Software Engineering

Lecture 08

SOFTWARE PROCESS AND PROJECT METRICS

Topic Covered

- Metrics in the process and project domains
- Process, project and measurement
- Process Metrics and Software Process Improvement

Process, project and measurement

Process Metrics:-

Are collected across all projects and over long periods of time. Their intent is to provide a set of process indicator that lead to long term software process improvement.

Project Metrics:-

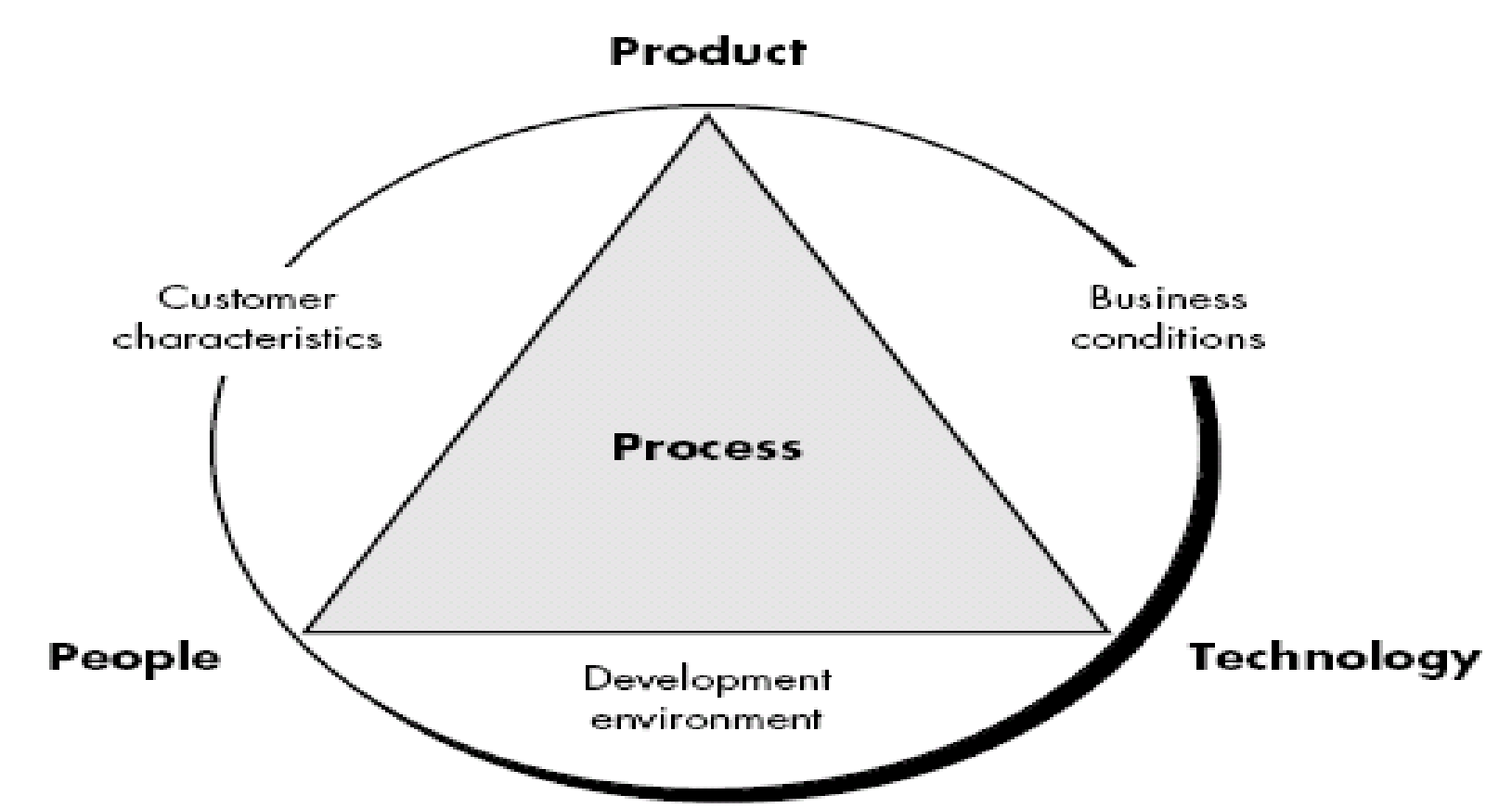
enables a software project manager to

1) **Assess the status of an ongoing project**
2) **Track potential risks.**
3) **Uncover problem areas before they go "Critical"**
4) **Adjust work flow or tasks**
5) **Evaluate the project team's ability to control quality of software work products.**

Measurement :-

Are collected by a project team and converted into process metrics during software process improvement.

Process Metrics and Software Process Improvement



Process Metrics and Software Process Improvement

- Process at the center connecting 3 factors that have a profound influence on software quality and organizational performance.
- Process triangle exists within a circle of environmental conditions that include the development environment, business conditions and customer characteristics.
- 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
    - calendar time 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.

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.

Software Measurement

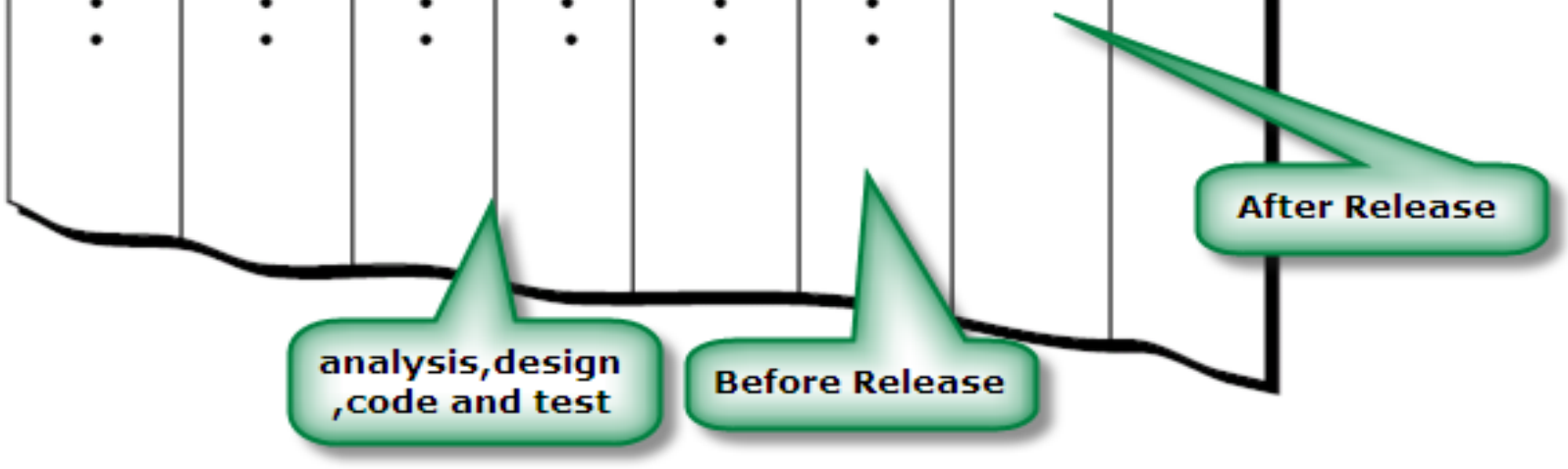
Categories in 2 ways:

- ***Direct measure*** of the software process & Product
  - E.g. Lines of code (LOC), execution speed, and defect)
- ***Indirect measures*** of the product that include functionality, complexity, efficiency, reliability, maintainability etc.
- Team A found : 342 errors
- Team B found : 184 errors
- It is depends on size or complexity of the projects.

Which team is more efficient ?

Size oriented metrics

Project	LOC	Effort	$(000)	Pp. doc.	Errors	Defects	People
alpha	12,100	24	168	365	134	29	3
beta	27,200	62	440	1224	321	86	5
gamma	20,200	43	314	1050	256	64	6
•	•	•	•	•	•		
•	•	•	•	•	•		
•	•	•	•	•	•		



Size-Oriented metrics

Size-oriented metrics measures on LOC as normalization value.

- Errors per KLOC (thousand lines of code)
- Defects per KLOC
- $ per LOC
- Pages of documentation per KLOC
- Errors per person-month
- Errors per review hour
- LOC per person-month
- $ per page of documentation

Function-Oriented Metrics

- It use a measure of functionality delivered by the application as a normalization value.
- Since ‘functionality’ cannot be measured directly, it must be derived indirectly using other direct measures
- Function Point (FP) is widely used as function oriented metrics.
- FP derived using an empirical relationship based on countable (direct) measures of software’s information domain and assessments of software complexity.
- FP is based on characteristic of Software information domain and complexity.
- Like LOC measure, FP is controversial.
- FP is programming language independent.
- It is ideal for applications using conventional and nonprocedural languages.

Reconciling LOC and FP metric

- Relationship between lines of code and function points depends upon the programming language that is used to implement the software and the quality of the design.
- Following table provides rough estimates of the average number of LOC required to build one FP in various programming languages:

Programming Language	LOC/FP (average)
Assembly language	320
C	128
COBOL	106
FORTRAN	106
Pascal	90
C++	64
Ada95	53
Visual Basic	32
Smalltalk	22
Powerbuilder (code generator)	16
SQL	12