投票 最多可选1项

设置

Are you ready?

A Yes

B No

提交

Review - SCM 4

- The Management Spectrum: The Four P's ?
  - People (most important)
  - Process
  - Product
  - Project (Common-Sense Approach)

Software Engineering

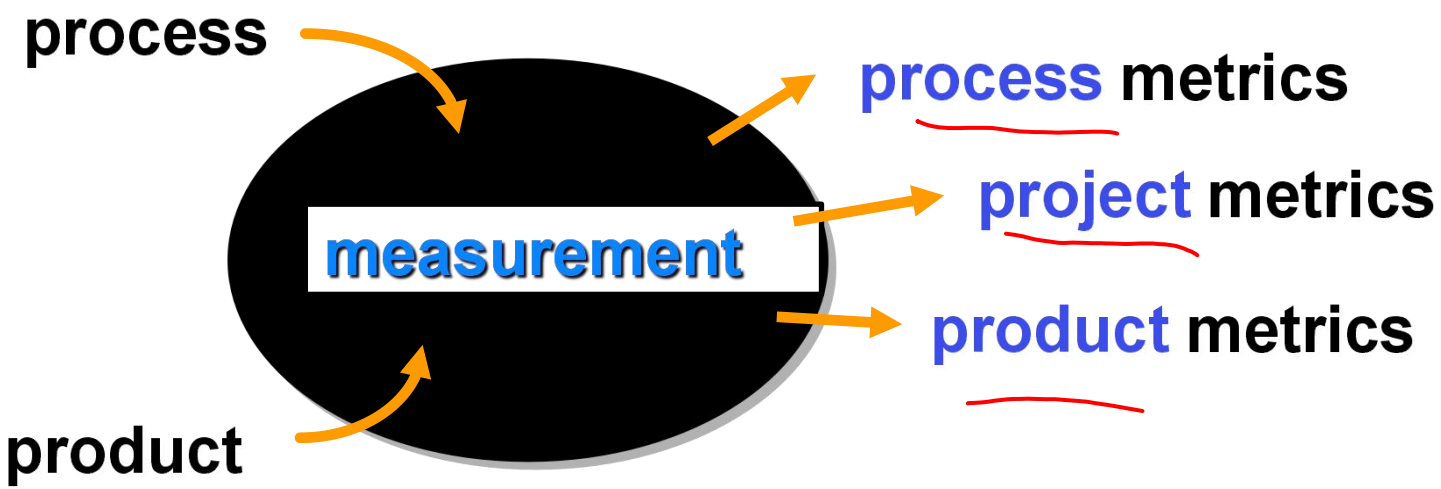
Part 4
Project Management

Chapter 32
Process and Project Metrics

Contents

32.1 A Good Manager Measures

quantitative measures



32.1 Why Do We Measure?

- Assess the status of an ongoing project
- Track potential risks
- Uncover problem areas before they go “critical”
- Adjust workflow or tasks
- Evaluate the project team’s ability to control quality of software work products

A+

32.1 A Good Manager Measures

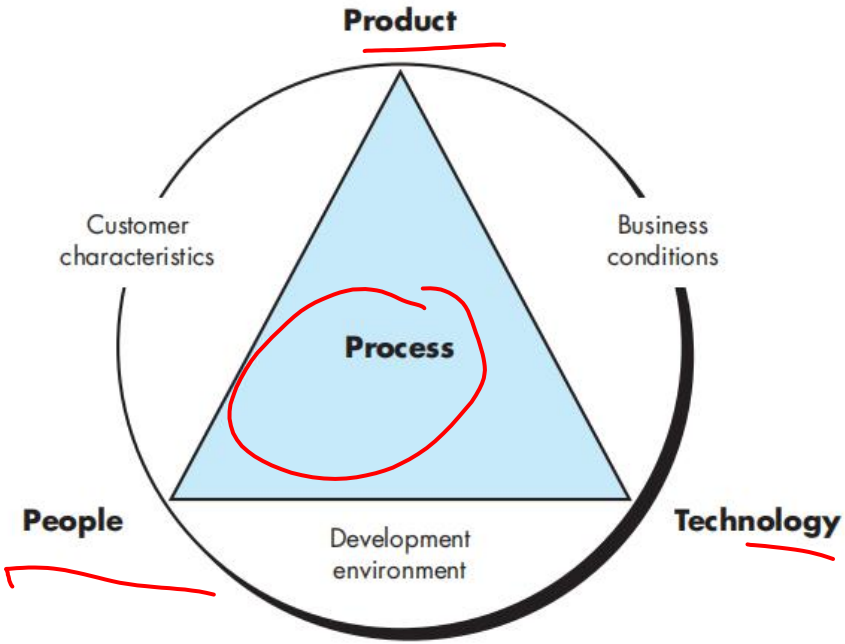
- Measure(noun-度量): provides a quantitative indication of the extent, amount, dimension, capacity, or size of some attribute of a product or process. (the number of errors of a single software component, directly collected).
- Measurement(verb-测量): the act of determining a measure.
- Metric(noun): It relates the individual measures in some way. (e.g the average number of errors found per review)

Measures that are collected by a project team and converted into metrics.

32.1.1 Process Metrics and Software Process Improvement

FIGURE 32.1

Determinants for software quality and organizational effectiveness.
Source: Adapted from [Pau94].



defect
size
effort

Many of the same metrics are used in both the process and project domains.

主观题 10分

设置

How to measure the efficiency of a software process?

正常使用主观题需2.0以上版本雨课堂

作答

32.1.1 Process Metrics

- Quality-related
  - –focus on quality of work products and deliverables
    （ e.g. errors uncovered before release of the software
    defects delivered to and reported by end-users ）
- Productivity-related
  - –Production of work-products related to effort expended
- Statistical SQA data
  - – defect categorization & analysis
- Defect removal efficiency
  - – propagation of errors from process activity to activity
- Reuse data
  - –The number of components produced and their degree of reusability

DRE

32.1.1 Process Metrics

How much code can a coder code per day?

- Fred Brooks claimed in 'The Mythical Man-Month' that programmers working on the OS/360 operating system averaged around 10 LOC per day.
- Capers Jones measured productivity of around 16 to 38 LOC per day across a range of projects.
- McConnell measured productivity of 20 to 125 LOC per day for small projects (10,000 LOC) through to 1.5 to 25 LOC per day for large projects (10,000,000 LOC).

https://successfulsoftware.net/2017/02/10/how-much-code-can-a-coder-code/#:~:text=Capers%20Jones%20measured%20productivity%20of,large%20projects%20(10%2C000%2C000%20LOC).

- solid embedded software : 1-2 LOC per developer-hour

32.1.2 Project Metrics

Why?

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

How?

inputs： the resources (e.g., people, tools)

outputs： work products created in this project.

results： the effectiveness of the deliverables.

32.1.2 Typical Project Metrics

- Effort/time per software engineering task
  - FD/DD: 1 person-months
  - UT/IT: 1 person-months
- Errors uncovered per review hour
- Scheduled vs. actual milestone dates
  - Diff: 10 days
- Changes (number) and their characteristics
- Distribution of effort on SE tasks

50% 20% 30%

32.1.3 Product Metrics (textbook Ch30)

- REQUIREMENTS : Function-Based Metrics, Specification Quality
- DESIGN MODEL： System complexity, Object-Oriented Design (CK Metrics）
- SOURCE CODE: Halstead, LOC(directly)
- TESTING: Halstead Metrics
- MAINTENANCE: Software Maturity Index

Software size: LOC、FP

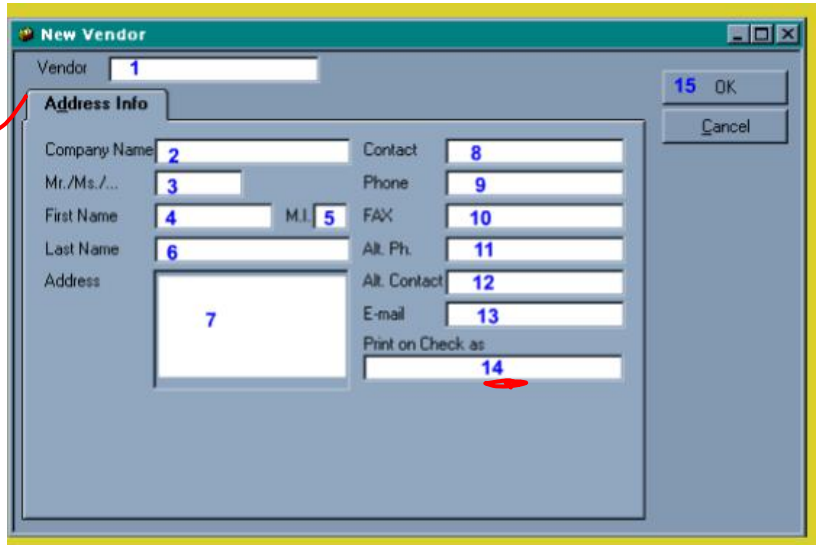
32.1.3 Product Metrics - Why FP?

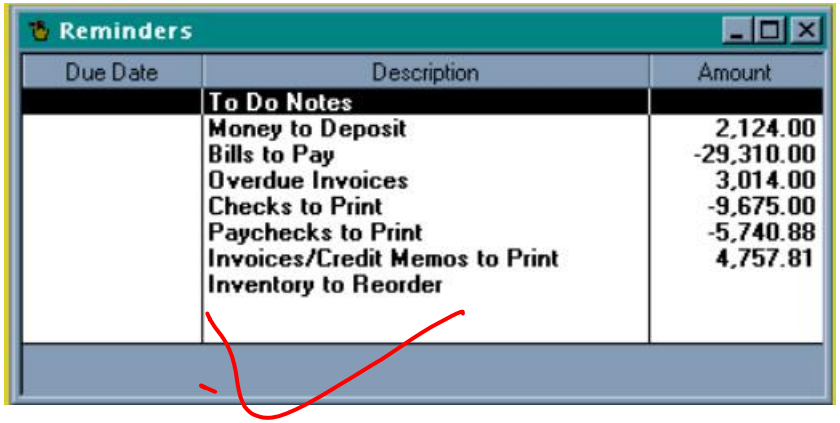
- Programming language independent
- Used readily countable characteristics that are determined early in the software process
- Does not “penalize” inventive (short) implementations that use fewer LOC that other more clumsy versions

32.1.3 Product Metrics - FP

1. Determine the number of components (EI, EO, EQ, ILF, and ELF)

- EI - The number of external inputs. E.g. Input user name
- EO - The number of external output. E.g. Display a list of books
- EQ - The number of external queries.
  Input+compute+output E.g. books queries given a condition
- ILF - The number of internal logic files. E.g. Data file
- ELF - The number of external logic files. E.g. Interface file





https://alvinalexander.com/FunctionPoints/

32.1.3 Product Metrics - FP

2. Compute the Unadjusted Function Point Count (UFC)

1) Rate each component as low, average, or high.

- **For transactions (EI, EO, and EQ), the rating is based on:**
  - FTR - The number of files updated or referenced.
  - DET - The number of user-recognizable fields.
  - Example: an EI that references 2 files and 10 data elements would be ranked as average.

FTR's	DET's		
	1 - 5	6 - 15	> 15
0 - 1	Low	Low	Average
2 - 3	Low	Average	High
> 3	Average	High	High

- **For files (ILF and ELF), the rating is based on is based on:**
  - RET - The number of user-recognizable data elements in an ILF or ELF.
  - DET - The number of user-recognizable fields.
  - Example: an ILF that contains 10 data elements, and 5 fields would be ranked as average

RET's	DET's		
	1 - 5	6 - 15	> 15
1	Low	Low	Average
2 - 5	Low	Average	High
> 5	Average	High	High

2) Convert ratings into UFC's.

Rating	Values				
	EO	EQ	EI	ILF	ELF
Low	4	3	3	7	5
Average	5	4	4	10	7
High	6	5	6	15	10

https://www.csee.umbc.edu/~mgrass2/cmsc645/function_point.html

32.1.3 Product Metrics - FP

UFC

FIGURE 30.1

Computing function points

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

FIGURE 30.3

Computing function points

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

32.1.3 Product Metrics - FP

$$FPC = UFC * VAF$$

$$VAF = 0.65 + [(\sum Di) \times 0.01]$$

Di: 14 General System Characteristics (GSC), **Di:[0, 5], different influence**

Sl. No	Degree of Influence	Value (0-5)	Comments
1	Data Communications	0	
2	Distributed Data Processing	0	
3	Performance	0	
4	Heavily used configuration	0	
5	Transaction rate	0	
6	Online data entry	0	
7	End-user efficiency	0	
8	Online update	0	
9	Complex processing	0	
10	Reusability	0	
11	Installation ease	0	
12	Operational ease	0	
13	Multiple sites	0	
14	Facilitate change	0	
	Total	0	

	Value Adjustment Factor (VAF)	0.65

Value	Characteristic
0	Not Present, No influence
1	Incidental influence
2	Moderate influence
3	Average influence
4	Significant influence
5	Strong influence throughout

https://sourceforge.net/p/functionpoints/wiki/Function%20Point%20Analysis/

32.1.3 Product Metrics - FP

- **4. Convert FPC to lines of source code (SLOC)**

one function point : rough estimates of the average number of lines of code required with various programming languages [QSM02]

Programming Language	LOC per Function Point			
	Avg.	Median	Low	High
Ada	154	—	104	205
ASP	56	50	32	106
Assembler	337	315	91	694
C	148	107	22	704
C++	59	53	20	178
C#	58	59	51	704
COBOL	80	78	8	400
ColdFusion	68	56	52	105
DBase IV	52	—	—	—
Easytrieve+	33	34	25	41
Focus	43	42	32	56
FORTRAN	90	118	35	—
FoxPro	32	35	25	35
HTML	43	42	35	53
Informix	42	31	24	57
J2EE	57	50	50	67
Java	55	53	9	214
JavaScript	54	55	45	63
JSP	59	—	—	—
Lotus Notes	23	21	15	46

Representative values developed by QSM

32.1.3 Product Metrics - FP

- **4. Convert FPC to lines of source code (SLOC)**

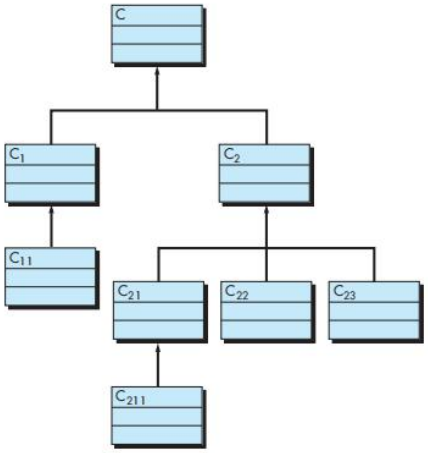
one function point : rough estimates of the average number of lines of code required with various programming languages

Programming Language	LOC per Function Point			
	Avg.	Median	Low	High
Mantis	71	27	22	250
Natural	51	53	34	60
.NET	60	60	60	60
Oracle	42	29	12	217
OracleDev2K	35	30	23	100
PeopleSoft	37	32	34	40
Perl	57	57	45	60
PL/1	58	57	27	92
Powerbuilder	28	22	8	105
RPG II/III	61	49	24	155
SAS	50	35	33	49
Smalltalk	26	19	10	55
SQL	31	37	13	80
VBScript	38	37	29	50
Visual Basic	50	52	14	276

Representative values developed by QSM

32.1.3 Product Metrics - Object-Oriented Metrics

CK



- Number of scenario scripts
- Number of key classes
- Number of support classes
- Average number of support classes per key class
- Number of subsystems

Take a break

Five minutes

32.2 Software Measurement

How to use metrics? Guidelines

- Use common sense and organizational sensitivity when interpreting metrics data.
- Provide regular feedback to the individuals and teams who have worked to 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.

32.3.1 Measuring Quality

- Correctness — the degree to which a program operates according to specification
- Maintainability—the degree to which a program is amenable to change
- Integrity—the degree to which a program is impervious to outside attack
- Usability—the degree to which a program is easy to use

32.3.2 Defect Removal Efficiency

$$DRE = \frac{E}{E + D}$$

$$DRE_i = \frac{E_i}{E_i + E_{i+1}}$$

where:

E is the number of errors found before delivery of the software to the end-user

D is the number of defects found after delivery.

Example: suppose that
- 100 defects were during before delivery
- 2 defects were found after delivery

The DRE would be calculated as: 100 / (100 + 2) = 98%

DRE can also be used within development (different stage)

E_i : the number of errors found in : action i (FD);

E_{i+1} : the number of errors found in : action *i* +1 (DD)

32.3.2 Defect Removal Efficiency

Question:

1. At the conclusion of a project, it has been determined that 30 errors were found during the modeling phase and 12 errors were found during the construction phase that were traceable to errors that were not discovered in the modeling phase. What is the DRE for the modeling phase?

30/(30+12) = 30/42

2. A software team delivers a software increment to end users. The users uncover eight defects during the first month of use. Prior to delivery, the software team found 242 errors during formal technical reviews and all testing tasks. What is the overall DRE for the project after one month's usage?

242/(242+8) =

Summary

- **frequently used software metrics**

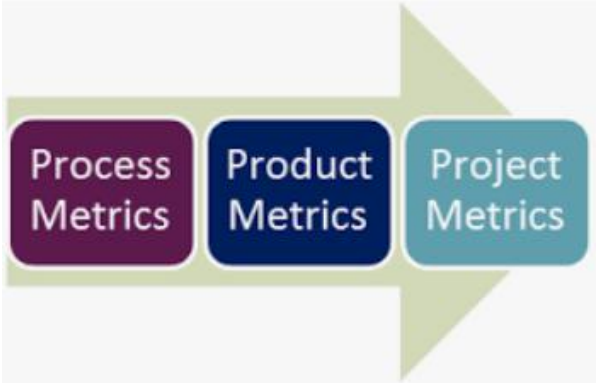
- defects per KLOC/FP ,
- $ per LOC /FP, $ per page of documentation
- pages of documentation per KLOC /FP
- errors per person-month/ per review hour
- errors LOC /FP per person-month

- Number of scenario scripts (use-cases)
- Number of key classes
- Average number of support classes per key class
- Number of subsystems

effort

Summary

- **process metrics**
  Quality-related, Productivity-related,
  Statistical SQA data, DRE, Reuse
- **product metrics**
  LOC, Function-Based Metrics, Specification Quality
  System complexity, CK Metrics, Halstead,
  Software Maturity Index
- **project metrics**
  Effort/time per software engineering task,
  Errors uncovered per review hour,
  Distribution of effort on SE tasks,
  Scheduled vs. actual milestone dates



Many of the same metrics are used in both the process and project.

Assignment5: Deadline: 13 April

32.1. Describe the difference between process and project metrics in your own words.

32.5. Team A found 342 errors during the software engineering process prior to release. Team B found 184 errors. What additional measures would have to be made for projects A and B to determine which of the teams eliminated errors more efficiently? What metrics would you propose to help in making the determination? What historical data might be useful?

32.7. Compute the function point value for a project with the following information domain characteristics:
Number of user inputs: 32
Number of user outputs: 60
Number of user inquiries: 24
Number of fifi les: 8
Number of external interfaces: 2

32.9. The software used to control a photocopier requires 32,000 lines of C and 4,200 lines of Smalltalk. Estimate the number of function points for the software inside the copier.

THE END