CMSC 128

Introduction to Software Engineering
Second Semester AY 2009-2010

jachermocilla@uplb.edu.ph

Project Metrics

'When you can measure what you are speaking about and express it in numbers, you know something about it;'

--Lord Kelvin

Project Metrics

- Measurements are used to improve the software process
  - Estimation, quality control, productivity assessment, and project control
  - Assessment of quality and tactical decision making
- Concerns with *productivity* and *quality* metrics

Project Metrics

- Productivity Metrics
  - Measure of software development output as a function of effort and and time applied
- Quality Metrics
  - 'fitness for use' of the work products produced
- For estimation, historical data is needed

Definitions

- *Measure* – quantitative indication of extent, amount, dimensions, capacity, or size of some attribute of a product or process
- *Measurement*-act of determining a measure
- *Metric*-a quantitative measure of the degree to which a system, component, or process possesses a given attribute

Definitions

- *Indicator*-is a metric or combination of metrics that provide insight into the software process, a software project, or the product itself
  - Errors found per person-hour expended
- *Process indicators*
  - Gain insight into the efficacy of an existing process

Definitions

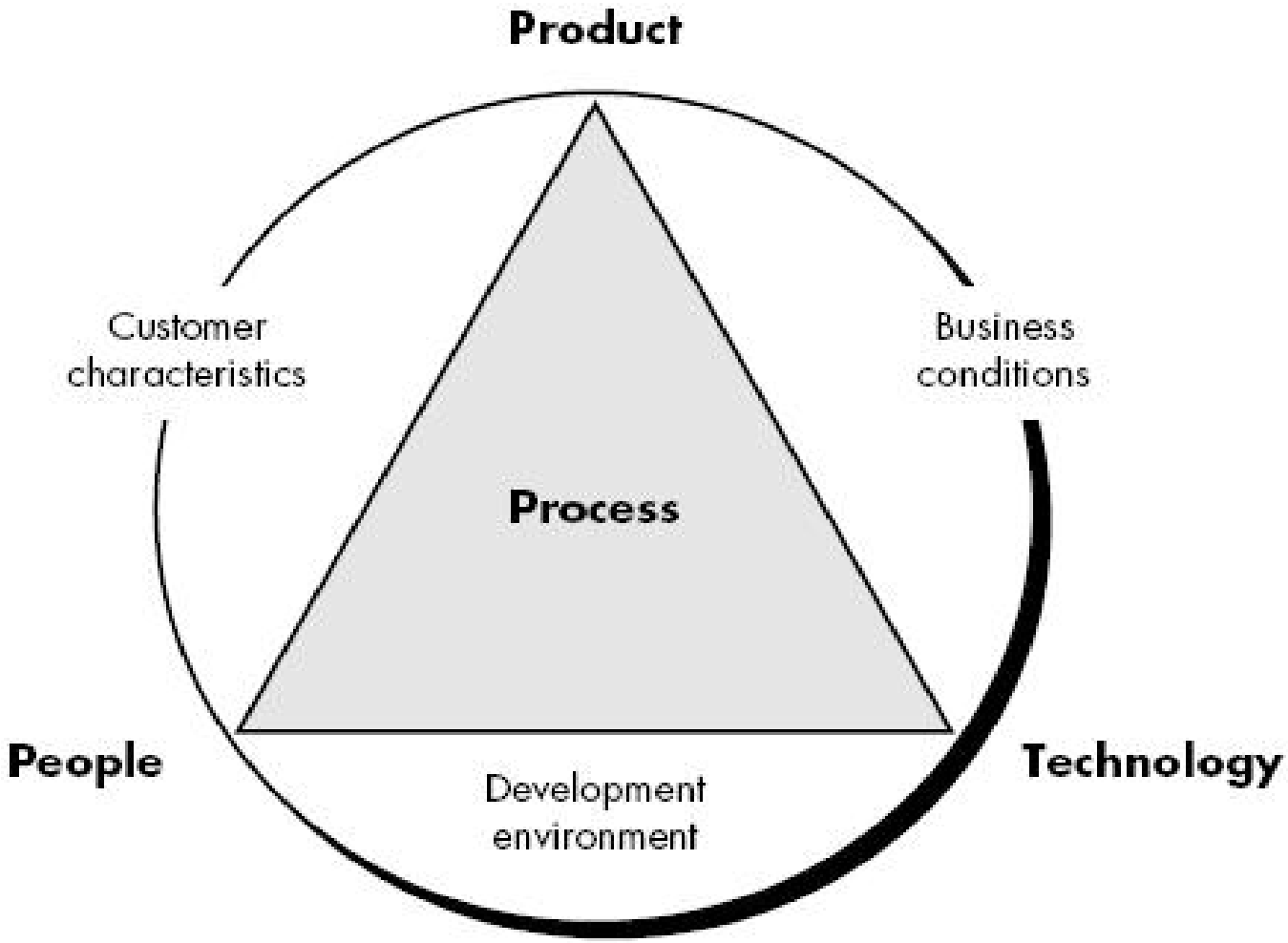
- *Project indicators*
  - Assess the status of an ongoing project
  - Track potential risks
  - Uncover problem areas
  - Adjust work flow or tasks
  - Evaluate the project team's ability to control quality of SE work products

Process Improvement

- Measure specific attributes of the process , develop a set of meaningful metrics based on attributes, use metrics as indicators that will lead to a strategy for improvement
- Process is only one of the controllable factors in improving software quality and organizational performance

Process Improvement

FIGURE 4.1
Determinants for software quality and organizational effectiveness (adapted from [PAU94])



Process Improvement

- Efficacy of process is measured indirectly-metrics is based on outcome that can be derived from the process
  - Errors uncovered, defects reported, work products, effort, calendar time, schedule conformance
- Also, by measuring characteristics of SE tasks
  - Time and effort for umbrella activities

Process Improvement

- Private metrics
  - private to the individual only
  - Defect rate
- Humphrey suggests a Personal Software Process approach-based on private metrics
  - Software process improvement can and should begin at the individual level

Process Improvement

- Some metrics are private to the software project team but public to all team members
  - Defects reported for major software functions
  - FP for module and function
- Public metrics assimilate information that originally was private

Metrics Etiquette

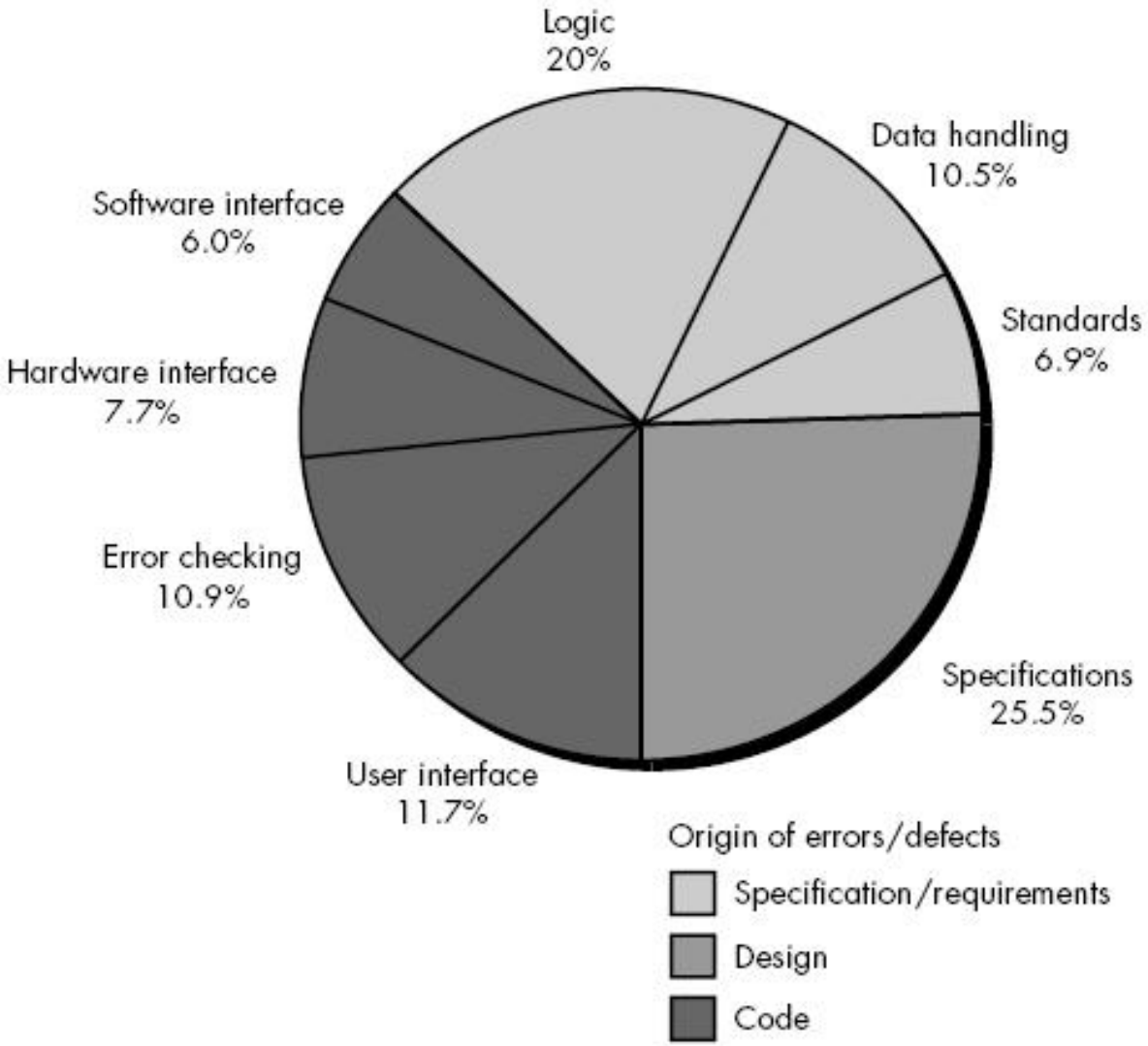
- Use common sense
- Provide regular feedback
- Don't use metrics to appraise individuals
- Work with practitioners and teams
- Don't use metrics to threaten individual or teams
- Don't consider problem area metrics as negative
- Don't use just a single metric

Process Improvement

- Statistical Software Process Improvement
  - Uses software failure analysis to collect information about all errors and defects
    - Errors and defects are categorized
    - Cost to correct error is recorded
    - Errors for each category are counted and ordered
    - Total cost of error in each category is computed
    - Determine categories with highest cost
    - Plans are developed to reduce highest cost categories

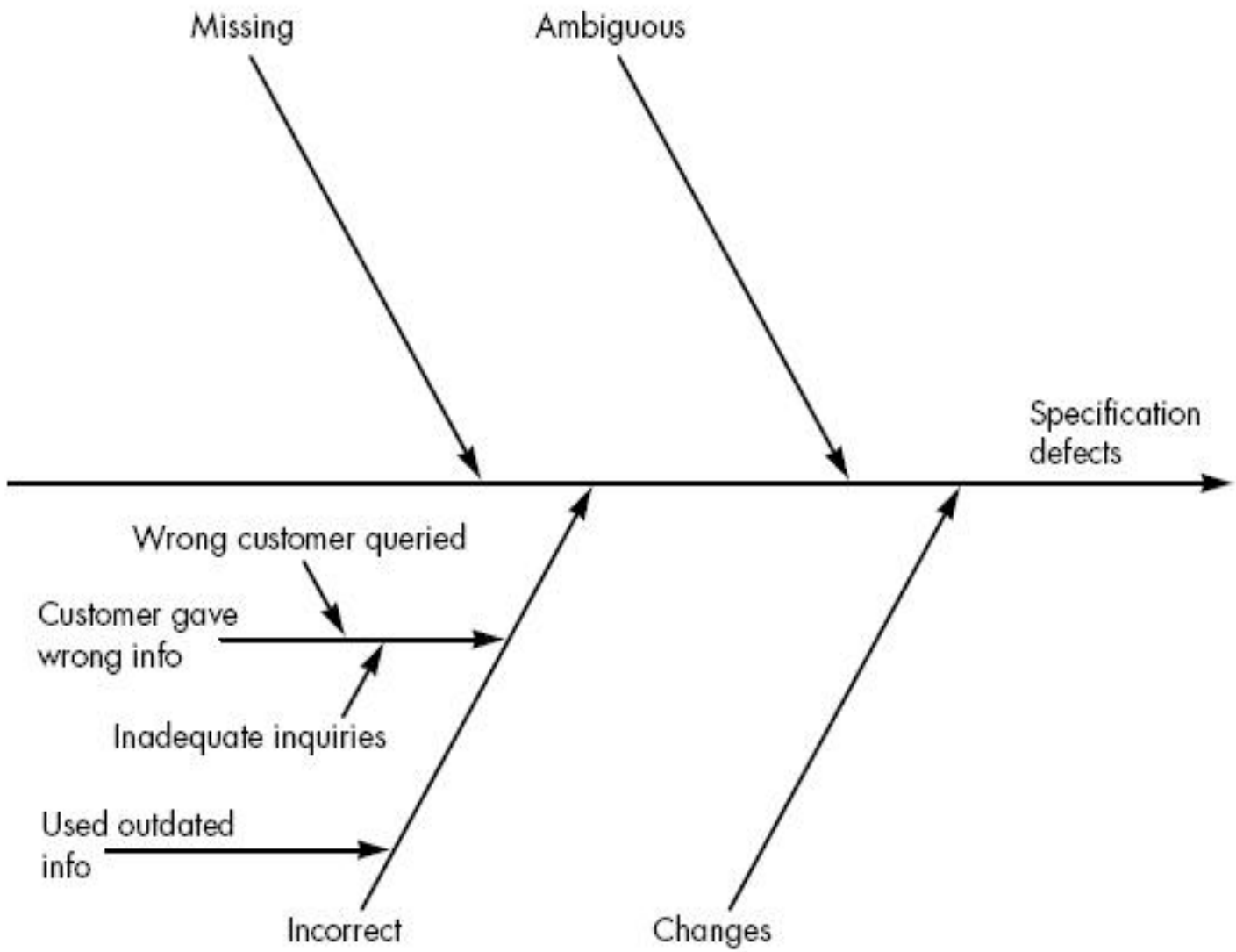
Process Improvement

FIGURE 4.2
Causes of defects and their origin for four software projects [GRA94]



Process Improvement

FIGURE 4.3
A fishbone diagram (adapted from [GRA92])



Project Metrics

- Project metrics are used to adapt workflow and technical activities
  - Historical data is used for effort and time estimation
  - Estimates are compared with actual measures of time and effort
  - Other metrics: production rates, review hours, function points, etc

Project Metrics

- Intent
  1. Minimize the development schedule by guiding the adjustments necessary to avoid delays and mitigate risks
  2. Used to assess product quality

Software Measurement

- Direct
  - Process: Cost and Effort
  - Product: Lines of Code, execution speed, memory footprint, defects reported over some period of time
- Indirect
  - Product: Functionality, complexity, reliability
- Metrics are normalized
  - Measurements come from different projects

Software Measurement

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
⋮	⋮	⋮	⋮	⋮	⋮		

FIGURE 4.4
Size-oriented metrics

Size-oriented metrics

- Lines of Code (LOC) as normalizing value
  - Errors per KLOC
  - Defects per KLOC
  - $ per KLOC
  - Pages of documentation per KLOC
- Others
  - errors/person-month
  - LOC per person-month
  - $/page of doc

Size-oriented metrics

- Not universally accepted(LOC)
- Proponents claim that LOC is an artifact of all software-easily measured
- Opponents claim that LOC is programming language dependent

Function-Oriented Metrics

- Functionality is an indirect measure
  - Must be derived from other direct measures
- Function Points (Albrecht)
  - Derived using an empirical relationship based on *countable measure of information domain* and *assessment of software complexity*

Function-Oriented Metrics

FIGURE 4.5

Computing function points

Measurement parameter	Count		Weighting factor: Simple	Average	Complex		
Number of user inputs		×	3	4	6	=	
Number of user outputs		×	4	5	7	=	
Number of user inquiries		×	3	4	6	=	
Number of files		×	7	10	15	=	
Number of external interfaces		×	5	7	10	=	
Count total							

Function-Oriented Metrics

- Information domain
  - Number of user inputs
  - Number of user outputs
  - Number of user inquiries
  - Number of files
  - Number of external interfaces

Function-Oriented Metrics

$$FP = count_total \times [0.65 + 0.01 \times Sum(Fi)]$$

Function-Oriented Metrics

- Complexity adjustment values (Fi)
  - Backup and data recovery
  - Data Communications
  - Distributed Processing
  - Performance Critical
  - Existing operating environment
  - Online Data Entry
  - Input transaction over multiple screens

Function-Oriented Metrics

- Complexity adjustment values (Fi)
  - Master files updated online
  - Information domain values complex
  - Code designed for reuse
  - Conversion/Installation in design
  - Multiple installations
  - Application designed for change
  - Complexity adjustment factor

Function-Oriented Metrics

- Extensions
  - Feature points
    - Includes algorithms
  - 3D Function Point (Boeing)
    - Functional, data, and control
- Pros: PL independent, based on data known early in the project
- Cons: Subjective, no direct physical meaning

LOC and FP

- Often used to derive productivity metrics
- Should these metrics be used to appraise the performance of individuals?

  –NO!

- Other factors influence productivity: people, problem, process, product, resource, etc.

Metrics for Quality

- Measure to achieve high quality software
- The quality of a system is as good as the requirements that describes the problem, the design that models the solution, the code that leads to the executable program, and the tests to uncover errors
- At the project level, measure errors and defects

Measuring Quality

- Correctness
  - Degree to which the software performs its required function; defects/KLOC, where defect is a verified lack of conformance to requirements
- Maintainability
  - Ease with which a program can be corrected if an error is encountered, adapted if environment changes, or enhanced when customer desires change in requirements

Measuring Quality

- Maintainability
  - Mean-time-to-change (MTTC), time-oriented
  - Spoilage, cost-oriented, cost to correct defects after release
- Integrity
  - System'a ability to withstand attacks on program, data, and documents
  - Integrity = Sum(1 – threat x (1 – security)]), for all types of attacks

Measuring Quality

- Usability
  - User-friendliness
    - Physical/Intellectual skill to learn the system
    - Time to become moderately efficient in the use of the system
    - Net increase in productivity
    - User's subjective assessment

Defect Removal

- Defect Removal Efficiency (DRE) is a measure of filtering ability of quality assurance and control activities as applied throughout the process framework activities
  - DRE = E / (E +D)
    - E- errors found before delivery
    - D- defects found after delivery

Summary

- Measurements improve process, project, and product
- Process metrics provides strategic view by providing insight to the effectiveness of process
- Project metrics are tactical and allows managers to adapt work flow
- Industry uses size- and function-oriented metrics

Reference

- Roger S. Pressman.Software Engineering: A Practitioner's Approach, 4th Ed.McGraw-Hill,1997. Chapter 4