Software Metrics

•

Software Metrics

- A software metric is a measure of software characteristics which are measurable or countable.
- Software metrics are valuable for many reasons, including measuring software performance, planning work items, measuring productivity, and many other uses.

Software metrics can be classified into two types as follows:

1. Product Metrics: These are the measures of various characteristics of the software product. The two important software characteristics are:

- Size and complexity of software.
- Quality and reliability of software.

2. Process Metrics: These are the measures of various characteristics of the software development process. For example, the efficiency of fault detection. They are used to measure the characteristics of methods, techniques, and tools that are used for developing software.

Software Metrics

Types of Metrics

- Internal metrics: Internal metrics are the metrics used for measuring properties that are viewed to be of greater importance to a software developer. For example, Lines of Code (LOC) measure.
- External metrics: External metrics are the metrics used for measuring properties that are viewed to be of greater importance to the user, e.g., portability, reliability, functionality, usability, etc.
- Hybrid metrics: Hybrid metrics are the metrics that combine product, process, and resource metrics. For example, cost per FP (Function Point) Metric.
- Project metrics: Project metrics are the metrics used by the project manager to check the project's progress. Data from the past projects are used to collect various metrics, like time and cost

Software Metrics - Advantage of Software Metrics

- Comparative study of various design methodologies of software systems.
- For analysis, comparison, and critical study of different programming languages concerning their characteristics.
- In comparing and evaluating the capabilities and productivity of people involved in software development.
- In the preparation of software quality specifications.
- In the verification of compliance of software systems requirements and specifications.
- In making inferences about the effort to be put into the design and development of the software systems.
- In getting an idea about the complexity of the code.
- In making decisions regarding whether further division of a complex module is to be done or not.
- In comparison and making design tradeoffs between software development and maintenance cost.

Software Metrics

Disadvantage of Software Metrics

- The application of software metrics is not always easy, and in some cases, it is difficult and costly.
- The verification and justification of software metrics are based on historical/empirical data whose validity is difficult to verify.
- These are useful for managing software products but not for evaluating the performance of the technical staff.
- The definition and derivation of Software metrics are usually based on assuming which are not standardized and may depend upon tools available and working environment.
- Most of the predictive models rely on estimates of certain variables which are often not known precisely.

Size Oriented Metrics - LOC Metrics

- It is one of the earliest and simplest metrics for calculating the size of the computer program. It is generally used to calculate and compare the productivity of programmers.

The following are the points regarding LOC measures:

- In size-oriented metrics, LOC is considered to be the normalization value.
- It is an older method that was developed when FORTRAN and COBOL programming were very popular.
- Productivity is defined as KLOC / EFFORT, where effort is measured in person-months.
- Size-oriented metrics depend on the programming language used.
- As productivity depends on KLOC, assembly language code will have more productivity.

Size Oriented Metrics

- LOC measure requires a level of detail which may not be practically achievable.
- The more expressive is the programming language, the lower is the productivity.
- LOC method of measurement does not apply to projects that deal with visual (GUI-based) programming. As already explained, Graphical User Interfaces (GUIs) use forms basically. LOC metric is not applicable here.
- It requires that all organizations must use the same method for counting LOC. This is so because some organizations use only executable statements, some useful comments, and some do not. Thus, the standard needs to be established.
- These metrics are not universally accepted.

Based on the LOC/KLOC count of software, many other metrics can be computed:

- Errors/KLOC. Defects/KLOC.
- Pages of documentation/KLOC. Errors/PM.
- Productivity = KLOC/PM (effort is measured in person-months).

Size Oriented Metrics

Advantages of LOC

- Simple to measure

Disadvantage of LOC

- It is defined on the code. For example, it cannot measure the size of the specification.
- It characterizes only one specific view of size, namely length, it takes no account of functionality or complexity
- Bad software design may cause an excessive line of code
- It is language dependent
- Users cannot easily understand it

Size Oriented Metrics

Halstead's Software Metrics

- According to Halstead "A computer program is an implementation of an algorithm considered to be a collection of tokens which can be classified as either operators or operands."

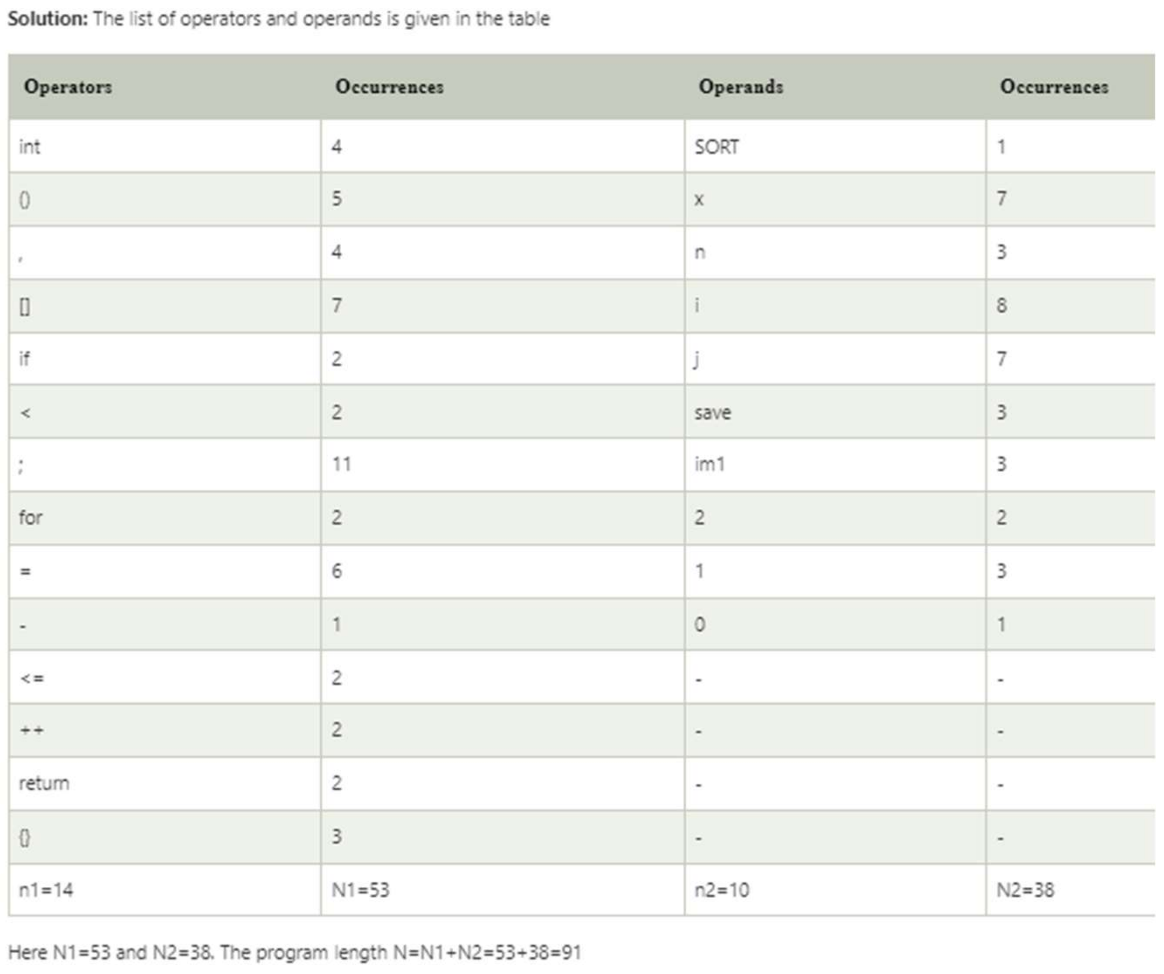
Token Count

- All software metrics can be defined in terms of these basic symbols. These symbols are called tokens.
- The basic measures are

  n1 = count of unique operators.
  n2 = count of unique operands.
  N1 = count of total occurrences of operators.
  N2 = count of total occurrence of operands.

- In terms of the total tokens used, the size of the program can be expressed as N = N1 + N2.

Size Oriented Metrics



Solution: The list of operators and operands is given in the table

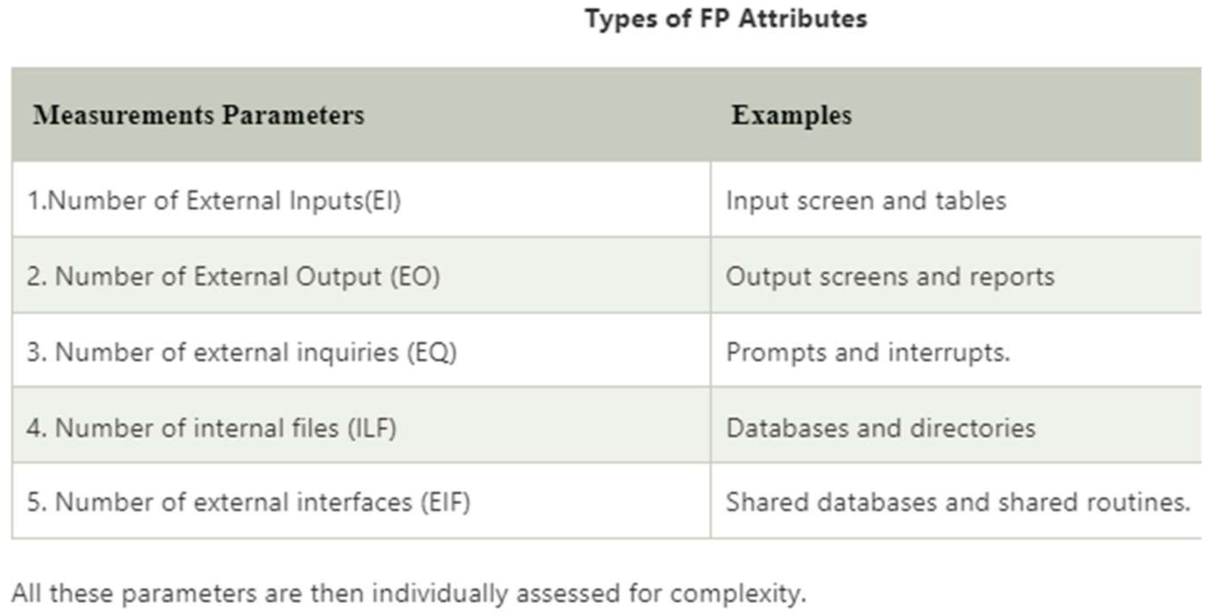
Operators	Occurrences	Operands	Occurrences
int	4	SORT	1
()	5	x	7
,	4	n	3
[]	7	i	8
if	2	j	7
<	2	save	3
;	11	im1	3
for	2	2	2
=	6	1	3
-	1	0	1
<=	2	-	-
++	2	-	-
return	2	-	-
{}	3	-	-
n1=14	N1=53	n2=10	N2=38

Here N1=53 and N2=38. The program length N=N1+N2=53+38=91

Size Oriented Metrics

Functional Point (FP) Analysis and objectives

- functional point analysis is to measure and provide the software application functional size to the client, customer, and the stakeholder on their request. Further, it is used to measure the software project development along with its maintenance, consistently throughout the project irrespective of the tools and the technologies.

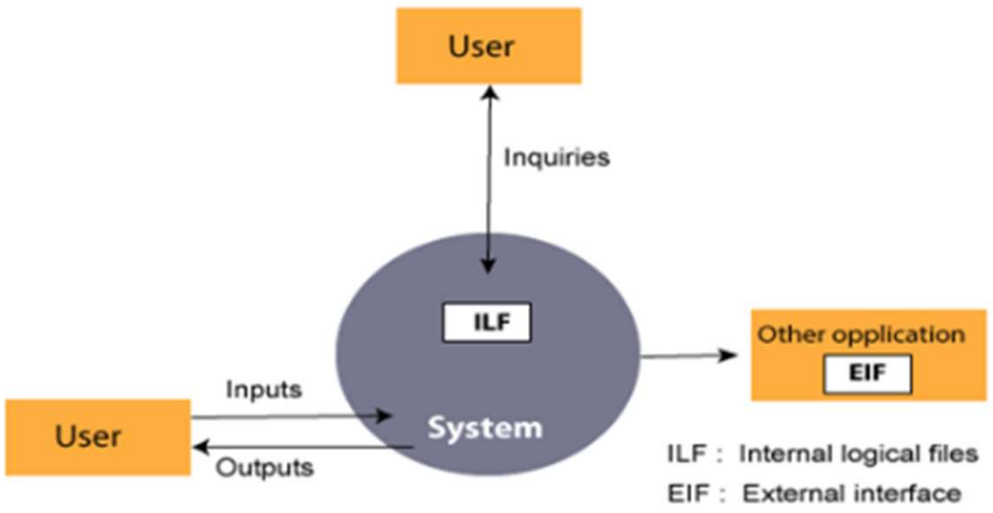


Types of FP Attributes

Measurements Parameters	Examples
1.Number of External Inputs(EI)	Input screen and tables
2. Number of External Output (EO)	Output screens and reports
3. Number of external inquiries (EQ)	Prompts and interrupts.
4. Number of internal files (ILF)	Databases and directories
5. Number of external interfaces (EIF)	Shared databases and shared routines.

All these parameters are then individually assessed for complexity.

Size Oriented Metrics



FPAs Functional Units System

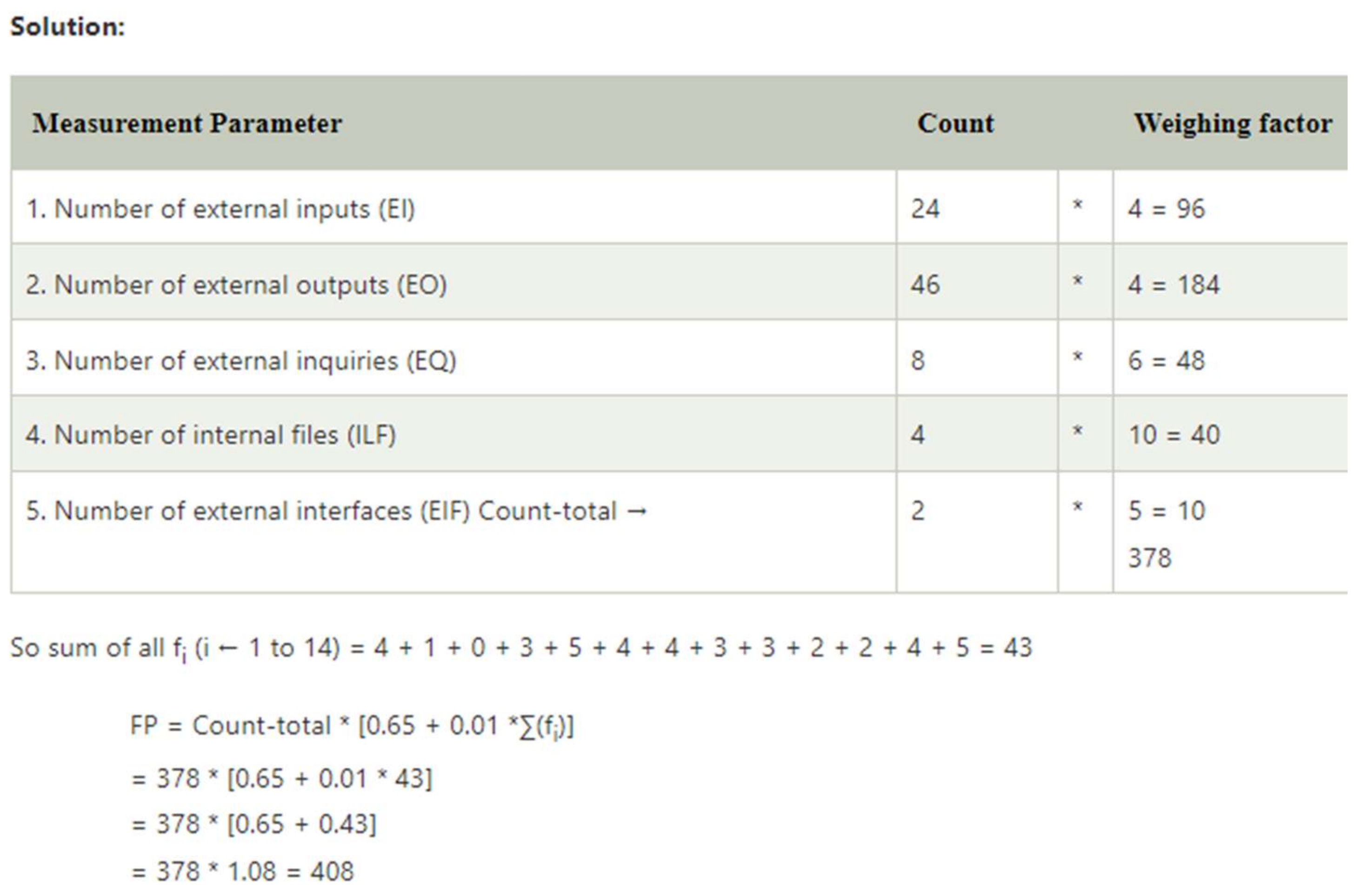
Size Oriented Metrics

Example: Compute the function point, productivity, documentation, cost per function for the following data:

- Number of user inputs = 24
- Number of user outputs = 46
- Number of inquiries = 8
- Number of files = 4
- Number of external interfaces = 2
- Effort = 36.9 p-m
- Technical documents = 265 pages
- User documents = 122 pages
- Cost = $7744/ month

Various processing complexity factors are: 4, 1, 0, 3, 3, 5, 4, 4, 3, 3, 2, 2, 4, 5.

Size Oriented Metrics



Solution:

Measurement Parameter	Count		Weighing factor
1. Number of external inputs (EI)	24	*	4 = 96
2. Number of external outputs (EO)	46	*	4 = 184
3. Number of external inquiries (EQ)	8	*	6 = 48
4. Number of internal files (ILF)	4	*	10 = 40
5. Number of external interfaces (EIF) Count-total →	2	*	5 = 10 378

So sum of all f_i (i ← 1 to 14) = 4 + 1 + 0 + 3 + 5 + 4 + 4 + 3 + 3 + 2 + 2 + 4 + 5 = 43

FP = Count-total * [0.65 + 0.01 *$\sum(f_i)$]

= 378 * [0.65 + 0.01 * 43]

= 378 * [0.65 + 0.43]

= 378 * 1.08 = 408

Size Oriented Metrics

$$\text{Productivity} = \frac{\text{FP}}{\text{Effort}} = \frac{408}{36.9} = 11.1$$

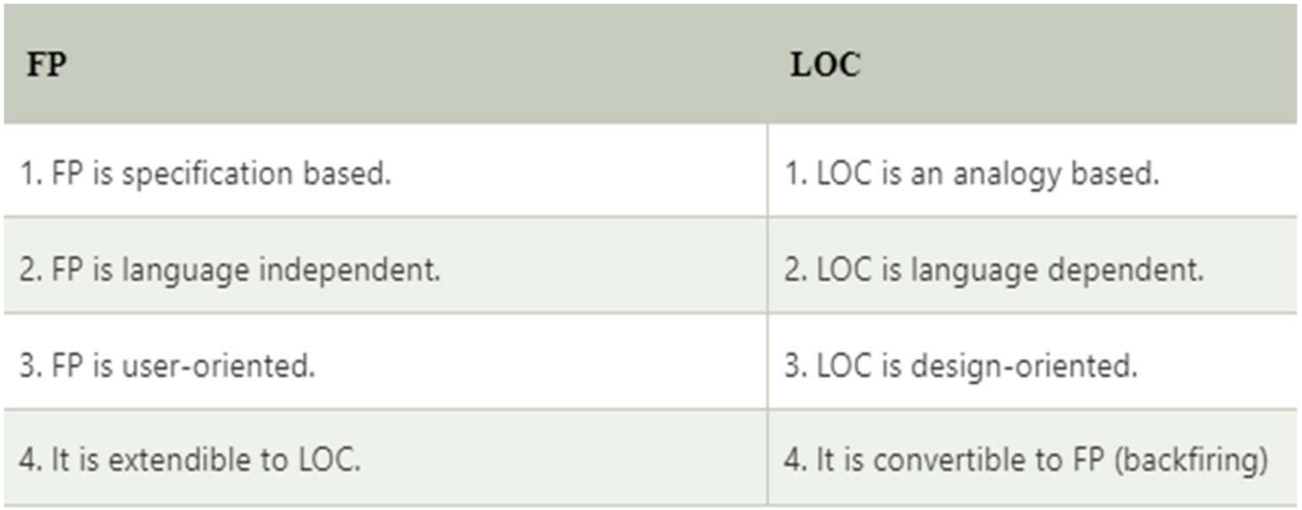
Total pages of documentation = technical document + user document

= 265 + 122 = 387pages

Documentation = Pages of documentation/FP

= 387/408 = 0.94

$$\text{Cost per function} = \frac{\text{cost}}{\text{productivity}} = \frac{7744}{11.1} = \$700$$



FP	LOC
1. FP is specification based.	1. LOC is an analogy based.
2. FP is language independent.	2. LOC is language dependent.
3. FP is user-oriented.	3. LOC is design-oriented.
4. It is extendible to LOC.	4. It is convertible to FP (backfiring)

Software Metrics - Cyclomatic Complexity

- Cyclomatic complexity is a software metric used to measure the complexity of a program. It is a quantitative measure of independent paths in the source code of a software program.
- Cyclomatic complexity can be calculated by using control flow graphs or with respect to functions, modules, methods or classes within a software program.
- McCabe (1976) proposed the cyclomatic number, V (G) of graph theory as an indicator of software complexity. The cyclomatic number is equal to the number of linearly independent paths through a program in its graph representation. For a program control graph G, the cyclomatic number, V (G), is given as:

$$V(G) = E - N + 2$$

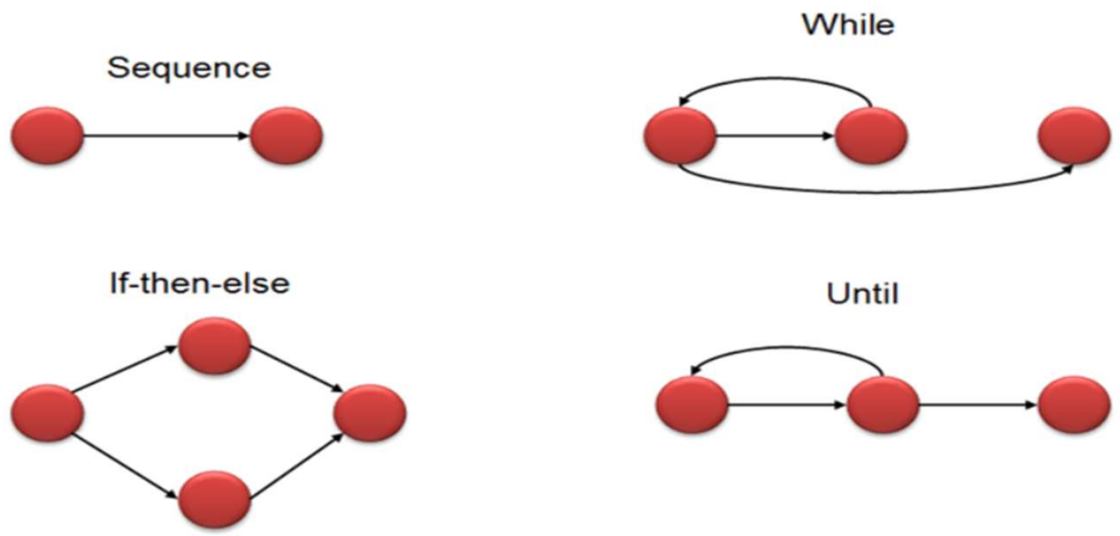
$$V(G) = P + 1$$

- E = The number of edges in graphs G
- N = The number of nodes in graphs G
- P = Number of predicate nodes (node that contains condition)

Software Metrics - Cyclomatic Complexity

Flow graph notation for a program

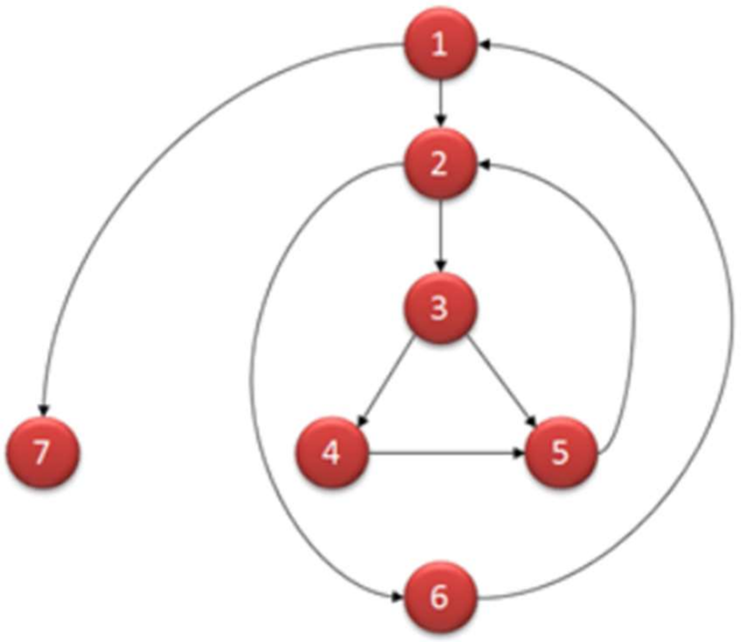
Flow Graph notation for a program defines several nodes connected through the edges. Below are Flow diagrams for statements like if-else, While, until and normal sequence of flow.



Complexity can be found in the number of decision points in a program. The decision points are if, for, for-each, while, do, catch, and case statements in a source code.

Software Metrics - Cyclomatic Complexity

```
i = 0;
n=4; //N-Number of nodes present in the graph

while (i<n-1) do
j = i + 1;

while (j<n) do

if A[i]<A[j] then
swap(A[i], A[j]);

end do;
j=j+1;

end do;
```


$$\begin{tikzcd}
 & & 1 \arrow[d] \arrow[lldd, bend right] & \\
 & & 2 \arrow[d] \arrow[ddd, bend right] & \\
 & & 3 \arrow[dl] \arrow[dr] & \\
7 & 4 \arrow[rr] & & 5 \arrow[uul, bend right] \\
 & & 6 \arrow[uuuu, bend right] &
\end{tikzcd}$$

Software Metrics - Cyclomatic Complexity

Computing mathematically,

V(G) = 9 – 7 + 2 = 4

V(G) = 3 + 1 = 4 (Condition nodes are 1,2 and 3 nodes)

Basis Set – A set of possible execution paths of a program

- 1, 7
- 1, 2, 6, 1, 7
- 1, 2, 3, 4, 5, 2, 6, 1, 7
- 1, 2, 3, 5, 2, 6, 1, 7

Properties of Cyclomatic Complexity

The following are the properties of Cyclomatic complexity:

V (G) is the maximum number of independent paths in the graph

V (G) >=1

G will have one path if V (G) = 1

Minimize complexity to 10

Software Metrics

Complexity Number	Meaning
1-10	Structured and well written code High Testability Cost and Effort is less
10-20	Complex Code Medium Testability Cost and effort is Medium
20-40	Very complex Code Low Testability Cost and Effort are high
>40	Not at all testable Very high Cost and Effort

Software Metrics

Case Tools For Software Metrics

Many CASE tools (Computer Aided Software Engineering tools) exist for measuring software. They are either open source or are paid tools. Some of them are listed below:

- **Analyst4j tool** is based on the Eclipse platform and available as a stand-alone Rich Client Application or as an Eclipse IDE plug-in. It features search, metrics, analyzing quality, and report generation for Java programs.
- **CCCC is an open source command-line tool**. It analyzes C++ and Java lines and generates reports on various metrics, including Lines of Code and metrics proposed by Chidamber & Kemerer and Henry & Kafura.
- **Chidamber & Kemerer Java Metrics** is an open source command-line tool. It calculates the C&K object-oriented metrics by processing the byte-code of compiled Java.
- **Dependency Finder** is an open source. It is a suite of tools for analyzing compiled Java code. Its core is a dependency analysis application that extracts dependency graphs and mines them for useful information. This application comes as a command-line tool, a Swing-based application, and a web application.
- **Eclipse Metrics Plug-in 1.3.6** by Frank Sauer is an open source metrics calculation and dependency analyzer plugin for the Eclipse IDE. It measures various metrics and detects cycles in package and type dependencies.

Software Risk Management

•

Software Risk Management

- Risk Management is the system of identifying addressing and eliminating risk/problems before they can damage the project.

There are three main classifications of risks which can affect a software project:

- **Project risks:** Project risks concern diverse forms of budgetary, schedule, personnel, resource, and customer-related problems. A vital project risk is schedule slippage.
- **Technical risks:** Technical risks concern potential method, implementation, interfacing, testing, and maintenance issue. It also consists of an ambiguous specification, incomplete specification, changing specification, technical uncertainty, and technical obsolescence.
- **Business risks:** This type of risks contain risks of building an excellent product that no one need, losing budgetary or personnel commitments, etc.

Software Risk Management

Other Categories of Risk

1. Known risks: Those risks that can be uncovered after careful assessment of the project program, the business and technical environment in which the plan is being developed, and more reliable data sources (e.g., unrealistic delivery date)

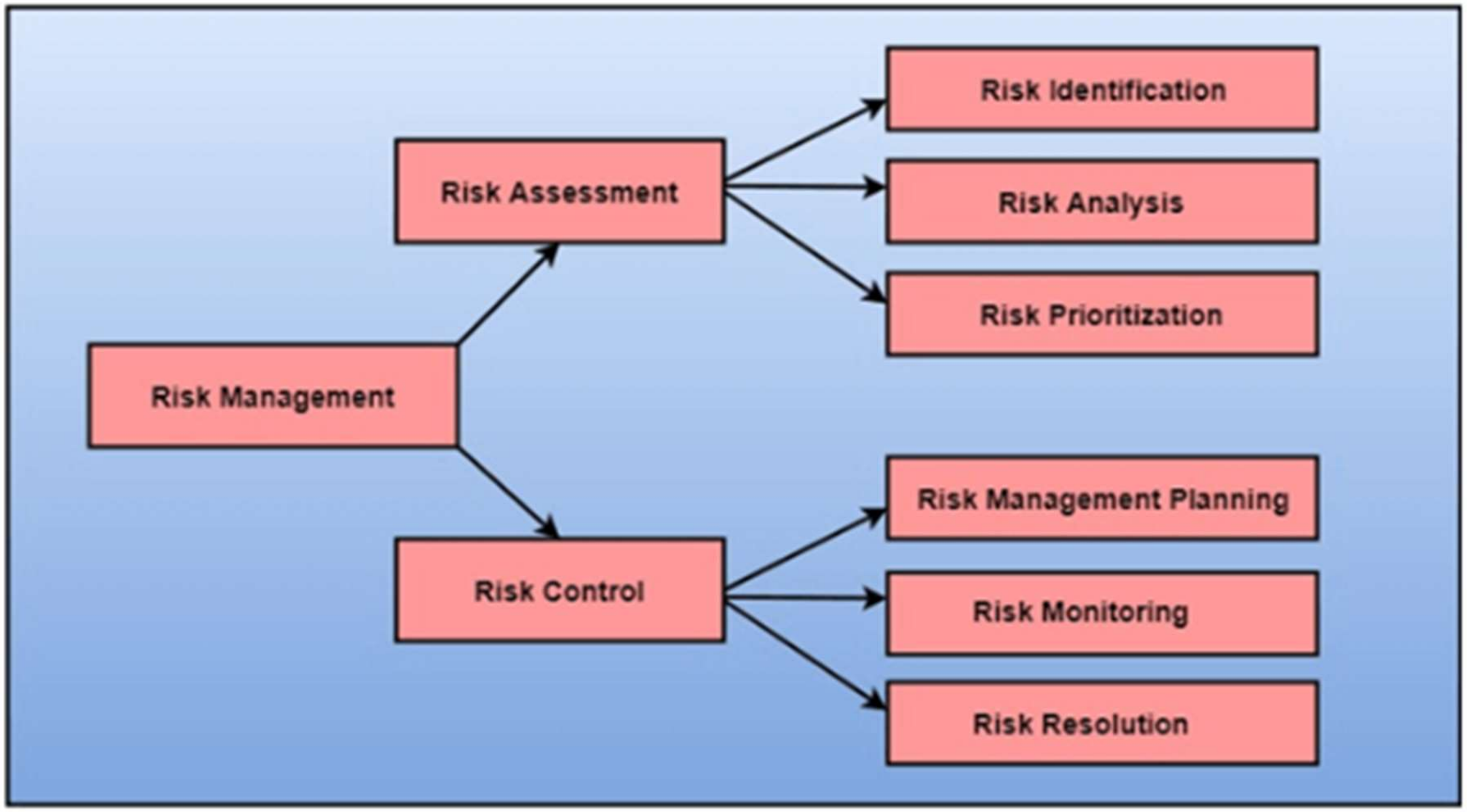
2. Predictable risks: Those risks that are hypothesized from previous project experience (e.g., past turnover)

3. Unpredictable risks: Those risks that can and do occur, but are extremely tough to identify in advance.

Principle of Risk Management

- **Global Perspective:** In this, we review the bigger system description, design, and implementation. We look at the chance and the impact the risk is going to have.
- **Take a forward-looking view:** Consider the threat which may appear in the future and create future plans for directing the next events.
- **Open Communication:** This is to allow the free flow of communications between the client and the team members so that they have certainty about the risks.
- **Integrated management:** In this method risk management is made an integral part of project management.
- **Continuous process:** In this phase, the risks are tracked continuously throughout the risk management paradigm.

Risk Management Activities



Risk Management Activities

Risk Assessment: The objective of risk assessment is to divide the risks in the condition of their loss causing potential.

Risk Identification: The project organizer needs to anticipate the risk in the project as early as possible so that the impact of risk can be reduced by making effective risk management planning.

Risk Analysis: During the risk analysis process, you have to consider every identified risk and make a perception of the probability and seriousness of that risk.

Risk Management Activities

- **Risk Control:** It is the process of managing risks to achieve desired outcomes.
- **Risk Leverage:** To choose between the various methods of handling risk, the project plan must consider the amount of controlling the risk and the corresponding reduction of risk.
- **Risk Planning:** The risk planning method considers each of the key risks that have been identified and develop ways to maintain these risks.
- **Risk Monitoring:** Risk monitoring is the method king that your assumption about the product, process, and business risks has not changed.

Risk Management Planning

- **There are three main methods to plan for risk management**
- **Avoid the risk:** This may take several ways such as discussing with the client to change the requirements to decrease the scope of the work, giving incentives to the engineers to avoid the risk of human resources turnover, etc.
- **Transfer the risk:** This method involves getting the risky element developed by a third party, buying insurance cover, etc.
- **Risk reduction:** This means planning method to include the loss due to risk. For instance, if there is a risk that some key personnel might leave, new recruitment can be planned.