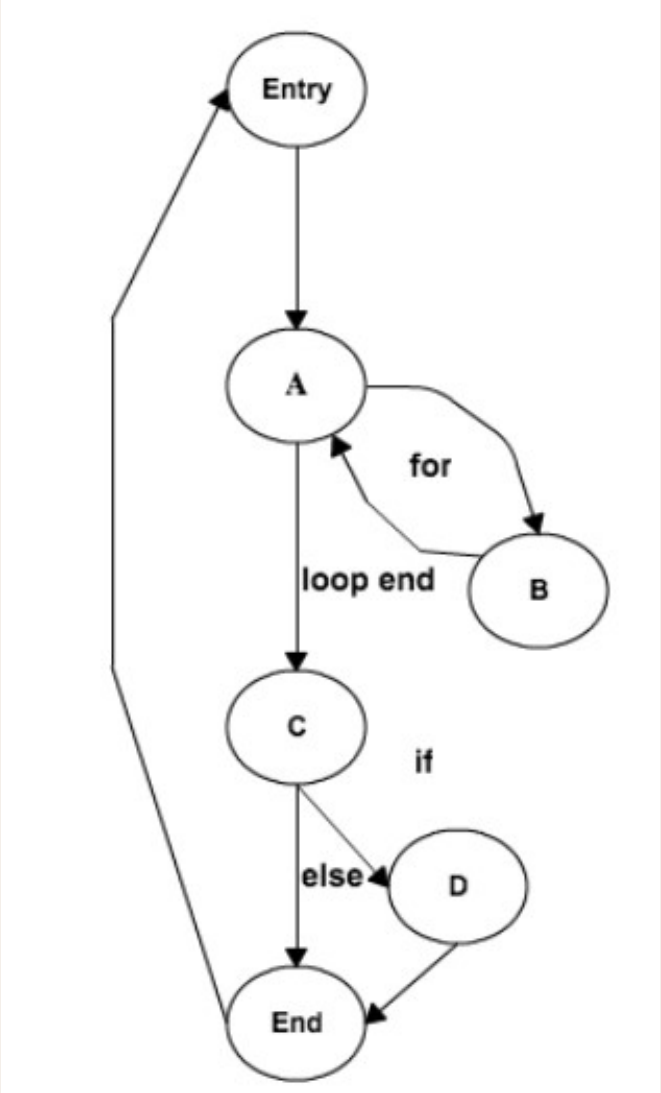
- Sorting: the algorithm requires $O(n^2)$ operation; what can you say about its quality?
- Algorithms placed to the library has $O(n \log(n))$ computational time
- What is the idea of such algorithms?

Examples

- Sorting: the algorithm requires $O(n^2)$ operation; what can you say about its quality?
- Algorithms placed to the library has $O(n \log(n))$ computational time
- What is the idea of such algorithms?

Excercise: find MCC

Control flow graph



Excercise: find Weighted Method Per Class

A class X has 20 operations. The complexity for operations 1 to 20 is 5, 4, 3, 3, 6, 8, 2, 2, 5, 5, 4, 4, 6, 8, 2, 2, 5, 5, 4, 4, respectively. Compute the weighted methods per class.