Chapter 23

- **Product Metrics**

Measures, Metrics and Indicators

- A *measure* provides a quantitative indication of the extent, amount, dimension, capacity, or size of some attribute of a product or process
- The IEEE glossary defines a *metric* as "a quantitative measure of the degree to which a system, component, or process possesses a given attribute."
- An *indicator* is a metric or combination of metrics that provide insight into the software process, a software project, or the product itself

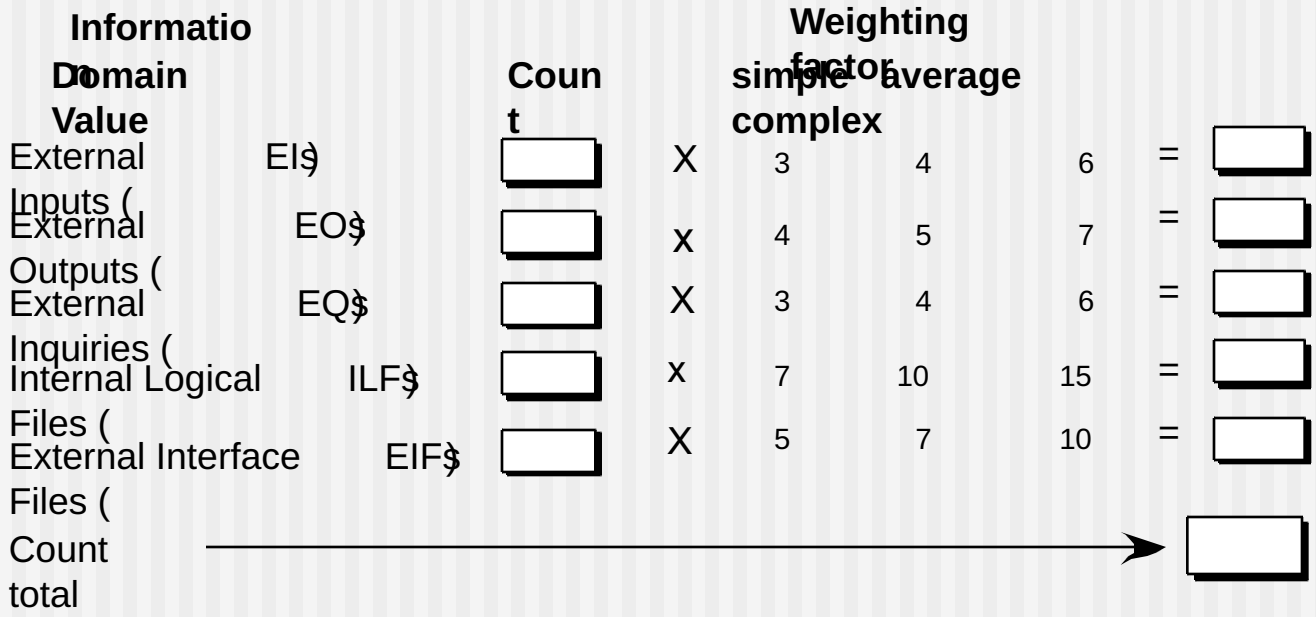
Metrics for the Requirements Model

- Function-based metrics: use the function point as a normalizing factor or as a measure of the "size" of the specification
- Specification metrics: used as an indication of quality by measuring number of requirements by type

Function-Based Metrics

- The *function point metric* (FP), can be used effectively as a means for measuring the functionality delivered by a system.
- Function points are derived using an empirical relationship based on countable (direct) measures of software's information domain and assessments of software complexity
- Information domain values are defined in the following manner:
  - number of external inputs (EIs)
  - number of external outputs (EOs)
  - number of external inquiries (EQs)
  - number of internal logical files (ILFs)
  - Number of external interface files (EIFs)

Function Points



$$FP = \text{count total} * [0.65 + 0.01 * \sum(F_i)]$$

Where F_i is value adjustment factors

The F_i ($i = 1$ to 14) are *value adjustment factors* (VAF) based on responses to the following questions [Lon02]:

1. Does the system require reliable backup and recovery?
2. Are specialized data communications required to transfer information to or from the application?
3. Are there distributed processing functions?
4. Is performance critical?
5. Will the system run in an existing, heavily utilized operational environment?
6. Does the system require online data entry?
7. Does the online data entry require the input transaction to be built over multiple screens or operations?
8. Are the ILFs updated online?
9. Are the inputs, outputs, files, or inquiries complex?
10. Is the internal processing complex?
11. Is the code designed to be reusable?
12. Are conversion and installation included in the design?
13. Is the system designed for multiple installations in different organizations?
14. Is the application designed to facilitate change and ease of use by the user?

Each of these questions is answered using a scale that ranges from 0 (not important or applicable) to 5 (absolutely essential). The constant values in Equation (23.1) and the weighting factors that are applied to information domain counts are determined empirically.

[illegible] FP metric in this context, we consider a simple analy-

Code Metrics

- Halstead's Software Science: a comprehensive collection of metrics all predicated on the number (count and occurrence) of operators and operands within a component or program
  - It should be noted that Halstead's "laws" have generated substantial controversy, and many believe that the underlying theory has flaws. However, experimental verification for selected programming languages has been performed (e.g. [FEL89]).

Maintenance Metrics

- IEEE Std. 982.1-1988 [IEE94] suggests a *software maturity index* (SMI) that provides an indication of the stability of a software product (based on changes that occur for each release of the product). The following information is determined:
  - M_T = the number of modules in the current release
  - F_c = the number of modules in the current release that have been changed
  - F_a = the number of modules in the current release that have been added
  - F_d = the number of modules from the preceding release that were deleted in the current release
- The software maturity index is computed in the following manner:
  - SMI = $[M_T - (F_a + F_c + F_d)]/M_T$
- As SMI approaches 1.0, the product begins to stabilize.

Exercise

- Define Measure , Metric and Indicator
- Calculation of Function point
- Explain Maintenance Metrics