4. Software Process And Project Metrics

Measures, Metrics, And Indicators

- Measurement enables us to gain insight by providing a mechanism for objective evaluation. Measurement can be applied to the software process with the intent of improving it on a continuous basis.

- Measurement can be used throughout a software project to assist in estimation,
  - Quality control,
  - Productivity Assessment,
  - project Control.

- Finally, measurement can be used by software engineers to help assess the quality of technical work products and to assist in tactical decision making as a project proceeds.

Measures, Metrics, And Indicators

- The terms: (are often used interchangeably)
  - *measure*,
  - *measurement*,
  - *Metrics*
- The IEEE Standard Glossary of Software Engineering Terms [IEE93] defines metric as "a quantitative measure of the degree to which a system, component, or process possesses a given attribute.
- For Example: a single data point has been collected (e.g., the number of errors uncovered in the review of a single module), a measure has been established.
- A SE collects measures and develops metrics so that indicators will be obtained.

Metrics In The Process And Project Domains

- **Process indicators :**
  - Enable a SE to gain insight into the efficacy of an existing process,
  - The paradigm, software engineering tasks, work products, and milestones.
  - Process metrics are collected across all projects and over long periods of time.
- **Project indicators:**
  - Enable a software project manager,
  - Assess the status of an ongoing project,
  - Track potential risks,
  - Uncover problem areas before they go "critical,”
  - Adjust work flow or tasks, and
  - Evaluate the project team’s ability to control quality of software work products.

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

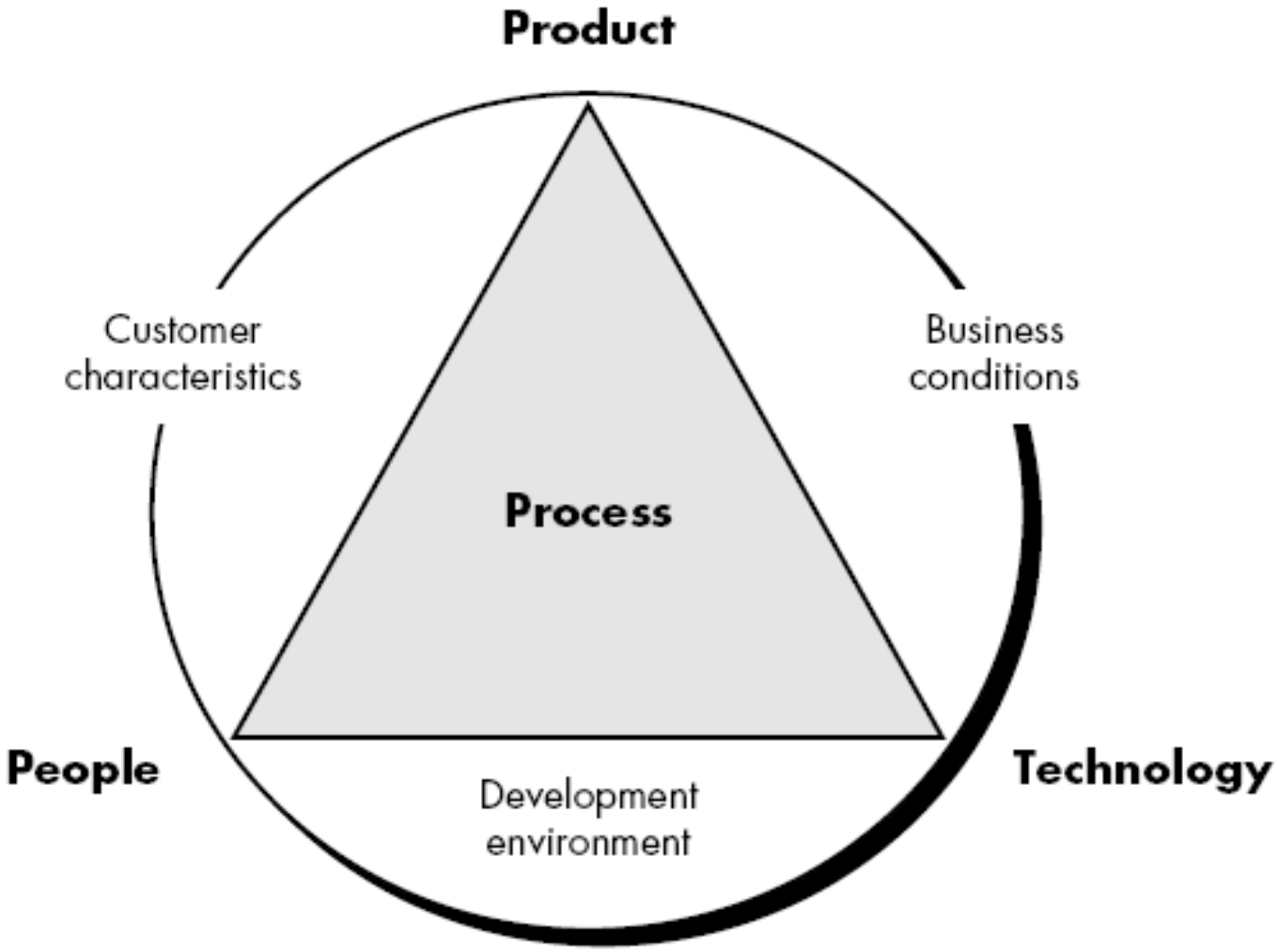


FIGURE 4.2
Causes of defects and their origin for four software projects [GRA94]

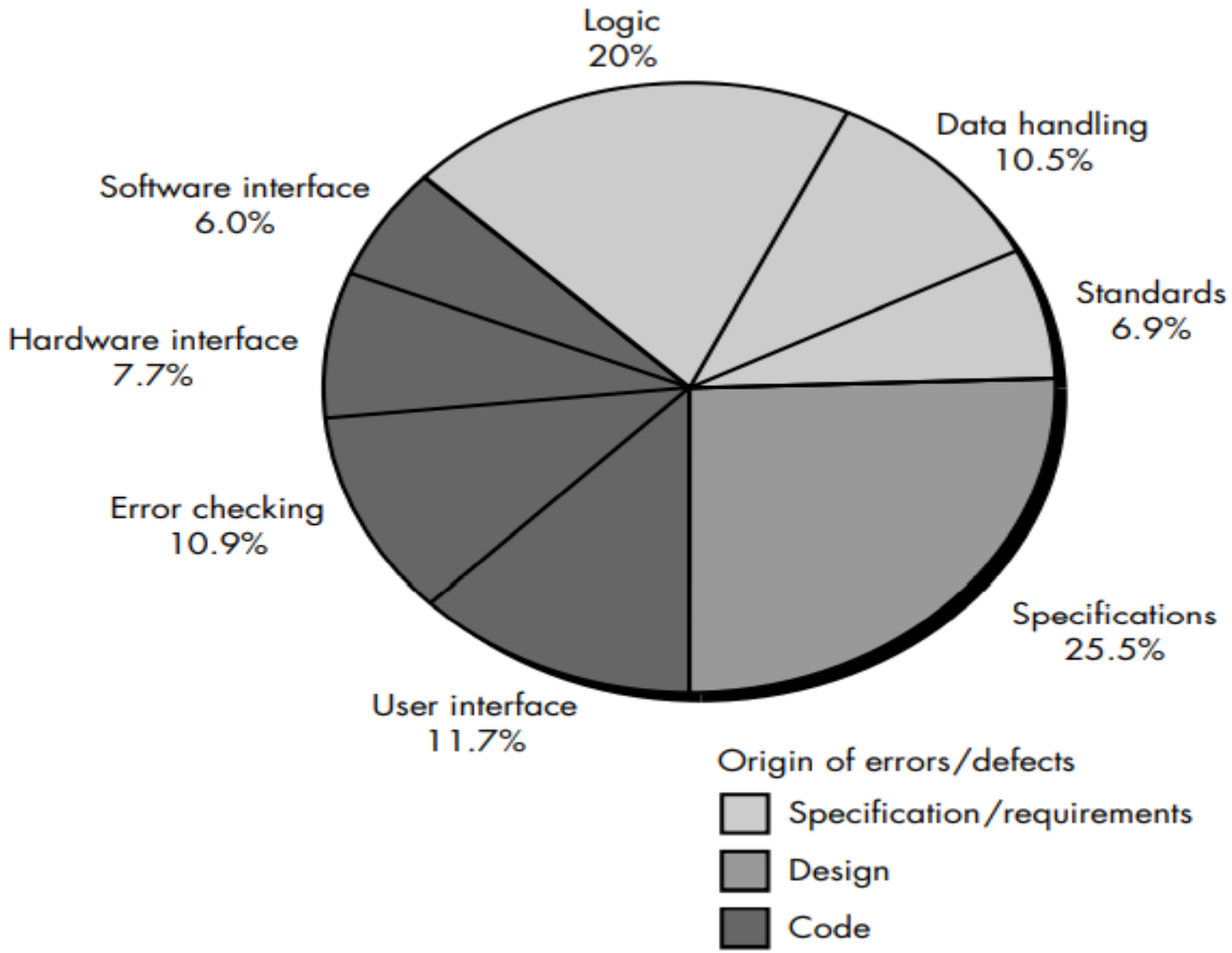
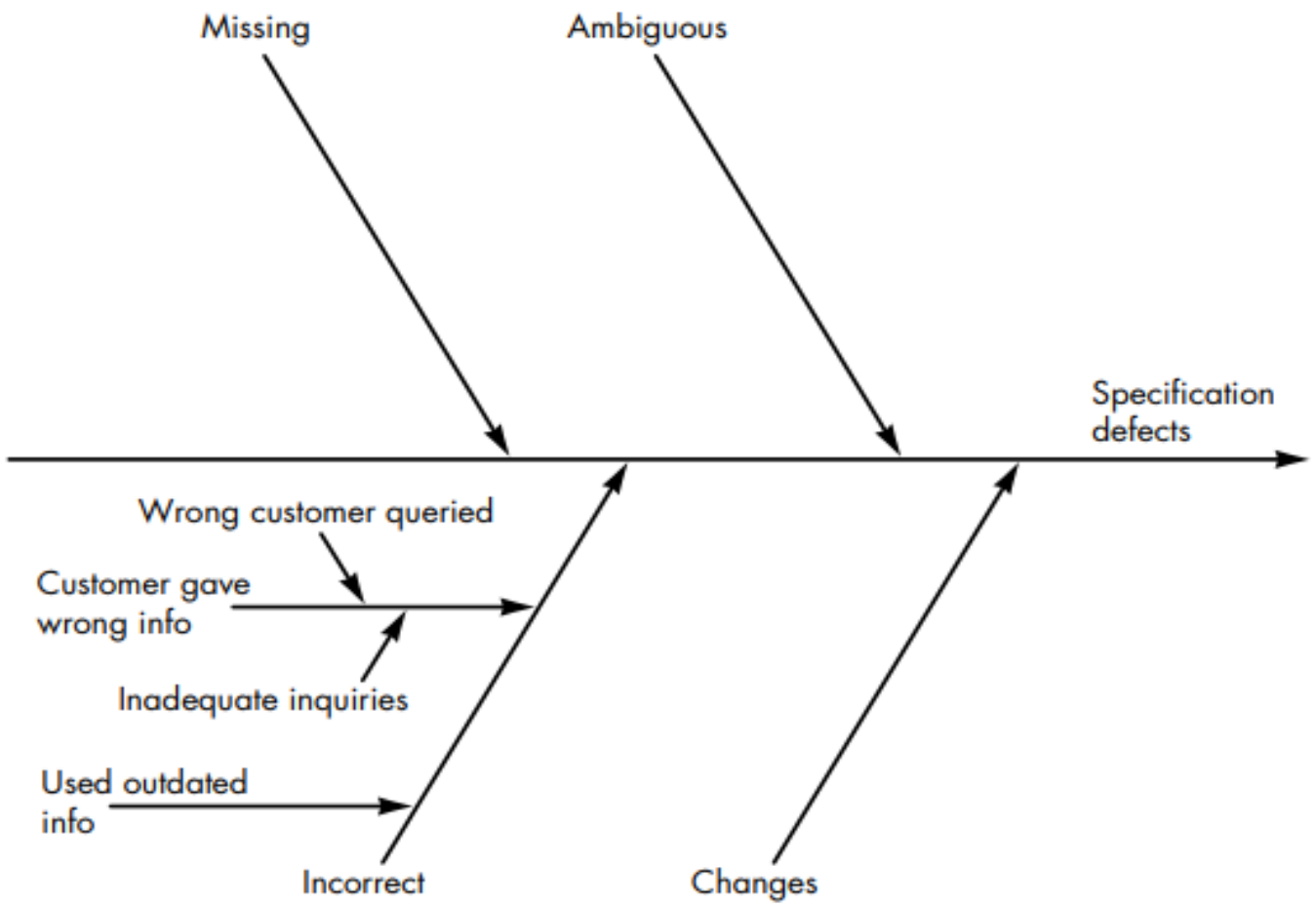


FIGURE 4.3
A fishbone diagram (adapted from [GRA92])



Software Measurement

- In this world , two types of measurement exist,

    1. Direct measurement,
    2. Indirect measurement,

- In the case of SE,

    Direct measure: Cost , product LOC , execution speed, memory size etc.

    Indirect measure: functationalty, QoS, Complexity, efficiency, reliability, maintenance.

- Moreover, direct measure easily collated, but, the indirect measure sometimes difficult such functionality, Software performance etc.

Software Measurement

- For example , we have two teams A and B.

  A found 342 errors

  B found 184 errors.

- What you think, which one is more efficient , A or B?

- **Size– Oriented Metrics (SOM)**
- SOM derived by normalizing **(Quality/Productivity)**, which is based on size of the software.
- SOM maintain by a tabular form such as in figure.

Size–Oriented Metrics (SOM)

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

Function-oriented Metrics

- FOM proposed by Albrecht, called Function Point (FP).
- FP are derived using an empirical relationship (Measure Software Complexity).

Computing function points

			Weighting factor			
Measurement parameter	**Count**		**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

To compute function points (FP), the following relationship is used:

$$FP = \text{count total} \times [0.65 + 0.01 \times \Sigma(F_i)] \quad (4\text{-}1)$$

where count total is the sum of all FP entries obtained from Figure 4.5.

- **Extended Function point Metrics……..? HW**

The F_i (i = 1 to 14) are "complexity adjustment values" based on responses to the following questions [ART85]:

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?
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?

Metrics For Software Quality

- The goal of SE is to produce a high quality System software or application software.
- To achieve the goal, Software Engineer…?
- A good SE uses the measure to obtain the good quality.
- The PM also see the quality (analysis) or collects the error and defects.

An Overview of Factors That Affect Quality

- Mc-Call and Cavano, defined a set of factors to measure the Quality of Software,
  - Product Operation
  - Product Revision
  - Product Transition
- All three, known as Frame Work or SEP.

Measuring Quality

- Gilb, introduced software measuring quality factor,

1. Correctness:

- Program must operate correctly without any errors .
- Defect per KLOC, checked on each year.

2. Maintainability:

- Needs more efforts then other SE activities.
- Mean- time-to-change (MTTC)
- Hitachi used cost oriented metric for maintainability (Spoilage).

Measuring Quality

3. Integrity:

- This is very important measure to investigate system ability.

- Hacker and firewall.

$$\text{integrity} = \text{summation}\ [(1 - \text{threat}) \times (1 - \text{security})]$$

4. Usability

Integrating Metrics Within The Software Process

- Last class I have discussed about?, Majority of software developers don’t use measure schemes and having little desire to begin (cultural).
- Why do we need this ?
- Ask a harried PM?
- I do not see the point (PM),
  - Realistically, applying a wide software metrics program is a tough job?

Arguments for Software Metrics

- Why, it is so important to measure the process of Software Engineering and Product that it produced.
- If do not (Stop Improving), eventually, we are Lost.
- Mostly, Senior PM looks the productivity and quality measure (To achieve the Goal).
- Software (Development/Requirement) is strategic business issues for many Organization, but it mundane by SE as well as PM.

Arguments for Software Metrics

- In the tranches , Software metrics provides immediate benefits, Such as,
- which user requirements are most likely to change?
- Which components in this system are most error prone?
- How much testing should be planned for each component?
- How many errors (of specific types) can I expect when testing commences?

Establishing a Baseline

Introducing a baseline metric, advantage can be obtain at the Process, Project, and Product (technical) level respectively.

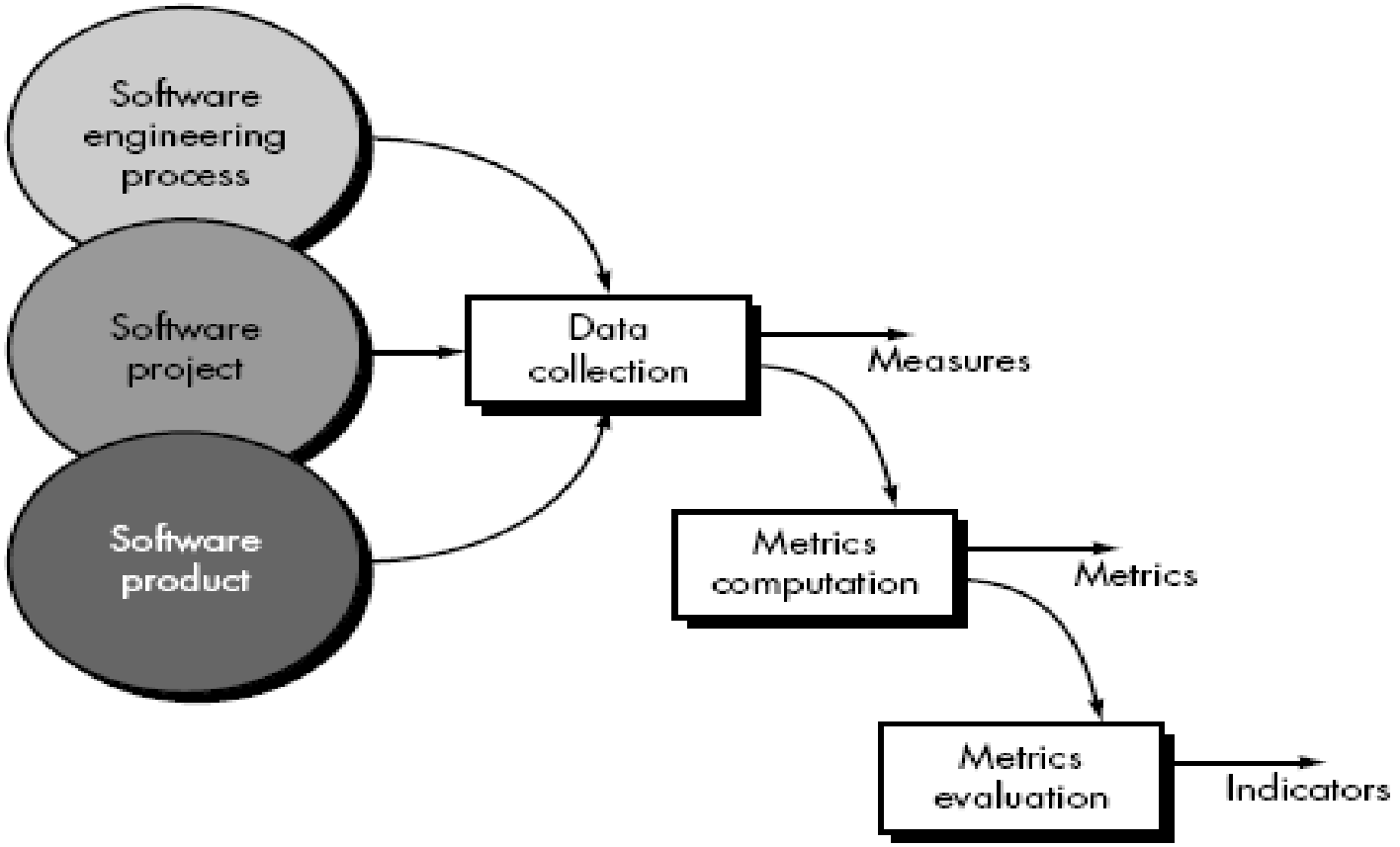
The example of base level metric in your mind……..?

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

Establishing a Baseline

Software metrics collection process



Managing Variation: Statistical Process Control

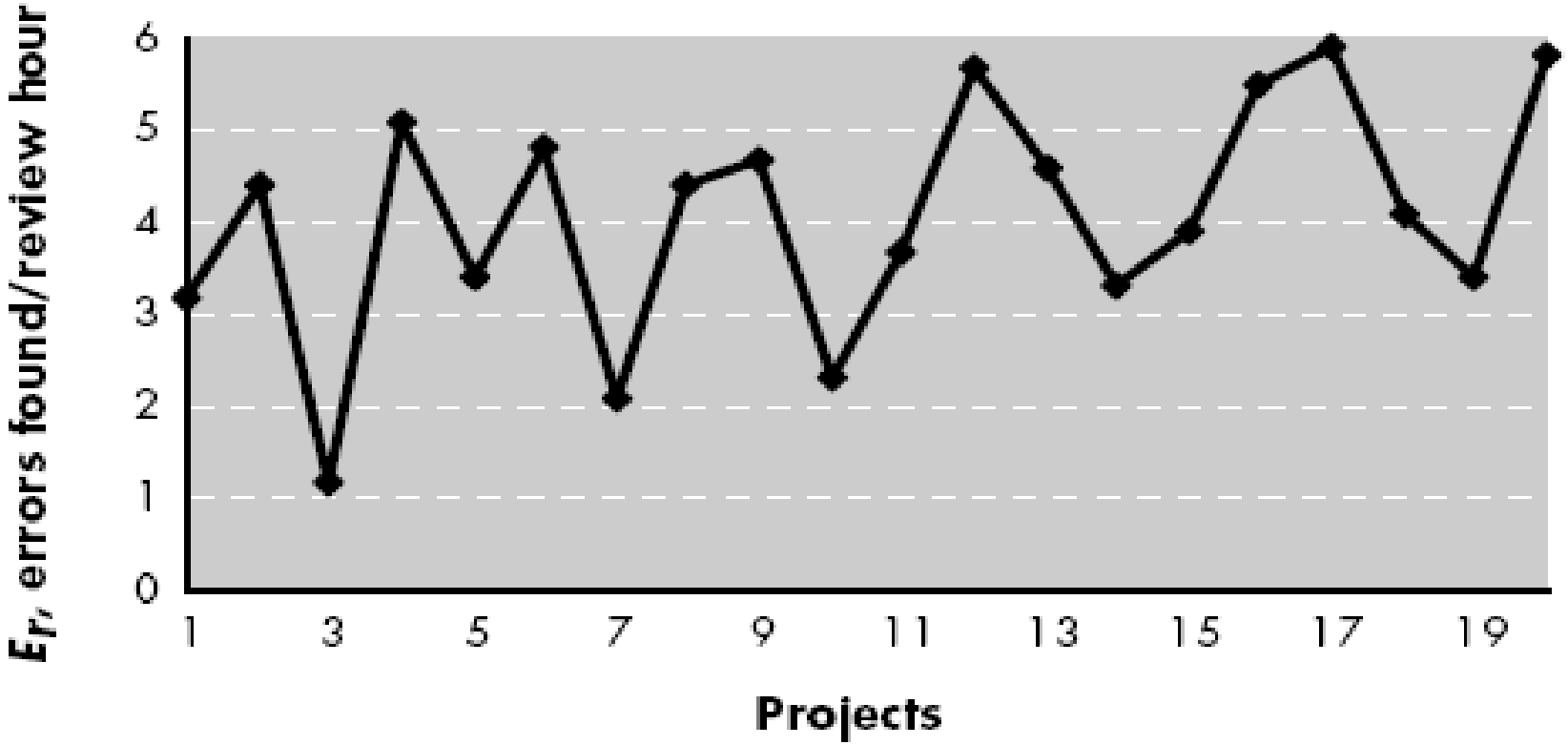
- Software process and product influenced by various parameters…..?
- However, metrics collected for one project and other project, even the same metrics varies.
- How we know about the changes ?, such as
  - What, we are looking at a statistically valid trend or
  - simply a result of statistical noise?
- A graphical technique is available known as Control chart, which was developed by walter shewart in 1920.

Managing Variation: Statistical Process Control

- To illustrate the control chart approach consider a software organization that obtain the process metric.
- Error uncovered per review hour E_r past 15 months, organization has collected E_r = 20 for small project.
- Figure shows the E_r varies from low of 1.2 for project 3 to a high of 5.9 for project 17 .
- In an effort to improve the effectiveness of reviews, the software organization provided training and mentoring to all project team members beginning with project 11.

Managing Variation: Statistical Process Control

Metrics data for errors uncovered per review hour



Metrics For Small Organizations

Establishing A Software Metrics Program