**REQUIREMENT BASED COMPLEXITY (RBC) Measure**

As said before, complexity has to do with all those factors which make source code difficult to comprehend. This measure is based on the factors derived from Software Requirement Specification (SRS) Document. The advantage of this approach is that it is able to estimate the software complexity in early phases of software life cycle, even before analysis and design is carried out. Due to this fact, this is cost effective and less time consuming complexity metric.

Now, to generate a measure based on SRS, attributes which are captured from SRS for finding the RBC are given in Figure 2.3.



Figure 2.3 Factor derivation from SRS for RBC (Sharma and Kushwaha, 2010)

Now the calculation method for this measure based on different parameters is illustrated.

**Complexity Attribute 1: Input Output Complexity (IOC)**

This complexity refers to the input and output of the software system and attached interfaces and files. Following four attributes are considered:

***Input***: As Information entering to the System

***Output:*** Information Leaving System

***Interface:*** User Interface where the Input are to be issued and output to be seen and specifically number of integration required

***Files:*** This refers to the data storage required during transformation

Now, Input Output Complexity can be defined as:

**IOC = No. of Input + No. of Output + No. of Interfaces + No. of files (1)**

**Complexity Attribute 2: Functional Requirement (FR)**

Functional Requirement defines the fundamental actions that must take place in the software in accepting and processing the inputs and in processing and generating outputs. Functionality refers to what system is supposed to do. This describes the general factor that affects the product and its functionality. Every stated requirement should be externally perceivable by users, operators or other external systems.

It may be appropriate to partition the functional requirement into sub-functions or sub-processes

**FR = No. of Functions \* **   **(2)**

Where SPF is Sub Process or Sub-functions received after decomposition.

**Complexity Attribute 3: Non Functional Requirement (NFR)**

This refers to the Quality related requirements for the software apart from functionality. These requirements are categorized into THREE categories with their associated precedence values as shown in Table 1. As high the precedence that much high will be the value, which will further depend upon the count. It can be mathematically described as:

NFR = ****  (3)

**Table 2.1: Types of Non Functional Requirement**

|  |  |
| --- | --- |
| Type | Count |
| Optional Requirement | 1 |
| Must be Type | 2 |
| Very Important Type | 3 |

**Complexity Attribute 4: Requirement complexity (RC)**

It refers to the sum of all requirements i.e. functional and its decomposition into sub-functions and non functional requirements:

**RC = FR \* NFR (4)**

**Complexity Attribute 5: Product Complexity**

This refers to the overall complexity based on its functionality of the system. This is a product of Requirement Complexity (RC)and Input Output Complexity (IOC). It can be mathematically described as:

**PC = IOC \* RC (5)**

**Complexity Attribute 6: Personal Complexity Attributes**

For effective development of software, technical expertise plays a very significant role. Now computation of the personal attributes lead to technical expertise, and this is referred to as the “Multiplier Values for Effort Calculation i.e. Cost Driver Attributes of Personal Category from COCOMO Intermediate model proposed by Berry Boehm and they are shown in table 2.2

**Table 2.2: Cost Driver Attributes and their values used in COCOMO Model**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Attribute** | **Rating** | | | | | | |
| **Very low** | **Low** | | **Normal** | **High** | **Very High** | |
| **Analyst Capability** | **1.46** | **1.19** | **1.00** | | **0.86** | **0.71** |
| **Application Exp.** | **1.29** | **1.13** | **1.00** | | **0.91** | **0.82** |
| **Programmer Cap.** | **1.42** | **1.17** | **1.00** | | **0.90** | **--** |
| **Virtual Machine Exp.** | **1.21** | **1.10** | **1.00** | | **0.90** | **--** |
| **Programming Language Exp.** | **1.14** | **1.07** | **1.00** | | **0.95** | **--** |

Mathematically PCA can be described as Sum of Product of attributes as mentioned in Table 2.2.

**PCA =**  **** (6)

where Mf are Multiplying Factor.

**Complexity Attribute 7: Design Constraints Imposed (DCI)**

It refers to number of cconstraints that are to be considered during development of software or system by any statutory body or agencies which includes number of regulatory constraints, hardware constraints, communication constraints, database constraints and so on. This metric can be mathematically defined as.

**DCI =  (7)**

Where Ci is the Number of Constraints and the values of Ci varies from 0 to n.



**Complexity Attribute 8: Interface Complexity**

This complexity attribute is used to define number of External Integration or Interfaces to the proposed module/ program/ system. These interfaces can be hardware interface, communication interface and software interface, and so on.

**IFC=  (8)**

Where EIi is Number of External Interfaces and value of EIi varies from 0 to n



**Complexity Attribute 9: Users/Location Complexity**

This measure refers to the number of user for accessing the system and locations (Single or Multiple) on which the system is to be deployed/ used

**ULC= No. of User \* No. of Location (9)**

**Complexity Attribute 10: System Feature Complexity**

This refers to the specific features to be added to the system so as to enhance look and feel feature of the system

**SFC = (Feature1 \* Feature2 \* …………. \* Feature n) (10)**

**Requirement Based Complexity Definition**

Finally, the Requirement Based Complexity can be obtained by considering all above

definitions. It can be mathematically shown as:

**RBC = ((PC \* PCA) + DCI + IFC + SFC) \* ULC (11)**

The Requirement Based Complexity will be higher for the programs, which have higher functionality to be performed and more quality attributes which is to be retained. All above measures would be illustrated with the programs that would be developed to implement Dijkstra algorithm.