Software College Northeastern University

Software Quality Assurance and Testing

Chapter 3 Software Quality Metrics

wuchenni@qq.com

Contents

Chapter 3

Software Quality Metrics

Question

- How big is the program?
  - Huge!
- How close are you to finishing?
  - We are almost there!
- Can you, as a manager, make any useful decisions from such subjective information?
- Need information like, cost, effort, size of project

3.1 Metrics and Software Metrics

- Measure
  - Quantitative indication of the extent, amount, dimension, capacity or size of some attribute of a product or process

- Measurement
  - The act of determining a measure

- Metric
  - A quantitative measure of the degree to which a system, component, or process possesses a given attribute (IEEE Standard Glossary of SE Terms)

3.1 Metrics and Software Metrics

- Indicator
  - An indicator is a metric or combination of metrics that provide insight into the software process, a software project or the product itself.

3.1 Metrics and Software Metrics

- Metrics Characteristics
  - Simple to understand and precisely defined
  - Inexpensive to use
  - Robust
  - Consistent and used over time

3.1 Metrics and Software Metrics

- Why do we measure?
  - To indicate the quality of the product
  - To assess the productivity of the people who produce the product
  - To assess the benefits derived from new software engineering methods and tools
  - To form a baseline for estimation
  - To help justify requests for new tools or additional training

3.1 Metrics and Software Metrics

- Why do we use metrics?

  - when you can measure what you are talking about, and express it in numbers, you know something about it

  - but when you cannot measure it, and when you cannot express it in numbers, your knowledge is of a meagre and unsatisfactory kind

  – Lord Kelvin

3.1 Metrics and Software Metrics

- Why do we use metrics?
  - you cannot control what you cannot measure.
    - DeMarco

3.1 Metrics and Software Metrics

- Why do we use metrics?
  - More accurate
  - Comparable
  - Measurement
  - Measurement is not the goal, the goal is improve the quality of software development through measurement, analysis and feedback

3.1 Metrics and Software Metrics

- 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 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

3.1 Metrics and Software Metrics

- Example Metrics
  - Defect rates
  - Error rates
  - Measured by
    - Individual
    - Module
    - During development
- Errors should be categorized by origin, type, cost

3.1 Metrics and Software Metrics

- A software metric is defined as a unit that enables one to quantitatively determine the extent to which a software process, product, or project possesses a certain attribute.

- Any type of measurement that relates to a software system, process or related artifact.
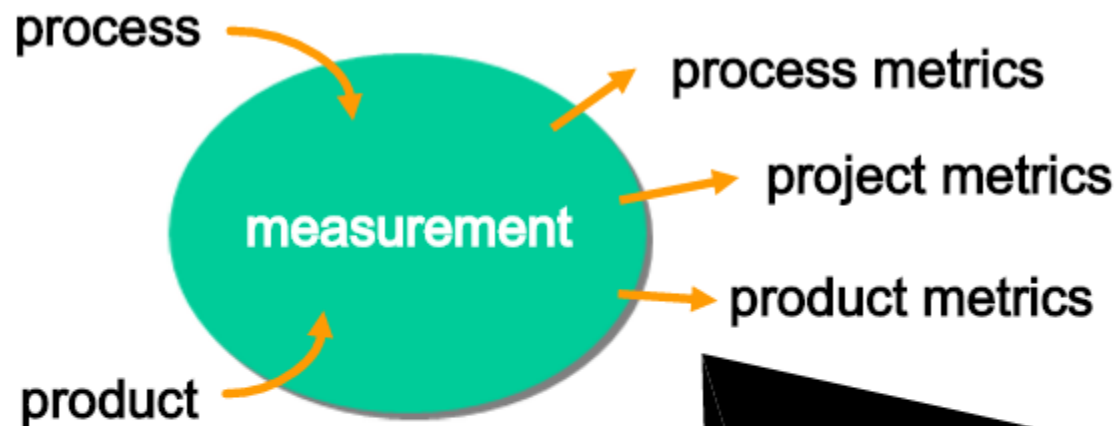
3.1 Metrics and Software Metrics

- Types of metrics
  - Process Metrics
  - Product Metrics
  - Project Metrics

- A Good Manager Measures

process → measurement → process metrics

project metrics

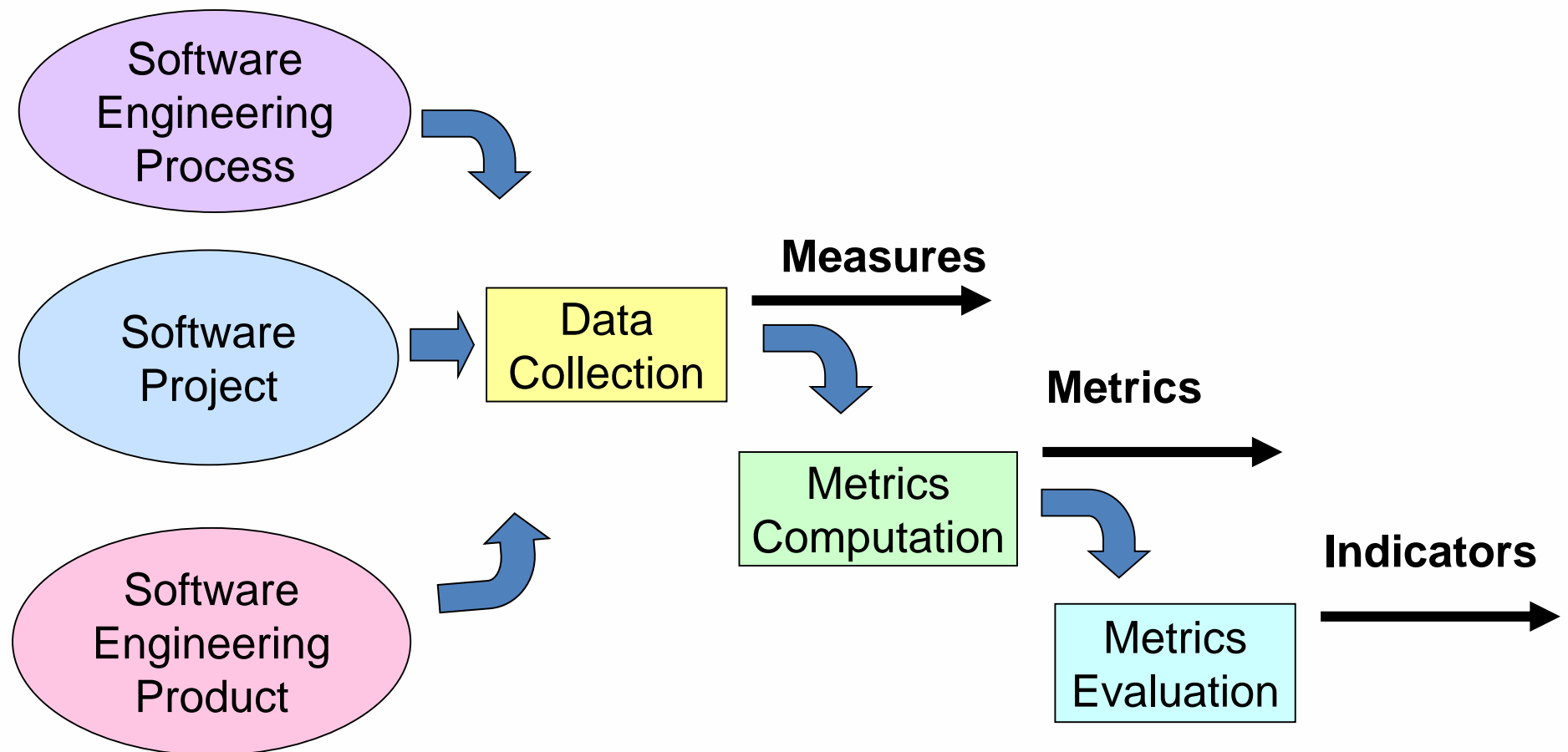product → measurement → product metrics

"*Not everything that can be counted counts, and not everything that counts can be counted .*" - Einstein

What do we use as a basis?
- size?
- function?

- Software Metrics Baseline Process

```
Software
Engineering
Process
        ↓
Software          →   Data          Measures →
Project               Collection
                         ↓
                      Metrics        Metrics →
                      Computation
                         ↓
Software                              Metrics      Indicators →
Engineering                           Evaluation
Product
```

3.1 Metrics and Software Metrics

- Process Metrics
  - Process metrics are measures of the software development process, such as
    - Overall development time
    - Type of methodology used
  - Process metrics are collected across all projects and over long periods of time.
  - Their intent is to provide indicators that lead to long-term software process improvement

3.1 Metrics and Software Metrics

- To improve any process, the rational way is:
  - Measure specific attributes of the process
  - Derive meaningful metrics from these attributes
  - Use these metrics to provide indicators
  - The indicators lead to a strategy for improvement

3.1 Metrics and Software Metrics

- Quantitative Management
  - Both the software process and products are quantitatively understood and controlled.
  - Quantitative Management is not equal to metrics. Metrics is indicators which represent the status of project or organization. On the other hand, quantitative management involves making decision based the metrics as well as historical data.
  - The focus of quantitative management is to make process stable and predictable.

3.1 Metrics and Software Metrics

- Process Metrics
  - Focus on quality achieved as a consequence of a repeatable or managed process. Strategic and long term
  - Statistical Software Process Improvement (SSPI). Error Categorization and Analysis:
    - All errors and defects are categorized by origin
    - The cost to correct each error and defect is recorded
    - The number of errors and defects in each category is computed
    - Data is analyzed to find categories that result in the highest cost to the organization
  - Defect Removal Efficiency (DRE-缺陷排除效率). Relationship between errors (E) and defects (D). The ideal is a DRE of 1
    - DRE= E / (E + D)

3.1 Metrics and Software Metrics

- How to measure effectiveness of a software process?
  - We measure the effectiveness of a software process indirectly
  - We derive a set of metrics based on the outcomes that can be derived from the process
  - Outcomes include
    - 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 etc.
    - Conformance to schedule

3.1 Metrics and Software Metrics

- Project Metrics
  - Project metrics are the measures of software project and are used to monitor and control the project.
  - They enable a software project manager to:
    - Minimize the development time by making the adjustments necessary to avoid delays and potential problems and risks.
    - Assess product quality on an ongoing basis and modify the technical approach to improve quality.

3.1 Metrics and Software Metrics

- Project Metrics
  - Metrics collected from past projects are used as a basis from which effort and time estimates are made for current software project.
  - As a project proceeds, actual values of human effort and calendar time expended are compared to the original estimates.
  - This data is used by the project manager to monitor and control the project.

3.1 Metrics and Software Metrics

- Project Metrics
  - Used by a project manager and software team to adapt project work flow and technical activities.
  - Metrics:
    - Effort or time per SE task
    - Errors uncovered per review hour
    - Scheduled vs. actual milestone dates
    - Number of changes and their characteristics
    - Distribution of effort on SE tasks

3.1 Metrics and Software Metrics

- Product Metrics

  - Product metrics are measures of the software product at any stage of its development, from requirements to installed system.

  - Product metrics may measure:

    - The complexity of the software design

    - The size of the final program

    - The number of pages of documentation produced

3.1 Metrics and Software Metrics

- Types of Software Measurements
  - Direct measures
    - Easy to collect
    - E.g. Cost, effort, lines of codes (LOC), Execution speed, memory size, defects etc.
  - Indirect measures
    - More difficult to assess & can be measured indirectly only.
    - Quality, functionality, complexity, reliability, efficiency, maintainability etc.

3.1 Metrics and Software Metrics

- Example
  - 2 different project teams are working to record errors in a software process
  - Team A
    - Finds 342 errors during software process before release
  - Team B
    - Finds 184 errors
  - Which team do you think is more effective in finding errors?

3.1 Metrics and Software Metrics

- Normalization of Metrics
  - To answer this we need to know the size & complexity of the projects.
  - But if we normalize the measures, it is possible to compare the two
  - For normalization we have 2 ways:
    - Size-oriented metrics
    - Function oriented metrics

3.1 Metrics and Software Metrics

- ## Size-oriented Metrics
  - Based on the "size" of the software produced

Project	Effort (person-month)	Cost ($)	LOC	kLOC	Doc. (pgs)	Errors	People
A	24	168,000	12100	12.1	365	29	3
B	62	440,000	27200	27.2	1224	86	5

3.1 Metrics and Software Metrics

- From the above data, simple size oriented metrics can be developed for each project
  - Errors per KLOC
  - $per KLOC
  - Pages of documentation per KLOC
  - Errors per person-month
  - LOC per person-month
  - Useful for projects with similar environment

3.1 Metrics and Software Metrics

- Advantages of size oriented metrics
  – LOC can be easily counted
  – Many software estimation models use LOC or KLOC as input.

- Disadvantages of size oriented metrics
  – LOC measures are language dependent, programmer dependent.
  – Their use in estimation requires a lot of detail which can be difficult to achieve.

3.1 Metrics and Software Metrics

- Function-oriented metrics
  - Based on "functionality" delivered by the software
  - Functionality is measured indirectly using a measure called function point
  - Function points (FP) are derived using an empirical relationship based on countable measures of software & assessments of software complexity

3.1 Metrics and Software Metrics

- Steps in calculating FP
  - Count the measurement parameters
  - Assess the complexity of the values
  - Calculate the raw FP (see next table)
  - Rate the complexity factors to produce the complexity adjustment value (CAV)
  - Calculate the adjusted FP as follows:
    - FP=raw FP $\times$ (0.65 + 0.01 $\times$ CAV)

- Function Point Metrics

Parameter	Count	Simple	Average	Complex	
Inputs	x	3	4	6	=
Outputs	x	4	5	7	=
Inquiries	x	3	4	6	=
Files	x	7	10	15	=
Interfaces	x	5	7	10	=

Count-total（raw FP）

3.1 Metrics and Software Metrics

- Software information domain values
  - Number of user inputs
  - Number of user outputs
  - Number of user inquiries
  - Number of files
  - Number of external interfaces

3.1 Metrics and Software Metrics

- Rate Complexity Factors
  - For each complexity adjustment factor, give a rating on a scale of 0 to 5
    - 0- No influence
    - 1- Incidental
    - 2- Moderate
    - 3- Average
    - 4- Significant
    - 5- Essential

3.1 Metrics and Software Metrics

- Complexity Adjustment Factors
  - 1. Does the system require reliable backup and recovery?
  - 2. Are data communications required?
  - 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 on-line data entry?
  - 7. Does the on-line data entry require the input transaction to be built over multiple screens or operations?

3.1 Metrics and Software Metrics

- Complexity Adjustment Factors
  - 8. Are the master files updated on-line?
  - 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?

3.1 Metrics and Software Metrics

- Complexity Adjustment Value
  - The rating for all the factors, F1 to F14, are summed to produce the complexity adjustment value (CAV)
  - CAV is then used in the calculation of the function point (FP) of the software

3.1 Metrics and Software Metrics

- Example of Function-Oriented Metrics
  - Errors per FP
  - Defects per FP
  - $ per FP
  - Pages of documentation per FP
  - FP per person month

3.1 Metrics and Software Metrics

- ## FP Characteristics
  - ### Advantages
    - Language independent, based on data known early in project, good for estimation
  - ### Disadvantages
    - Calculation complexity, subjective assessments, FP has no physical meaning (just a number)

3.1 Metrics and Software Metrics

- Example-FP based estimation
  - You are required to establish estimates for virtual class room system. For first increment you have to provide automation of course registration, attendance, lectures.

3.1 Metrics and Software Metrics

- For above mentioned system
  - Student can be registered
  - Courses can be added/ updated
  - Student can view courses
  - Student can inquire his/her registration status

3.1 Metrics and Software Metrics

Parameter	Count		Simple	Average	Complex		
Inputs	**3**	*	**3**	4	6	=	9
Outputs	**4**	*	**4**	5	7	=	16
Inquiries	**1**	*	**3**	4	6	=	3
Files	**4**	*	**7**	10	15	=	28
Interfaces	**0**	*	**5**	7	10	=	0

Count-total（raw FP） = 56

- CAV=29
- FP = raw FP $\times$ (0.65 + 0.01 $\times$ CAV)

  = 56 $\times$ (0.65 + 0.01 $\times$ 29)

  = 52.64 or 53

If organizational average productivity = 5 FP/pm

Labor rate = 8000Rs. per month

Cost = labor rate * Person Month

Based on the FP estimate and the historical productivity data, **the total estimated project cost is 84800Rs, and the estimated effort is 10.6 person-months.**
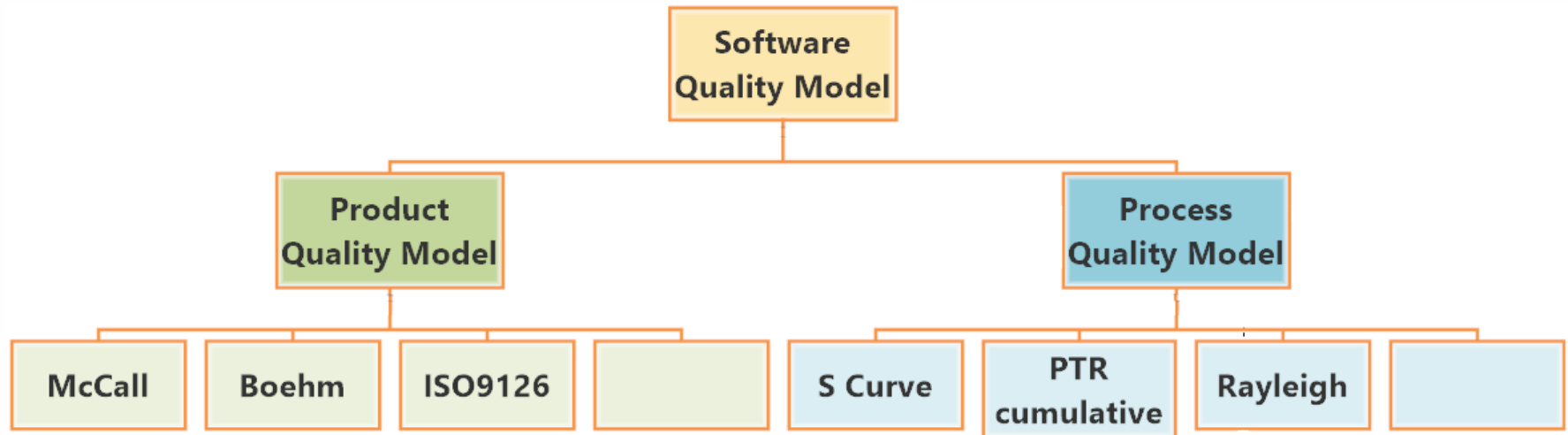
3.2 Software Quality Metrics

- Software quality metrics is a subset of software metrics.

- It focuses on the quality aspects of product, process and project.

3.2 Software Quality Metrics

- Correctness
  - This is the number of defects per KLOC, where a defect is a verified lack of conformance to requirements
  - Defects are those problems reported by a program user after the program is released for general use
- Maintainability
  - This describes the ease with which a program can be corrected if an error is found, adapted if the environment changes, or enhanced if the customer has changed requirements
  - Mean time to change (MTTC) : the time to analyze, design, implement, test, and distribute a change to all users
    - Maintainable programs on average have a lower MTTC