Measuring Internal Product Attributes: Size

Code Size measurement:

Code size is a internal Product attributes, Program code is an integral component of software. Such code includes source code, intermediate code, byte code, and even executable code. We can measure code size in apply many procedures such as Counting Lines of Code, Halstead's Approach, number of bytes of computer storage, number of characters (CHAR) Etc.

Counting Lines of Code : We can measurement code size using counting Lines of code, we build a program for counting line of code automated ..Automated is better then statically measure of line of code for code size measuring.

Halstead's Approach: Halstead's Approach is a another approach for code size measuring ,It $\mu 1$ = Number of unique operators, $\mu 2$ = Number of unique operands ,N1 = Total occurrences of operators ,N2 = Total occurrences of operands and finally calculated $V = N \times \log 2\mu$ where $\mu=(\mu 1+ \mu 2)$ and N=(N1+N2)..We use program to calculated it easily.

Number of bytes of computer storage: We can build a program for calculate the total byte of computer storage for measuring code size.

Number of bytes of computer storage: We can build a program for calculate the total byte of computer storage for measuring code size.

DESIGN SIZE measurement:

Design size is another Internal product attributes, we measure the design size easily when used OOP concept ,Today OOP(Object oriented concept) is very powerful concept for design the software system .We can measure Design Size using more mechanisms such as Packages, Design patterns Classes or interfaces or abstract classes, Methods or operations.

Packages: We can find out the total packages of the software system easily in manually.

Design patterns: Number of different design patterns used in a design measure the total uses design patterns, It will measure manually.

Classes or interfaces or abstract classes: Total Number of classes, Interface or Abstract classes measure easily in manually.

Methods or operations: Number of methods or operations measure in automated to make a program. weighted methods per class (WMC) also measure based on Total methods within Software system

REQUIREMENTS ANALYSIS AND SPECIFICATION SIZE:

Requirements Analysis and specification size in a another Internal product attributes, Requirements and specification documents generally combine text, graphs, and special mathematical diagrams and symbols. We can measure requirement analysis based on more items such as Use case diagrams, Use case, Domain model (expressed as a UML class diagram), UML OCL specifications, Alloy models, Data-flow diagrams used in structured analysis and design, Algebraic specifications Etc.

We can easily measurement to using Use case Diagrams, Use case, UML class Diagram and Data Flow Diagram.

Use case diagrams: Number of use cases, actors, and relationships of various types indicate the requirements analysis and specification size, we can measure it manual easily.

Use case: Number of scenarios, size of scenarios in terms of steps, or activity diagram model elements indicate the requirements analysis and specification size, we can measure it manual easily.

UML class diagram: Number of classes ,Actors and Relationship of various types indicate the requirements analysis and specification size, we can measure it manual easily.

Measuring Internal Product Attributes: Structure

Hierarchical Measures:

McCabe's Cyclomatic Complexity Measure:

Cyclomatic complexity is a software metric (measurement) used to indicate the complexity of a program. It is a quantitative measure of the number of linearly independent paths through a program's source code. Based on the numbers given per method in the source code, one can easily tell if the code is complex or not. We can say that a program per methods class is indicate a Cyclomatic complexity. Normally cyclomatic complexity measured using this steps-

1.Generate the Control flow graph from program

2.then define the number o nodes and arc(edges) and also decision nodes

3.After computing Step 1 and 2 then we used four approach for measuring the cyclomatic complexity like as v=e-n+2p, v=d+1, v=number of R.

We know that this step 1 is very difficult to apply so we can select to per methods class process for find out the cyclomatic complexity measure easily.. We build a program for measure the per methods in class approach.

Design level attribute:

We find out the level of design of software system for knowing the design output like as design is bad or good.. We follow more approach for measure the design-level such as Models of Modularity and Information Flow, Global Modularity, Morphology, Tree Impurity, Internal Reuse, Information Flow Etc.

Models of Modularity and Information Flow: we can find out all of module within the system and show the information flow from one module to another modules. We can see the dependency modules when information will sharing among the modules. It will be easily to saw the manually from the software system..

Global Modularity: "Global modularity" is difficult to define because there are many different views of what modularity means. For example, consider average module length as an intuitive measure of global modularity. metric was developed to calculate the average size of program modules as a measure of structuredness .We can measure global modularity using the program for automated calculated.

Morphology: They use the notion of morphology to refer to the "shape" of the over all system structure when expressed pictorially. Many morphological characteristics are measurable directly, including the Size, Depth, Width, Edge-to-node ratio. It measure be easily with manually when we draw the graph.

Tree Impurity: The more a system deviates from being a pure tree structure towards being a graph structure, the worse the design is … it is one of the few system design metrics* to have been validated on a real project. We can easily measure it using following function G(m)=2(e-n+2)/(n-1)(n-2) When we find out the graph manually then we use those information to estimate the above function automated.

Internal Reuse: We call internal reuse the extent to which modules within a product are used multiple times within the same product .We use the following method to estimate the Internal Reuse R(r)=(e-n+1) Above method will be automated when graph's edge and nodes find out the manually checking.

Information Flow: Each type of coupling corresponds to a particular type of information flowing through the module. Researchers have attempted to quantify other aspects of information flow, including

• The total level of information flow through a system, where the modules are viewed as the atomic components (an intermodular attribute)

• The total level of information flow between individual modules and the rest of the system (an intramodular attribute We can used the following technique to measure the Information flow complexity(M) – Information flow complexity(M) = length(M) × ((fan-in(M) × (fan-out(M))2---(1) We find out the Information flow complexity automated using the length of module,fan-in of module and also fan-out of module.