PRODUCT, PROCESS AND PROJECT METRICS

Lecture # 40

Lecture by Engr. Sidra

Product Metrics: Metrics for the Requirements Model

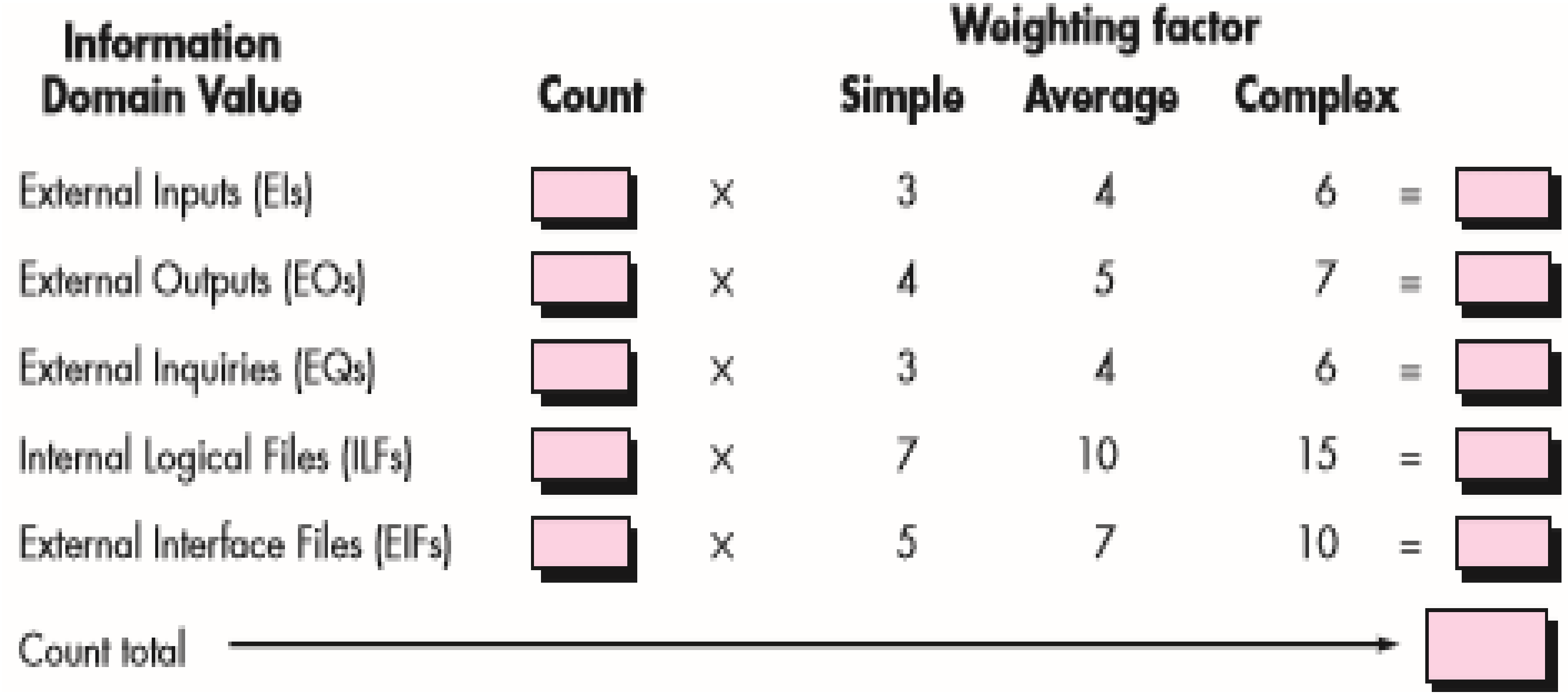
- Function-based metrics: use the function point as a normalizing factor or as a measure of the “size” of the specification
- Specification metrics: used as an indication of quality by measuring number of requirements by type.

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 qualitative 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)

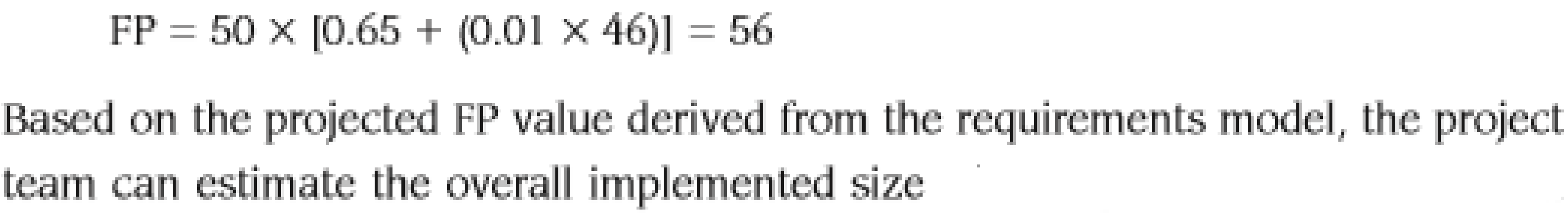
FUNCTION POINTS

- To compute function points (FP), the following relationship is used:
  - FP = count total *[0.65 + 0.01 * Σ(Fi)]



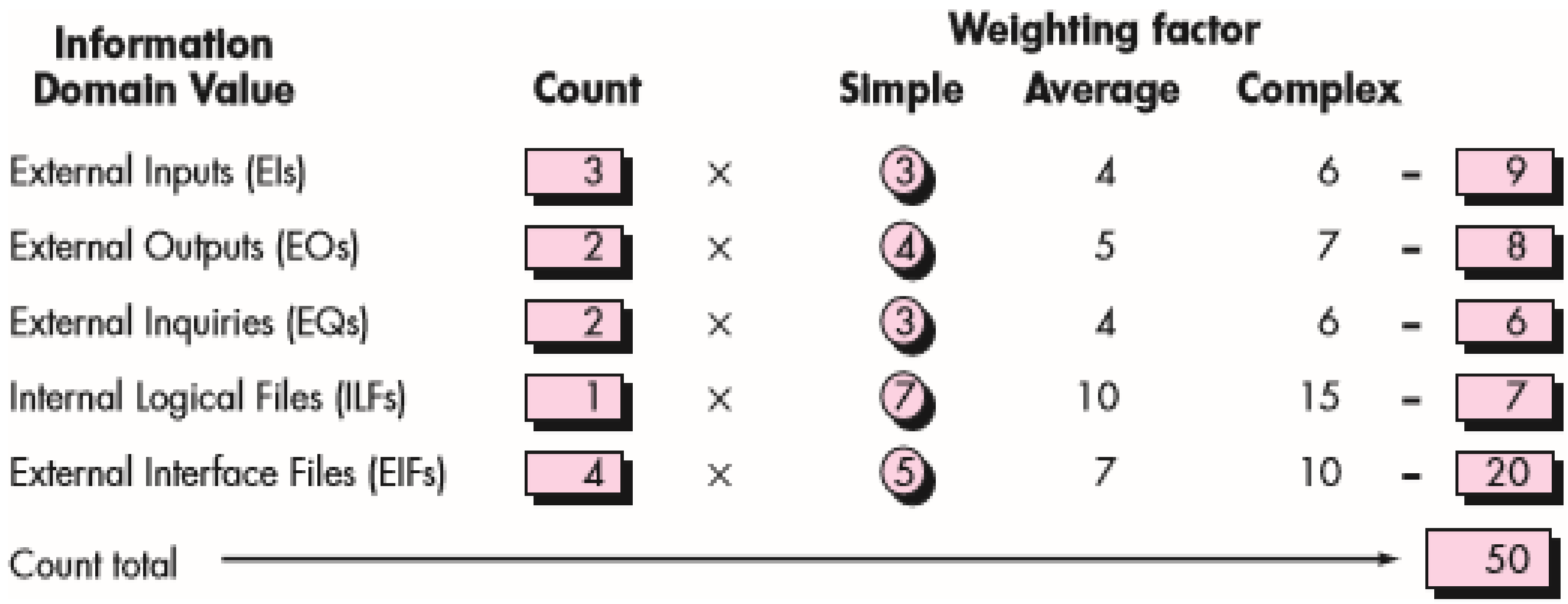
Information Domain Value	Count		Weighting factor Simple	Average	Complex		
External Inputs (EIs)		×	3	4	6	=	
External Outputs (EOs)		×	4	5	7	=	
External Inquiries (EQs)		×	3	4	6	=	
Internal Logical Files (ILFs)		×	7	10	15	=	
External Interface Files (EIFs)		×	5	7	10	=	
Count total						→	

Example



$$FP = 50 \times [0.65 + (0.01 \times 46)] = 56$$

Based on the projected FP value derived from the requirements model, the project team can estimate the overall implemented size



Information Domain Value	Count		Weighting factor				
			Simple	Average	Complex		
External Inputs (EIs)	3	×	③	4	6	=	9
External Outputs (EOs)	2	×	④	5	7	=	8
External Inquiries (EQs)	2	×	③	4	6	=	6
Internal Logical Files (ILFs)	1	×	⑦	10	15	=	7
External Interface Files (EIFs)	4	×	⑤	7	10	=	20
Count total →							50

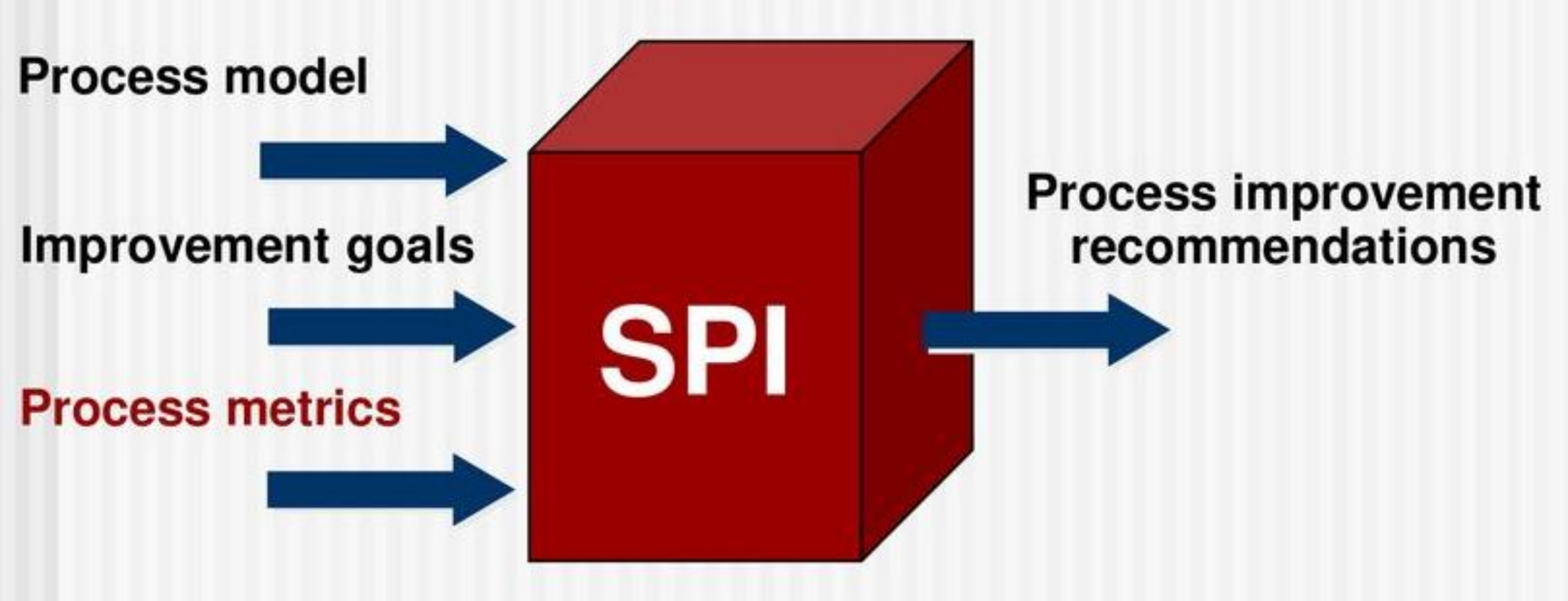
Process Measurement

- Process metrics are collected across all projects and over long periods of time. Their intent is to provide a set of process indicators that lead to long-term software process improvement
- We measure the efficacy of a software process indirectly.
  - That is, we derive a set of metrics based on the outcomes that can be derived from the 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
    - schedule conformance
    - other measures.
  - We also derive process metrics by measuring the characteristics of specific software engineering tasks.

Process Metrics Guidelines

- Use common sense and organizational sensitivity when interpreting metrics data.
- Provide regular feedback to the individuals and teams who collect measures and metrics.
- Don’t use metrics to appraise individuals.
- Work with practitioners and teams to set clear goals and metrics that will be used to achieve them.
- Never use metrics to threaten individuals or teams.
- Metrics data that indicate a problem area should not be considered “negative.” These data are merely an indicator for process improvement.
- Don’t obsess on a single metric to the exclusion of other important metrics.

Software Process Improvement



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.
- Every project should measure:
  - Inputs—measures of the resources (e.g., people, tools) 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.

Typical Project Metrics

- Effort/time per software engineering task
- Errors uncovered per review hour
- Scheduled vs. actual milestone dates
- Changes (number) and their characteristics
- Distribution of effort on software engineering tasks