SIZE MEASURE

1.Code Size

Definition : Lines of code are the "source code" of the program, and one line may generate one machine instruction or several depending on the programming language. Almost every compiler shows total lines of code without specifying how many blank lines, comment lines, data declarations or headings.

[Executable Statement (ES) ignores comments, data declaration and heading. Delivered source instruction (DSI) includes data declarations and headings as source instructions but not comment and blank lines]

Tecniques:

- **Total size**, LOC = NCLOC + CLOC
  - Measure procedure: Compiler.
  - Language : All.

- **Density of program**, DoP = CLOC / LOC
  - Measure procedure: programatic.
  - Language : All.

- **Halstead Approach**. Volume of Program , $V = N \times \log 2\mu$

  $N = N1 + N2$ & $\mu 1 = \mu 2 + \mu 2$

  - Measure procedure: Manually.
  - Language : All.

- **Byte of code**
  - Measure procedure: Manually.
  - Language : All.

- **Number of characters**

  CHAR = α LOC

  - Measure procedure: Programatic. [UNIX & Linux operating systems have the command **<wc>** to compute it]
  - Language : APL, LISP, C, C++, Java.

Here,

NCLOC = Non-commented line of code

CLOC = Commented line of code

$\mu 1$ = Number of unique operators

$\mu 2$ = Number of unique operands

N1 = Total occurrences of operators

N2 = Total occurrences of operands

A = Average character in a line of code

2.Design Size

Definition: Design size measure size in terms of packages, design patterns, classes, interfaces, abstract classes, operations, and methods in a software project.

Tecniques

- **Packages:** Number of subpackages, number of classes, interfaces (Java), or abstract classes (C++).
    - Measure procedure: Manually.
    - Language : Java, c++

- **Design patterns:**

    - Number of different design patterns used in a design.
    - Number of design pattern realizations for each pattern type.
    - Number of classes, interfaces, or abstract classes that play roles in each pattern realization.

    - Measure procedure: Manually.
    - Language : OOP.

- **Classes, interfaces, or abstract classes:** Number of public methods or operations, number of attributes.
    - Measure procedure: Manually.
    - Language : Java, C++

- **Methods or operations:** Number of parameters, number of over loaded versions of a method or operation.
    - Measure procedure: Programatic.
    - Language : All.

- **Weighted methods per class (WMC) :** Summing the weights of the methods in a class, where weights are unspecified complexity factors for each method.
    - Measure procedure: Programatic.
    - Language : OOP.

3.REQUIREMENTS ANALYSIS AND SPECIFICATION SIZE

Definition: SRS Requirements and specification documents generally combine text, graphs, and special mathematical diagrams and symbols. These document can consist of a mixture of text and diagrams.

Tecniques:

- **Use case diagrams:** Number of use cases, actors, and relationships of various types.
  - Measure procedure: Manually.
  - Language : All.

- **Use case:** Number of scenarios, size of scenarios in terms of steps, or activity diagram model elements.
  - Measure procedure: Manually.
  - Language : All.

- **Domain model (expressed as a UML class diagram):** Number of classes, abstract classes, interfaces, roles, operatons, and attributes.
  - Measure procedure: Manually / programatic.
  - Language : All.

- **Data-flow diagrams used in structured analysis and design:** Processes (bubbles nodes), external entities (box nodes), data-stores (line nodes) and data-flows (arcs).
  - Measure procedure: Manually / Programatic.
  - Language : All.

- **Algebraic specifications:** Sorts, functions, operations, and axioms .
  - Measure procedure: Programatic.
  - Language : All.

- **Z specifications:** The various lines appearing in the specification, which form part of either a type declaration or a (nonconjunctive) predicate.
  - Measure procedure: Manually.
  - Language : All.

STRUCTURE MEASURE

*Structural attributes: complexity, length, coupling, and cohesion

Definition: The control flow measures are usually modeled with directed graphs, where each node (or point) corresponds to a program statement or basic block (code that always executes sequentially), and each arc (or directed edge) indicates the flow of control from one statement or basic block to another. We call these directed graphs control flowgraphs or flowgraphs.

Techniques:

- The **in-degree** of a node is the number of arcs arriving at the node, and the **out-degree** is the number of arcs that leave the node.
  - Measure procedure: Programatic / Manually.
  - Requirement : Control Flowgraph.

Depth of expression

- Primes: if F is a prime ≠ P1, then $\alpha((F) = 1$.
- Sequence: $\alpha(F1; \ldots; Fn) = \max (\alpha(F1); \ldots; \alpha(Fn))$.
- Nesting: $\alpha(F(F1, \ldots, Fn)) = 1 + \max(\alpha(F1), \ldots, \alpha(Fn))$.
  - Measure procedure: Programatic / Manually.
  - Requirement : Control Flowgraph.

- **Cyclomatic Complexity Measure:** the number of linearly independent paths.

  $\nu(F) = e - n + 2$ or, $\nu(F) = 1 + d$

  $e\nu(F) = \nu(F) - m$

  Here, e = edges.

  n = nodes.

  d = predicate nodes.

  Ev(F) = essential complexity.

  M = number of sub-flowgraphs.
  - Measure procedure: Programatic / Manually.
  - Requirement : Control Flowgraph.

- **Test Effectiveness Ratio (TER)** : the extent to which the test cases satisfy a particular testing strategy.

TER = (Number of requirements executed at least once) / (Total number of test requirements for criterion)

   - Measure procedure: Programatic.
   - Language : C.

Morphology : The "shape" of the over all system structure when expressed pictorially.

- Size: Measured as number of nodes, number of edges, or a combination of these.
   - Measure procedure: Programatic.
   - Requirement : Graph / program module.

- Depth: Measured as the length of the longest path from the root node to a leaf node.
   - Measure procedure: Programatic.
   - Requirement : Graph / program module.

- Width: Measured as the maximum number of nodes at any one level.
   - Measure procedure: Manually.
   - Requirement : Graph / program module.

- Ratio: Ratio = edges / node.
   - Measure procedure: Programatic.
   - Requirement : Graph / program module.

- **Tree Impurity -** how far a given graph deviates from being a tree.

Property 1: m(G) = 0 if and only if G is a tree.

Property 2: m(G1) > m(G2) if G1 differs from G2 only by the insertion of an extra edge.

Property 3: For i = 1 and 2, let Ai denote the number of edges in Gi and Ni the number of nodes in Gi . Then if N1 > N2 and A1 − N1 + 1 = A2 − N2 + 1 then m(G1) < m(G2).

Property 4: For all graphs G, m(G) ≤ m(KN) = 1, where N = number of nodes of G and KN is the complete graph of N nodes.

$$m(G) = \frac{2(e - n + 1)}{(n - 1)(n - 2)}$$

   - Measure procedure: Programatic.
   - Requirement : Graph / program module.

- **Internal reuse:** The extent to which modules within a product are used multiple times within the same product.

  $r(G) = e - n + 1$

  - Measure procedure: Programatic.
  - Requirement : Graph / program module.

- **Information Flow** - A module invokes a second module and passes information to it.

  Information flow complexity(M) = length(M) × ((fan-in(M) × (fan-out(M))2

  - Measure procedure: Programatic / Manually.
  - Requirement : Graph / program module.

- **Coupling in Object-Oriented Systems** – connections between elements from one module to others.

  Instability metric , I = Ce / Ca + Ce.

  Here, Ce = Efferent coupling. [Fan-in]

  Ca = Afferent coupling. [Fan-out]

  - Measure procedure: Programatic.
  - Language : OOP.

- **Cohesion in Object-Oriented Systems** – connection between elements in a individual module.

  $TCC(C) = NDC(C)/NP(C)$

  $LCC(C) = (NDC(C) + NIC(C))/NP(C)$

  $RC(P) = (R(P) + 1)/N(P)$

Here, TCC = Tight Class Cohesion

LCC = Loose Class Cohesion

NDC = Number of Direct Cohesion

NIC = Number of Indirect Connection

NP = Number of Possible Connections

RC = Relational Cohesion

R(P) = number of relations between classes and interfaces

N(P) = number of classes and interfaces in the package

- Measure procedure: Manually.
- Language : OOP.