Software Metrics

- Sanghamitra De

Software Metrics

- A quantitative measure of the degree to which a system, component, or process possesses a given attribute
- Types: Process & Product

Process Metrics

- ➢ Uses process indicators
- ➢ Process metrics derived indirectly, based on outcomes of a process
- ➢ Outcomes include measures of:
  - ✓ errors uncovered before release of the software
  - ✓ defects delivered to and reported by end-users
  - ✓ work products delivered (productivity)
  - ✓ human effort expended
  - ✓ calendar time expended
  - ✓ schedule conformance.
- ➢ Process metrics are collected across all projects over long periods of time.
- ➢ Provides indicators that lead to long-term software process improvement

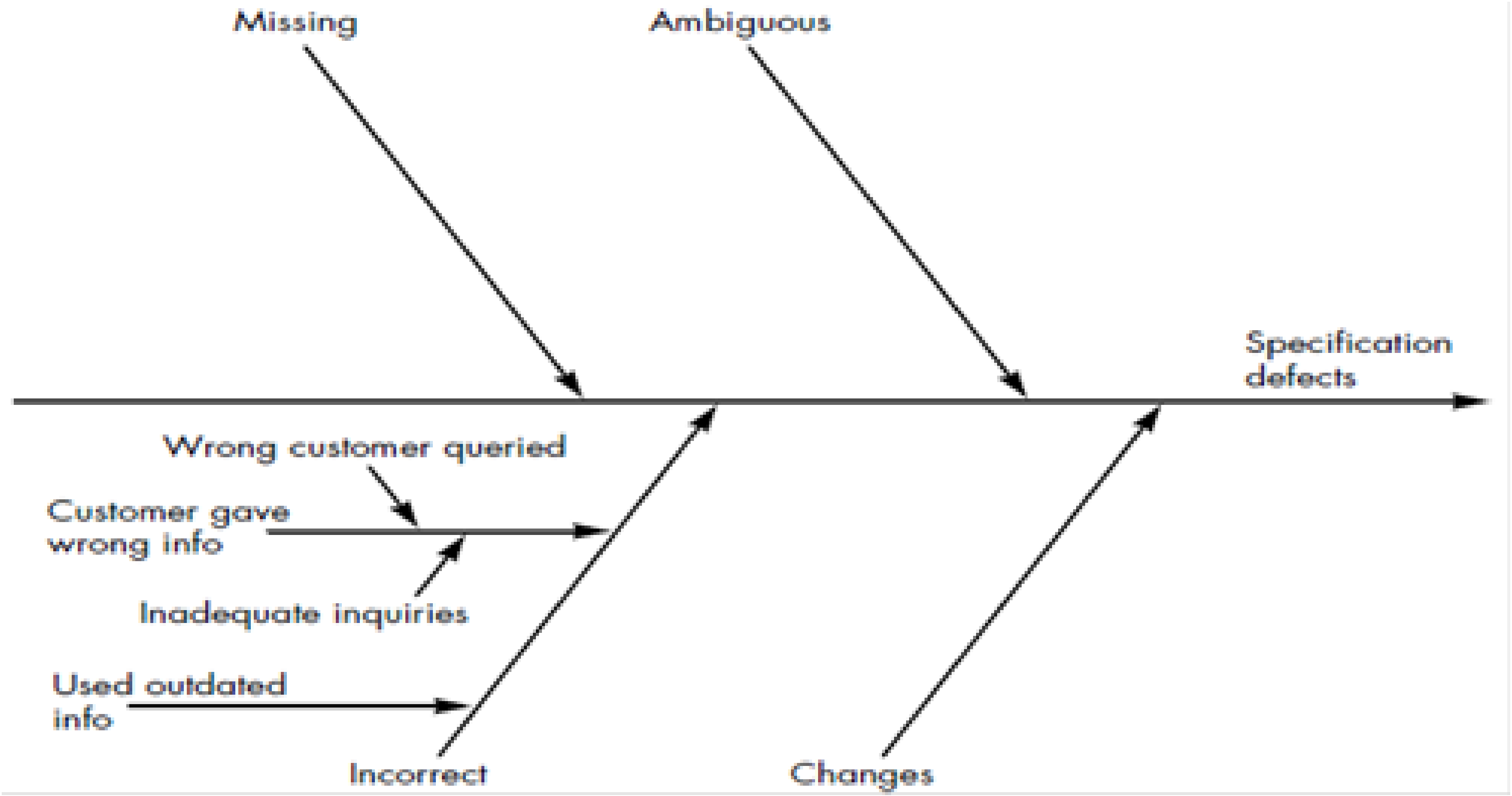
Process Metrics

- Statistical Software Process Improvement (SSPI) method is used to collect process metrics.
- SSPI uses software failure analysis to collect information about all errors and defects encountered as an application, system, or product is developed and used.

How does failure analysis work?

- All errors and defects are categorized by origin (e.g., flaw in specification, flaw in logic, non-conformance to standards).
- The cost to correct each error and defect is recorded.
- The number of errors and defects in each category is counted and ranked in descending order.
- The overall cost of errors and defects in each category is computed.
- Resultant data are analyzed to uncover the categories that result in highest cost to the organization.
- Plans are developed to modify the process with the intent of eliminating (or reducing the frequency of) the class of errors and defects that is most costly.

Use of fishbone diagram to diagnose defect data



Project Metrics

- Used to minimize the development schedule by making the adjustments necessary to avoid delays and mitigate potential problems and risks.
- Used to assess product quality on an ongoing basis and, when necessary, modify the technical approach to improve quality.

Project Metrics

- ➢ Production rates:
  - ✓ pages of documentation
  - ✓ review hours
  - ✓ function points
  - ✓ delivered source
  - ✓ errors uncovered during each software engineering task
- ➢ Technical metrics to assess design quality

Project Metrics

- Projects should measure:
  - ✓Inputs—measures of the resources (e.g., people, environment) required to do the work.
  - ✓Outputs—measures of the deliverables or work products created during the software engineering process.
  - ✓Results—measures that indicate the effectiveness of the deliverables.

Size oriented metrics

Typical size oriented metrics:

- Errors per KLOC (thousand lines of code).
- Defects per KLOC.
- $ per LOC.
- Page of documentation per KLOC.
- Errors per person-month.
- LOC per person-month.
- $ per page of documentation

Size oriented metrics: Merits

- LOC as an "artifact" of all software development projects can be easily counted
- Many existing software estimation models use LOC or KLOC as a key input
- A large body of literature and data predicated on LOC already exists.

Size oriented metrics: Demerits

- LOC measures are programming language dependent
- They penalize well-designed but shorter programs
- They cannot easily accommodate nonprocedural languages
- Their use in estimation requires a level of detail that may be difficult to achieve.

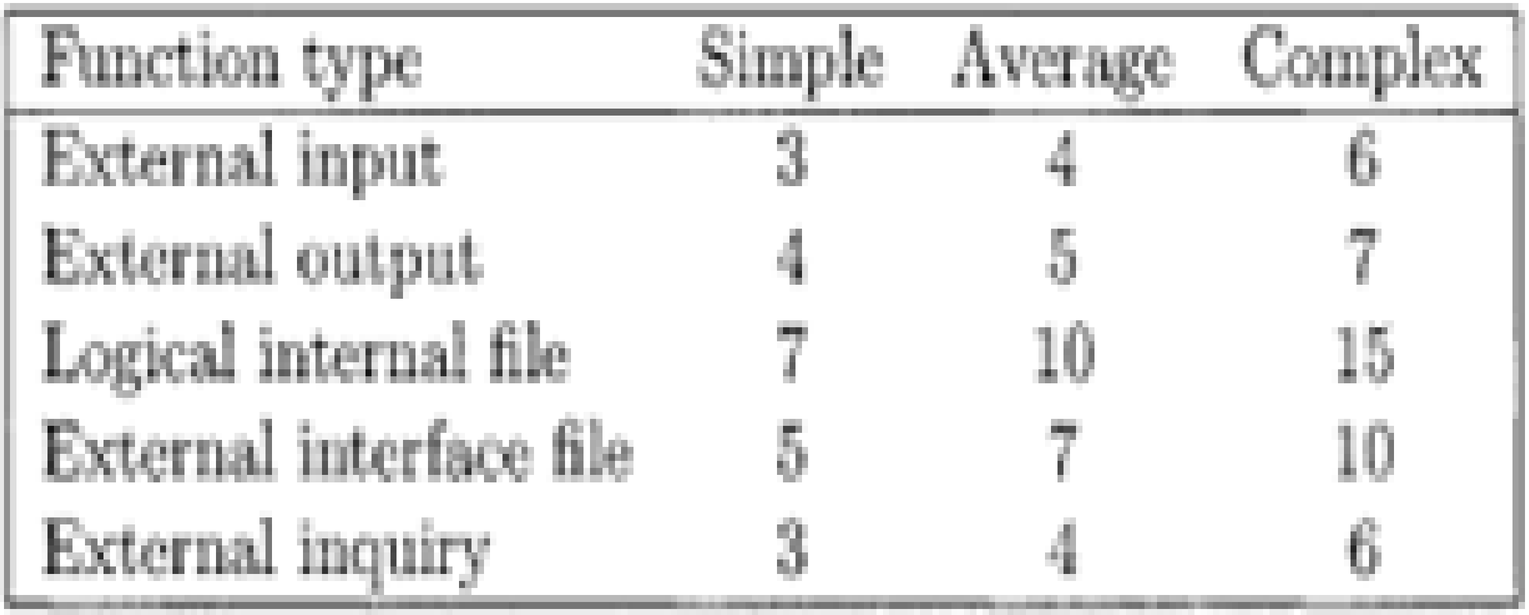
Function-Oriented Metrics (Function Points)

- ➢ First proposed by Albrecht
- ➢ Derived indirectly using other direct measures.
- ➢ The basis of function points is the "functionality" of a system
- ➢ Function points uses the count of five different parameters:
  - ✓ external input types
  - ✓ external output types
  - ✓ logical internal file types
  - ✓ external interface file types
  - ✓ external inquiry types.

Function Points

- To account for complexity, each parameter in a type is classified as *simple*, *average*, or *complex*
- Each element of the same type and complexity contributes a fixed and same amount to the overall function point count of the system.
- Contribution is different for the different types, and for a type, it is different for different complexity levels.

Function Points



Function type	Simple	Average	Complex
External input	3	4	6
External output	4	5	7
Logical internal file	7	10	15
External interface file	5	7	10
External inquiry	3	4	6

Table : Function point contribution of an element.

Function Points

$$UFP = \sum_{i=1}^{i=5}\sum_{j=1}^{j=3} w_{ij}C_{ij},$$

i reflects the row and ***j*** reflects the column in the table above; ***w***$\boldsymbol{w_{ij}}$ is the entry in the ith row and jth column of the table (i.e., it represents the contribution of an element of the type **i** and complexity **j**); and $\mathbf{C_{ij}}$ is the count of the number of elements of type **i** that have been classified as having the complexity corresponding to column **j** .

System characteristics

1) data communications
2) distributed processing
3) performance objectives
4) operation configuration load
5) transaction rate
6) on-line data entry
7) end user efficiency
8) on-line update
9) complex processing logic
10) re-usability
11) installation ease
12) operational ease
13) multiple sites
14) desire to facilitate change

Degree of influence

a) not present (0)

b) insignificant influence (1)

c) moderate influence (2)

d) average influence (3)

e) significant influence (4)

f) strong influence (5)

Complexity Adjustment Factor

- The 14 degrees of influence for the system are then summed, giving a total N (N ranges from 0 to 14*5=70). This N is used to obtain a complexity adjustment factor (CAF) as follows:

CAF = 0.65 + 0.01N

- With this equation, the value of CAF ranges between 0.65 and 1.35

Delivered Function Points = CAF * Unadjusted Function Point

Function Point: Merits

- Definition of Delivered Function Point (DFP) depends only on information available from the specifications, whereas the size in KLOC cannot be directly determined from specifications.
- DFP count is independent of the language in which the project is implemented.

Function Point: Demerits

- Computing function points involves subjective evaluation. Areas of subjectivity:
  - ✓ different interpretations of the SRS
  - ✓ complexity estimation of a user function is totally subjective and depends entirely on the analyst
  - ✓ value judgments for the environment complexity.
- Even when the project is finished, the DFP is not uniquely known and has subjectivity
- Determining the DFP—from either the requirements or a completed project—cannot be automated.

Questions..