Lecture 2

## Software Metrics

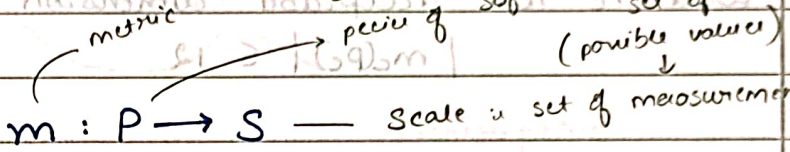
A quantitative measure of the degree to which a system, component or process possesses a given attribute. IEEE 1990

(31) Quality Metric :

1> A quantitative measure of the degree to which an item possesses a given quality attribute.

2> A function whose input are software data and whose output is a single numerical value that can be interpreted as the degree to which the software possesses a given quality attribute.

Definition :




"A metric is a function that assigns to each proband $p \in P$ a valuation $m(p) \in S$."
We call S the scale of m.

Examples :

* Agricultural Engineering : $m_a : P_a \to S_a$
  - probands $P_a$ : milk sample
  - Scale $S_a$ : $[0,100]_{\text{fat}} \times [0,100]_{\text{protein}}$ % fat & protein

can be interpreted as (the degree of) [...] quality

milk sample $p_a$ has acceptable quality if $m_a(p_a) \geq (4.0, 3.4)$

* Railway Engineering : $m_r : P_r \to S_r$
  - probands $P_r$ : trams
  - Scale $S_r$ : $R_0^+$ braking dist from 70 km/h to 0 km/h

tram Pr has acceptable evasive braking quality if $m_R(p_R) \leq 69$

* Construction Engineering : $m_C : P_C \rightarrow S_C$
  - proband $P_C$ : walls (length upto 3m)
  - Scale $S_C = R$ deviation from nominal in mm

wall $P_C$ has acceptable dimension if

$$|m_C(p_C)| \leq 12$$

## Uses of Software Metrics

1> Specify Product Properties
Eg : Code must be in MISRA C
Metric - No of MISRA-C violation = 0

2> Assess Product Properties / Support Decisions
Eg : The system is responsive for 100 concurrent users
Metric - average miliseconds b/w event & response

3> Project Management :
Eg : Do not have too many bug reports
Metric : No of open bug reports

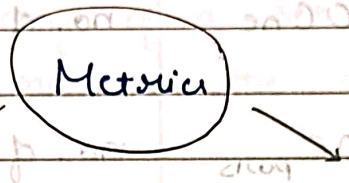
4> Predict / Estimate / Forecast :
Eg : Effort estimation for new project.
Metric : Person days / months.

8.

6> Research / State and Investigate Hypothesis

Eg: The SWT course audience is not homogenous regarding previous exp.

Metric : No of previous course.




Prescriptive

Stating need/demand non existing software

Descriptive

Stating diagnosed (measured) / prognosed (predicted) existing software.

## Desirable Properties of (Software) Metrics

1> Relevant

2> Plausible

3> Robust

4> Available

5> Economical

6> Comparable

7> Reproduceable

8> Differentiated.

Eg: Lines of Code [LOC]

| Dimension | Unit | measurement procedure |
|---|---|---|
| Program Size | $LOC_{tot}$ | no. of lines in total |
| net Prog Size | $LOC_{ne}$ | no. of non empty lines |
| Code Size | $LOC_{pars}$ | no. of lines with not only comments & non printable |
| delivered program Size | $DLOC_{tot}$<br>$DLOC_{ne}$<br>$DLOC_{pars}$ | LOC of code that is delivered to customer |

```
1  /* . . . .
2    * . . . . .
3    * . . . . . */
4  (
5  class Hallo {
6  ⌴ ↪ white Spac
7  ⌴ . . . // code
8  ⌴ . . . // code
9  ⌴ . . . // code
10 ⌴ . . . // code
11 ⌴ . . . }
12 }
```

$LOC_{tot} = 12$

$LOC_{ne} = 11$ $(12-1) = 11$ (4) comments

$LOC_{pars} = 7$ $(12-4-1)$ (1 to 4) (6) non printable

# Line with whit space is not empty.

# McCabe Complexity

**complexity :**

The degree to which a system / component has a design or implementation that is difficult to understand and verify.

→ Pertaining to any of a set of structure based metrics that measure the attribute.

**Definition [Cyclomatic Number (Graph Theory)]**

Let $G = (V, E)$ be a graph comprising vertices $V$ and edges $E$. ↳ CONTROL FLOW GRAPH

The cyclomatic number of $G$ is defined as

$$v(G) = |E| - |V| + 1$$

minimum no of Edges to be removed to make G cycle free

→ Let $G = (V, E)$ be the Control Flow Graph of program P.

⇒ Then the cyclomatic complexity of P is defined as

$$v(P) = |E| - |V| + (p)$$

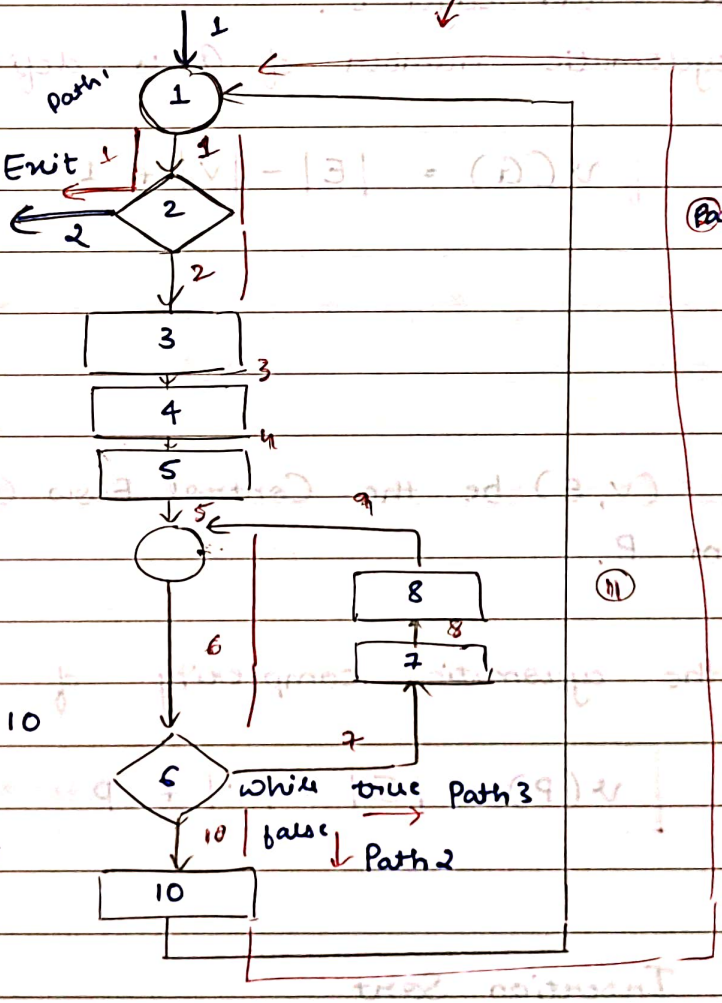
→ no. of entry / exit points.

Eg :

Insertion Sort

```
1>  void insertionSort (int[] array) {
2>      for (int i = 0 2 ; i < array.length ; i++) {
3>          tmp = array[i];
4>          array[0] = tmp;
5>          int j = i;
6>          while (j > 0 && tmp < array[j - 1]) {
7>              array[j] = array[j - 1];
8>              j--;
x 9 ————————}
10>         array[j] = tmp;
        }
    }
```

no 9 below




Path 2,3

No of edges :

$|E| = 11$

Number of nodes

$|V| = 6 + 2 + 2 = 10$

External connection :

$p = 2$

$\Rightarrow v(P) = |E| - |V| + P = 11 - 10 + 2$

$\boxed{v(P) = 3}$

If there is a if() else() block the no of paths is $2^n$ (n - no of blocks)

For 3 → $2^3 = 8$ paths for 3 if() else() blocks

→ Intuition : No of paths

→ Easy to compute ;

↳ Interval Scale (not absolute, no zero due to $p > 0$);

→ Somewhat independent from programming language.

→ Plausability

+ loops and conditions are harder to understand than seq. sequencing

− doesn't consider data

## Prescriptive USE

" For each procedure, either limit cyclomatic complexity to [agreed upon limit] or provide explanation of limit exceeded."

## CODE METRICS for OO Programs (Chidamber Kemerer)

(Object Oriented.

table in ppt.

## ASPECTS OF SOFTWARE QUALITY

SOFT. QUALITY

Process Quality

Product Quality


* Reliability : The capability of the software product to maintain a specified level of performance when used under specific conditions

* Fault Tolerance : The capability of the software product to maintain a specified level of performance in cases of software faults or infringement of specified interface.

* Functionality : The capability of the software product to provide functions which meet stated and implied needs when the software is used under specified condition

Suitability : The capability of the software product to provide an appropriate set of functions for specified tasks and user obj.

## Metrics ISSUES

| Base | Derived |
| --- | --- |
| → measure defined in terms of attribute & method for quantifying it. | → measure defined as function of two or more values of base measures. |
| Eg: LOC, execution time | Eg : avg lines of code, productivity. |

Derived measures are easier to get wrong
Be extra careful with them

### Issues with Scales

1> People like aggregated data
2> People like to Compare data

# Scales and Types of Scales

| Scale | $=, \neq$ (with transitivity) | $<, >$ | min, max eg: median | percentile | $\Delta$ | proportion | natural zero |
|---|---|---|---|---|---|---|---|
| nominal | ✓ | × | × | × | × | × | × |
| ordinal | ✓ | ✓ | ✓ | ✓ | × | × | × |
| interval | ✓ | ✓ | ✓ | ✓ | ✓ | × | × |
| rational | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| absolute | ✓ | rational scale where S comprises the key figure itself | | | | | |

↳ has a median but in general not an average in the scale

Eg: Nominal Scale → programming language
Ordinal Scale → leader board [CMMI scale]
Interval Scale → Temperature scale (used in in control system)
Rational Scale → Age, program runtime
Absolute Scale → No of seats on bus, no of known errors

## KINDS OF METRIC : by Measurement Procedure

| | Objective | Pseudo | Subjective |
|---|---|---|---|
| Procedure | measurement, counting Standardized | computation | review by inspectors verbal or scale |
| Eg | LOC, no of bugs | BMI ↳ | usability |
| Used for | collection of base measure | COCOMO cost estimation for predictions, overall assessment | quality assessment error weighing |
| Adv | exact, reproducible | relevant direct usable statement | not subjective, plausible |
| DisAdv | not always relevant, often subjective | hard to comprehend, does not actually measure | assessment costly quality depends on inspector |

**Pseudo Metrics** : For many of the most relevant aspects of software development projects
→ maintainability → product usability
Today we don't have good objective measures metrics
So only.

| | Plausible | Robust | Available | Economical | comparable | Reproducible |
|---|---|---|---|---|---|---|
| Subj | ✓ | ✓ | ✓ | × | ✓ | × |
| Pseudo | × | × | ✓ | ✓ | ✓ | ✓ |

**NOTE** Not every derived metric is a pseudo metric
→ average LOC per module (derived, not pseudo)
→ maintainability in average LOC per module (derived, pseudo)

**Useful and Non Useful Pseudo Metrics**

pseudo metric valuation m(P)

| Quality | LOW | HIGH |
|---|---|---|
| HIGH | False -ve | True +ve |
| LOW | True -ve | False +ve |

**USEFUL** : pseudo metric m with good correlation

**NOT USEFUL** : pretty random, too many false positives

What metrics should we USE ?

Approach : Understand what we need to know, then choose develop metrics that measure that.
Eg : Goal Question metric (GQM) Basili & Weiss

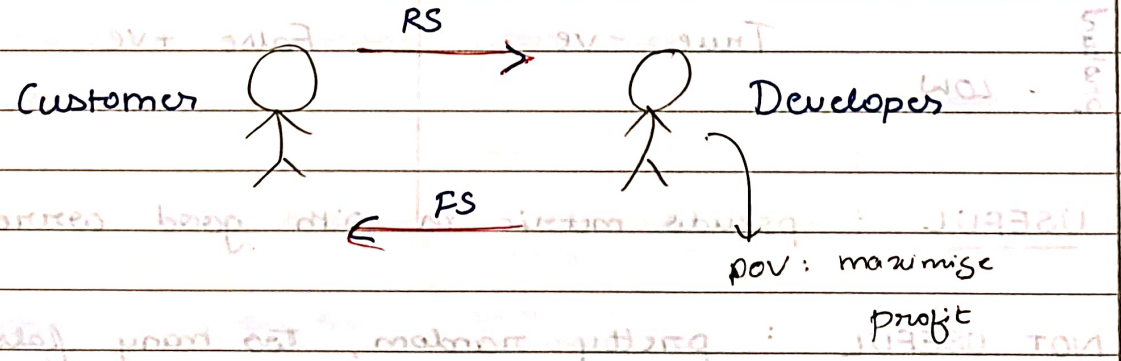
Often Useful : Collect some basic measures continously.

## COST ESTIMATION

Why estimate cost ?

Requirement Specification : Entire demands of deliverables and services of a developer within a contracted development, created by customer.

Feature Specification : Specification of how to realize a given requirements specification, created by developer.



# Principles of Software Cost Estimation

* Experience : We extrapolate similar project costs and finalize on the numbers.

Classification : (table in PPT) (Estimation Funnel)

## COCOMO

## Delphi Method

Step 1 : Leader asks team members to write down estimates.

Step 2 : Leaders ask team members to show estimates and explain

Step 3 : Estimate again and then Median is taken.

# Algorithmic Estimation — COCOMO

## # Constructive Cost Model

Flavors : COCOMO 81 , COCOMO II

All flavors are based on estimated program size S measured in DSI [Delivered Source Instructions] or KDSI (1000 DSI)

### Basic COCOMO

=> Effort Required : $E = a \cdot (S/KDSI)^{b}$ [PM] (person months)

(a) ← table, (S) ← size, KLOC lines of code in kilos, (b) ← table

=> Time to Develop : $T = C \cdot E^{d}$ [months]

=> head count : $H = E/T$ [FTE Full-time empl.]

=> Productivity : $P = S/E$ [DSI/PM]

↳ use to check for plausibility

| SIZE | CHARACTERISTIC OF TYPE: INNOVATION | DEADLINES | DEV ENVIRONMENT | a | b | Project Type |
|---|---|---|---|---|---|---|
| Small <50 KLOC | Little | Not tight | Stable | 3.2 | 1.05 | Organic |
| Medium <300 KLOC | Mid | Medium | Medium | 3.0 | 1.12 | Semi detatched |
| Large | Great | Tight | Complex HW Interface | 2.8 | 1.20 | Embedded |

## Intermediate COCOMO

$$E = M \cdot a \cdot (S/KDSI)^{b} \quad \text{(person months)}$$

$M = RELY \cdot CPLX \cdot TIME \cdot ACAP \cdot PCAP \cdot LEXP \cdot TOOL \cdot SEED$