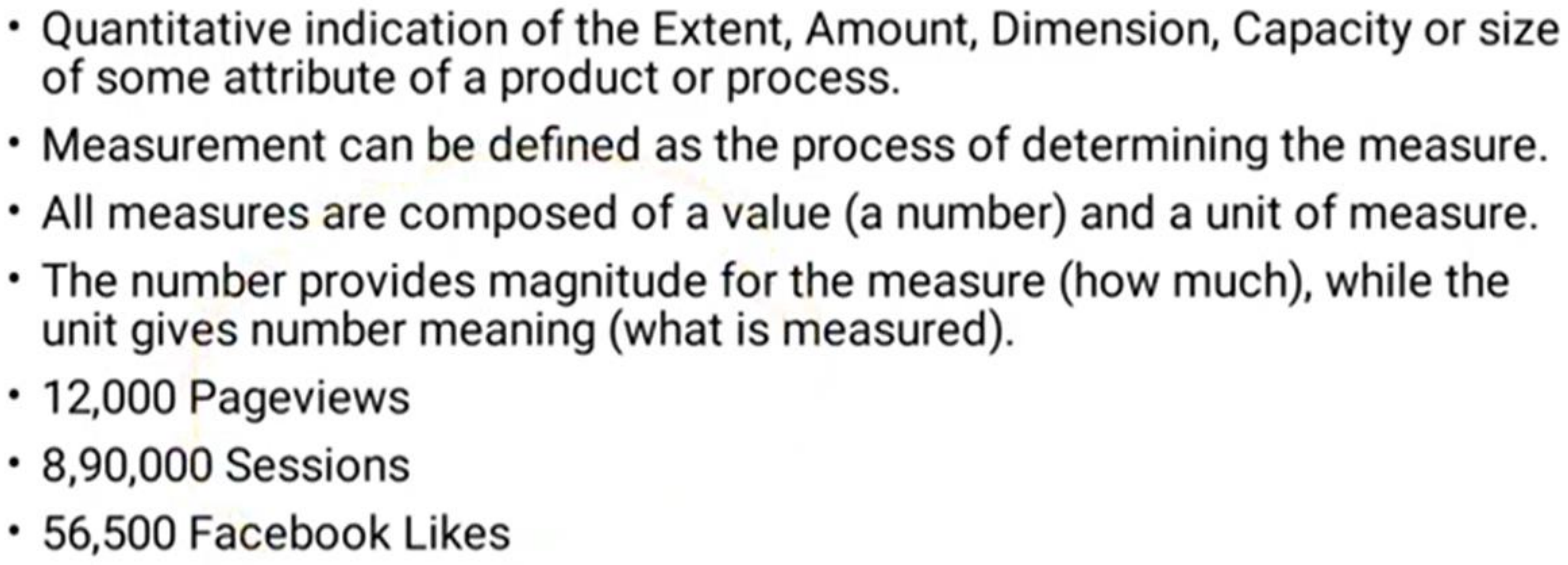
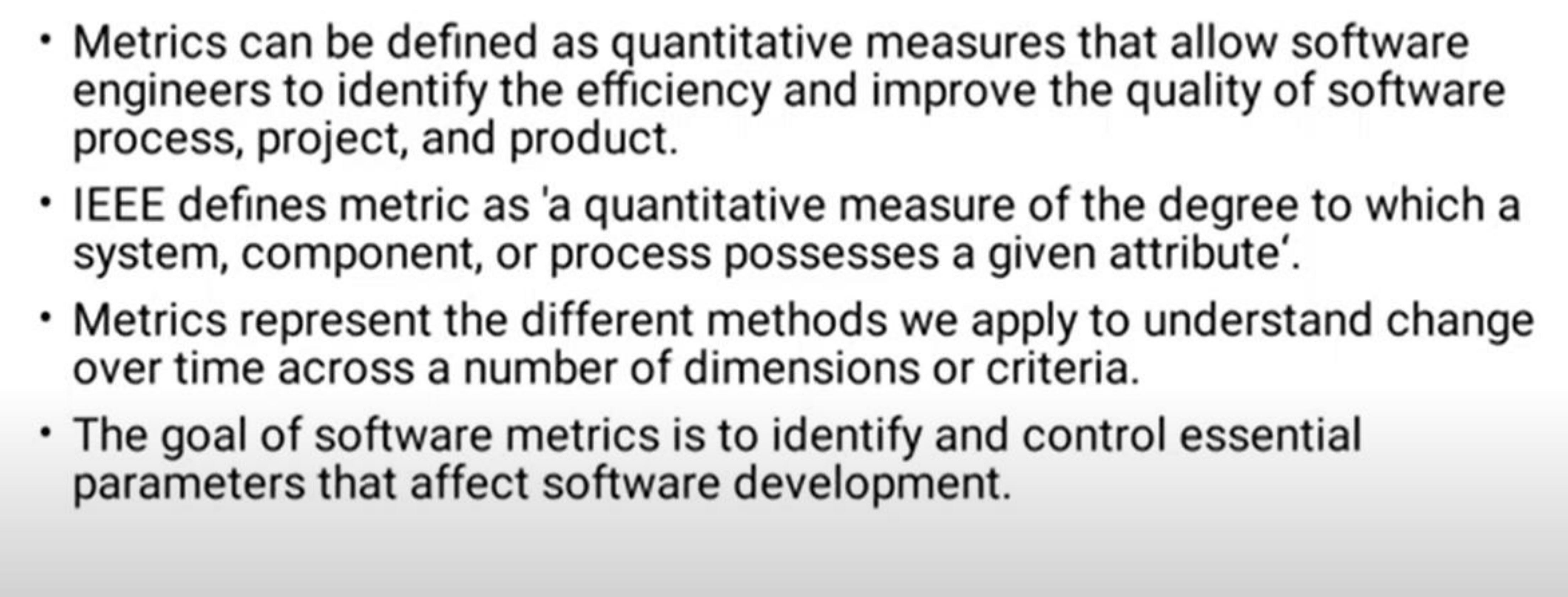
Measures metrics and Indicator

Measures

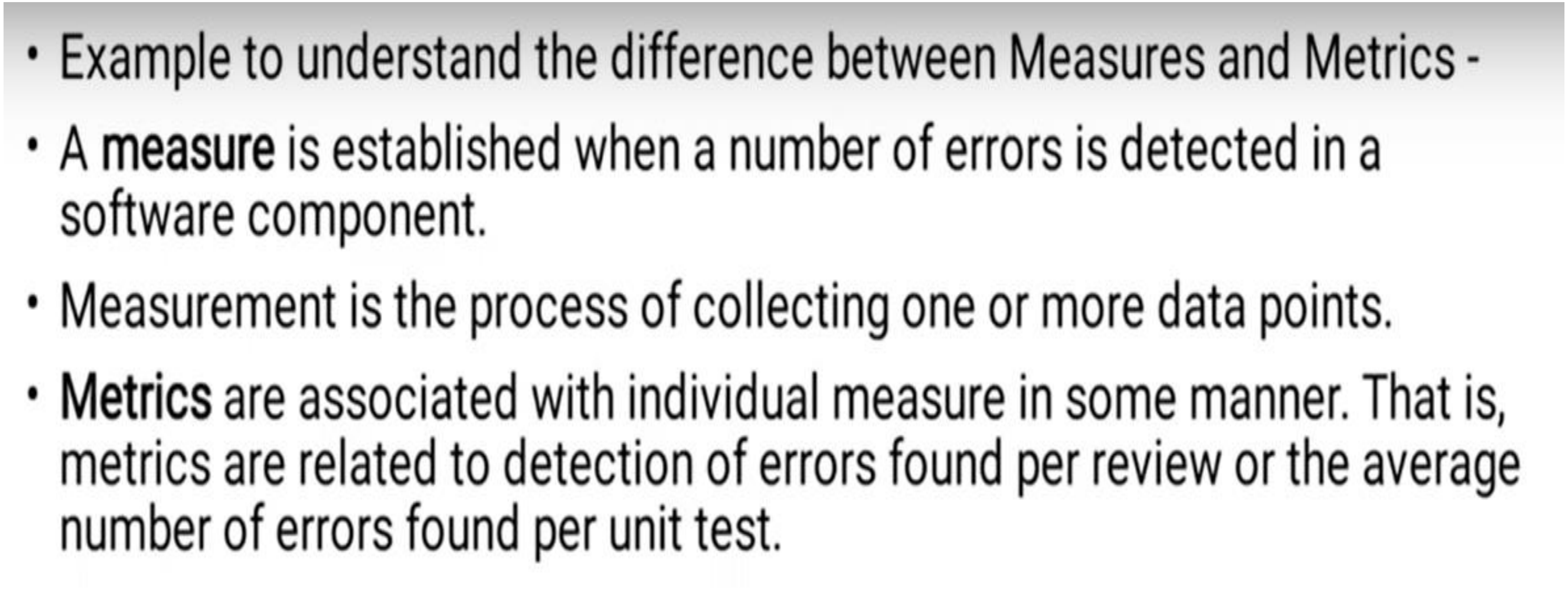


- Quantitative indication of the Extent, Amount, Dimension, Capacity or size of some attribute of a product or process.
- Measurement can be defined as the process of determining the measure.
- All measures are composed of a value (a number) and a unit of measure.
- The number provides magnitude for the measure (how much), while the unit gives number meaning (what is measured).
- 12,000 Pageviews
- 8,90,000 Sessions
- 56,500 Facebook Likes

Metrics

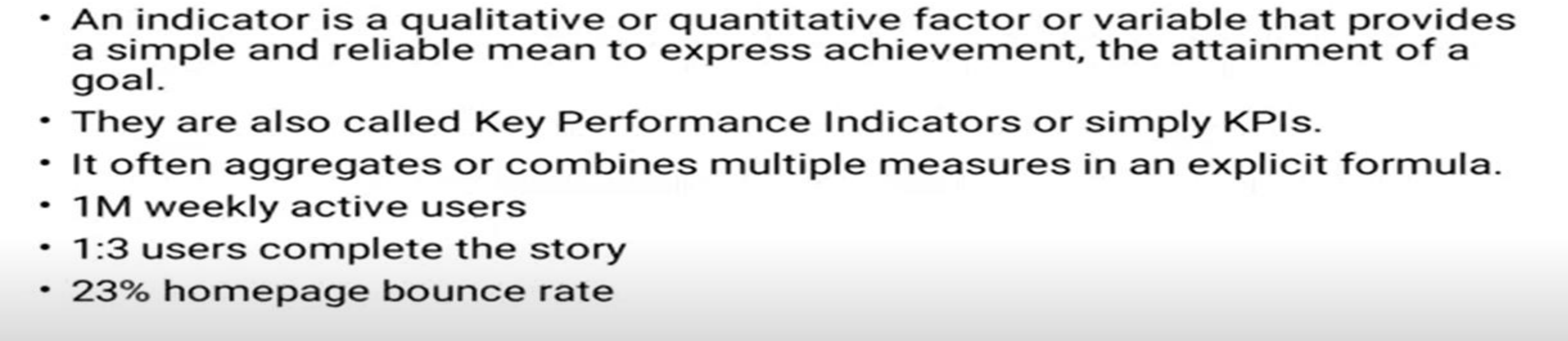


- Metrics can be defined as quantitative measures that allow software engineers to identify the efficiency and improve the quality of software process, project, and product.
- IEEE defines metric as 'a quantitative measure of the degree to which a system, component, or process possesses a given attribute‘.
- Metrics represent the different methods we apply to understand change over time across a number of dimensions or criteria.
- The goal of software metrics is to identify and control essential parameters that affect software development.

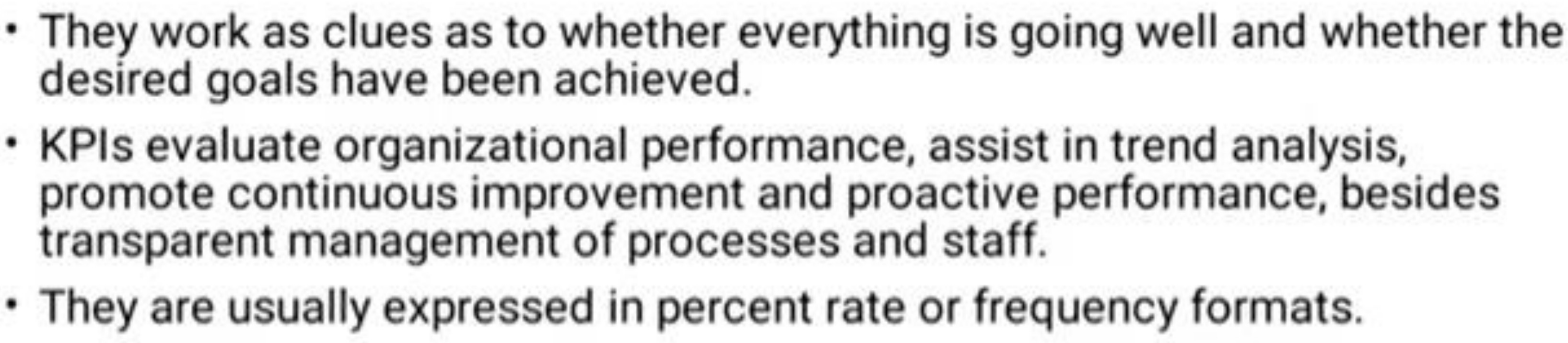


- Example to understand the difference between Measures and Metrics -
- A **measure** is established when a number of errors is detected in a software component.
- Measurement is the process of collecting one or more data points.
- **Metrics** are associated with individual measure in some manner. That is, metrics are related to detection of errors found per review or the average number of errors found per unit test.

Indicator



- An indicator is a qualitative or quantitative factor or variable that provides a simple and reliable mean to express achievement, the attainment of a goal.
- They are also called Key Performance Indicators or simply KPIs.
- It often aggregates or combines multiple measures in an explicit formula.
- 1M weekly active users
- 1:3 users complete the story
- 23% homepage bounce rate



- They work as clues as to whether everything is going well and whether the desired goals have been achieved.
- KPIs evaluate organizational performance, assist in trend analysis, promote continuous improvement and proactive performance, besides transparent management of processes and staff.
- They are usually expressed in percent rate or frequency formats.

What is Software Quality?

- Quality software is reasonably bug or defect free, delivered on time and within budget, meets requirements and/or expectations, and is maintainable.
- Key aspects of quality for the customer include:

> Good design – looks and style

> Good functionality – it does the job well

> Reliable – acceptable level of breakdowns or failure

> Consistency

> Durable – lasts as long as it should

> Good after sales service

> Value for money

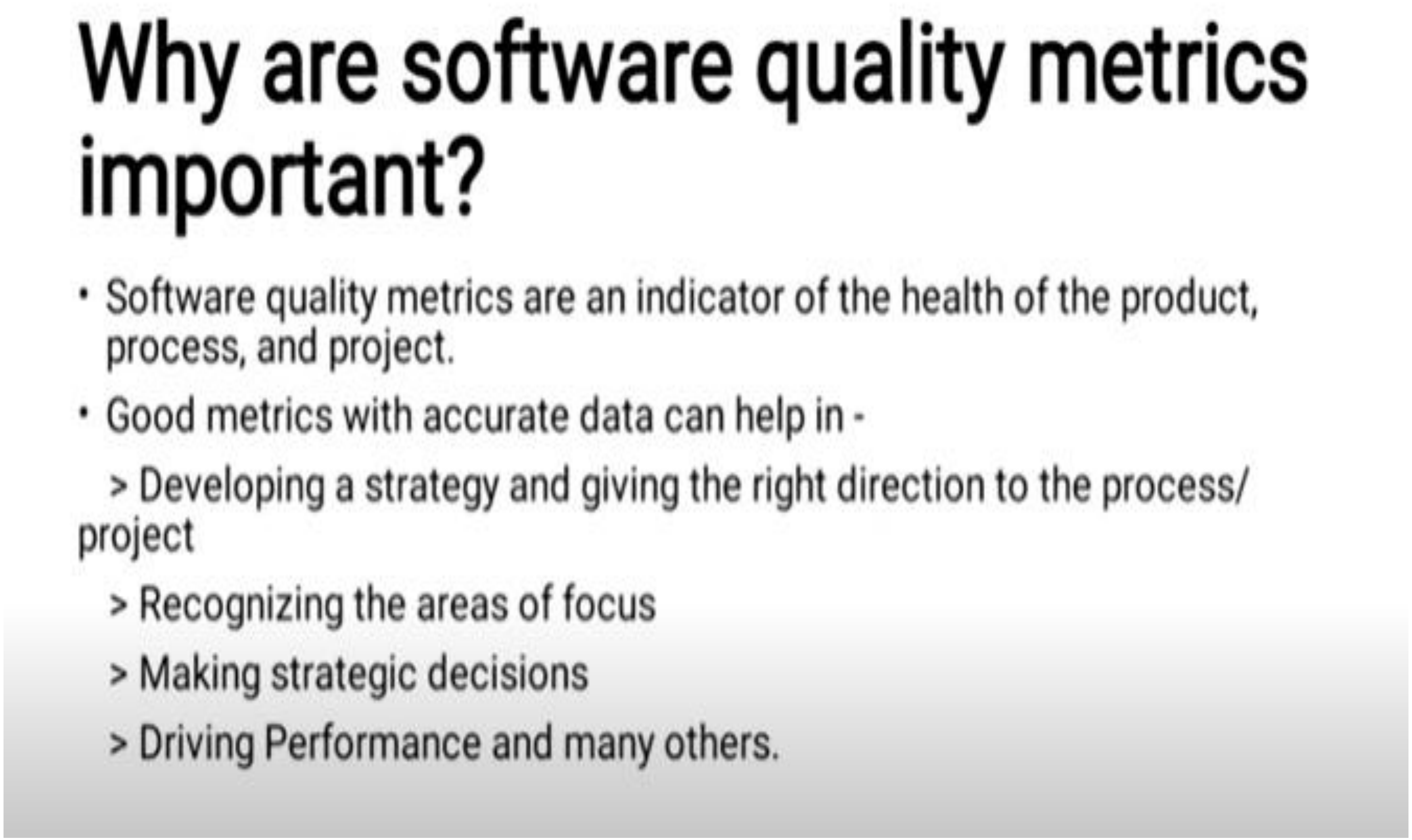
What is Software Quality Metrics?

- In any software project, you can go on building the code but at some point, you need to take a break and check if the work you are doing is right, if the process you followed is correct and so on.
- Metrics help you in exactly that.
- Metrics are pointers or numbers which help you understand-

> the attributes of a product, (like its complexity, its size, it's quality, etc.)

> the attributes of the process (which can be used to improve the quality and speed of development)

> the attributes of the project (which includes the number of resources, costs, productivity and timeline among others), popularly known as the three P's.



Why are software quality metrics important?

- Software quality metrics are an indicator of the health of the product, process, and project.
- Good metrics with accurate data can help in -

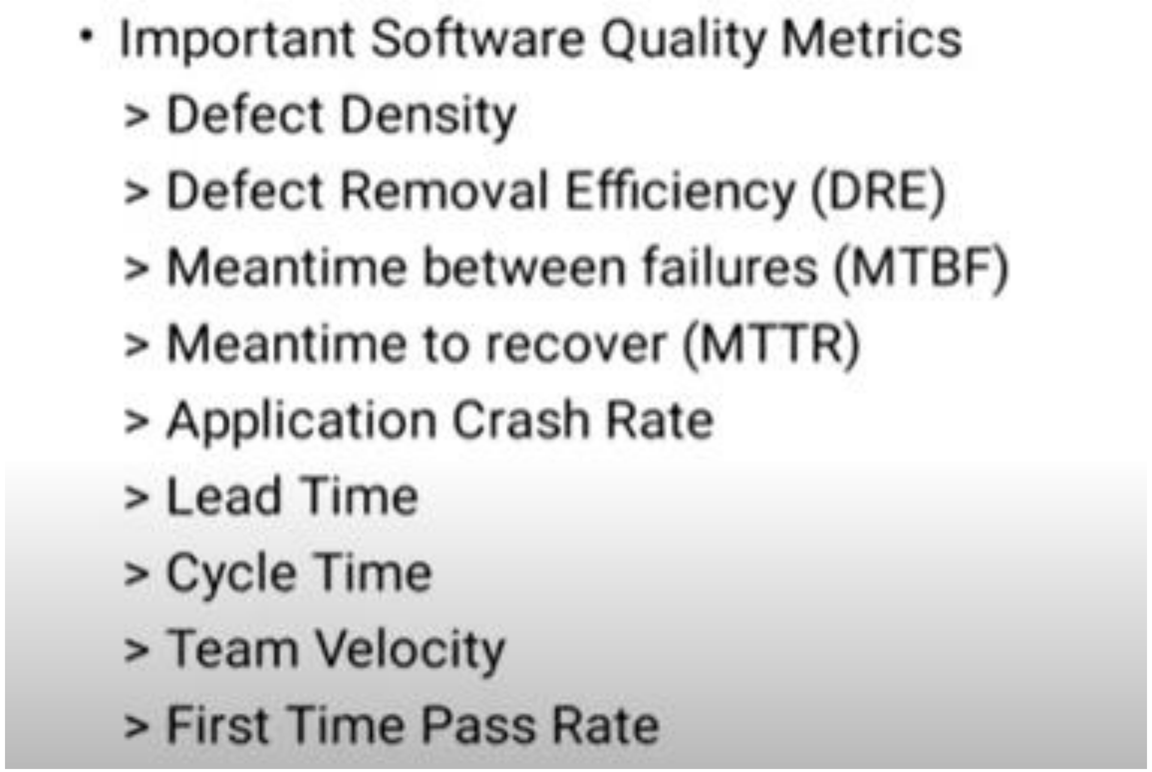
> Developing a strategy and giving the right direction to the process/ project

> Recognizing the areas of focus

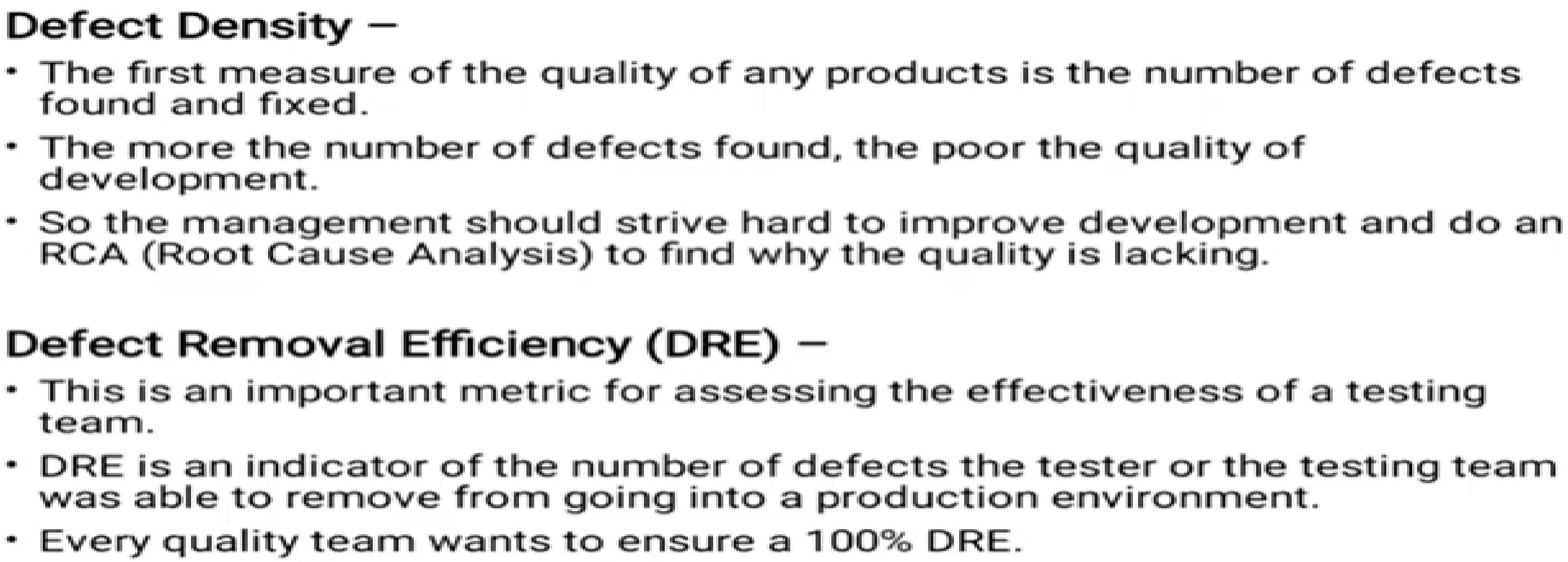
> Making strategic decisions

> Driving Performance and many others.

Software Quality metrics



- Important Software Quality Metrics
  - > Defect Density
  - > Defect Removal Efficiency (DRE)
  - > Meantime between failures (MTBF)
  - > Meantime to recover (MTTR)
  - > Application Crash Rate
  - > Lead Time
  - > Cycle Time
  - > Team Velocity
  - > First Time Pass Rate

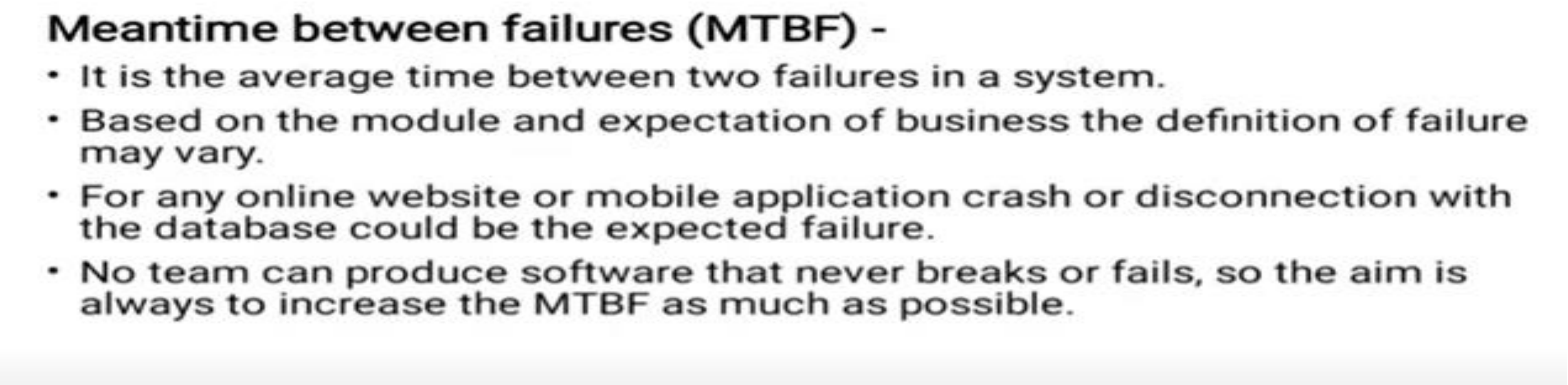


Defect Density –

- The first measure of the quality of any products is the number of defects found and fixed.
- The more the number of defects found, the poor the quality of development.
- So the management should strive hard to improve development and do an RCA (Root Cause Analysis) to find why the quality is lacking.

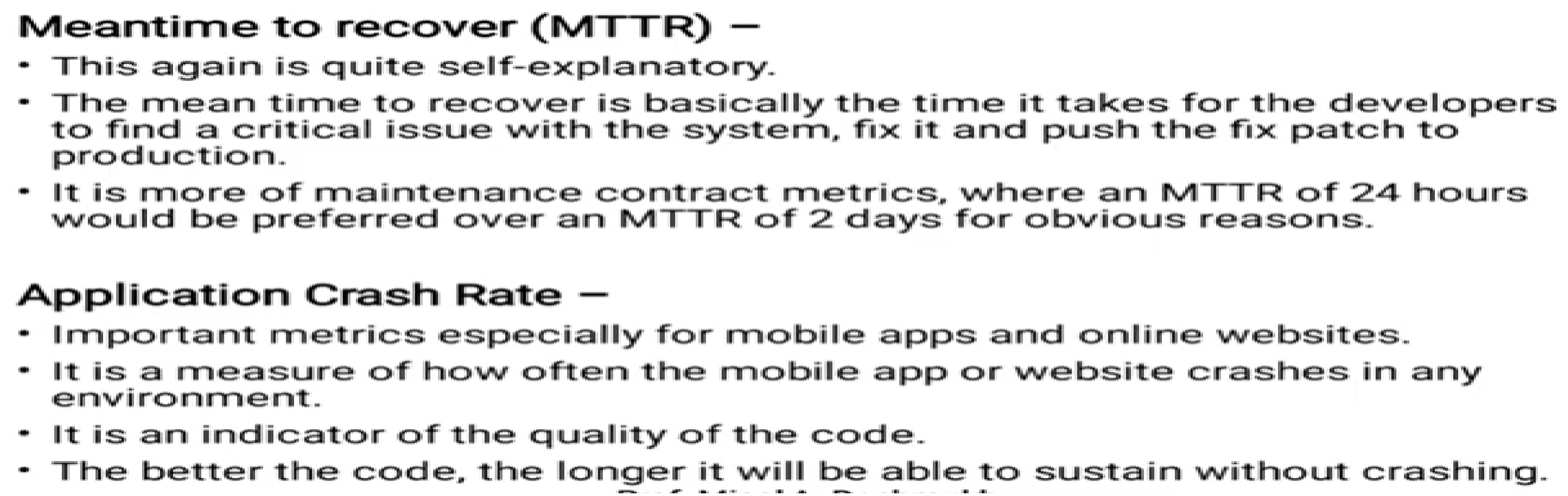
Defect Removal Efficiency (DRE) –

- This is an important metric for assessing the effectiveness of a testing team.
- DRE is an indicator of the number of defects the tester or the testing team was able to remove from going into a production environment.
- Every quality team wants to ensure a 100% DRE.



Meantime between failures (MTBF) -

- It is the average time between two failures in a system.
- Based on the module and expectation of business the definition of failure may vary.
- For any online website or mobile application crash or disconnection with the database could be the expected failure.
- No team can produce software that never breaks or fails, so the aim is always to increase the MTBF as much as possible.

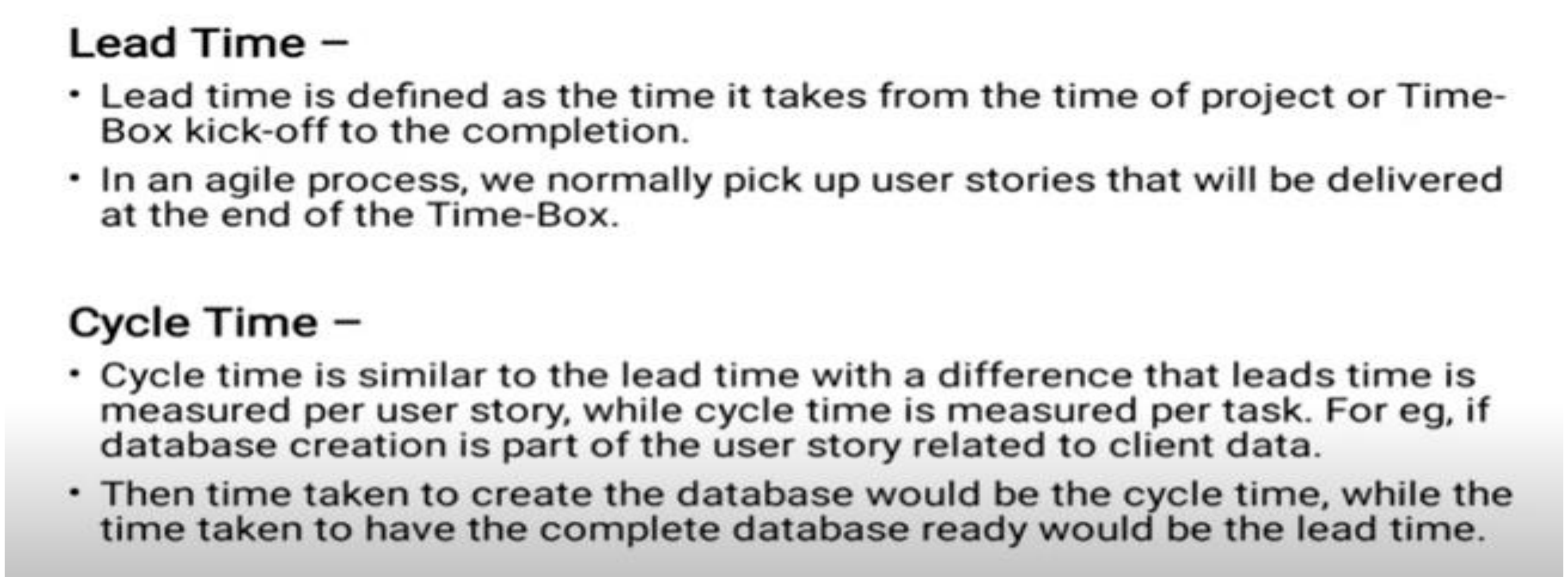


Meantime to recover (MTTR) –

- This again is quite self-explanatory.
- The mean time to recover is basically the time it takes for the developers to find a critical issue with the system, fix it and push the fix patch to production.
- It is more of maintenance contract metrics, where an MTTR of 24 hours would be preferred over an MTTR of 2 days for obvious reasons.

Application Crash Rate –

- Important metrics especially for mobile apps and online websites.
- It is a measure of how often the mobile app or website crashes in any environment.
- It is an indicator of the quality of the code.
- The better the code, the longer it will be able to sustain without crashing.

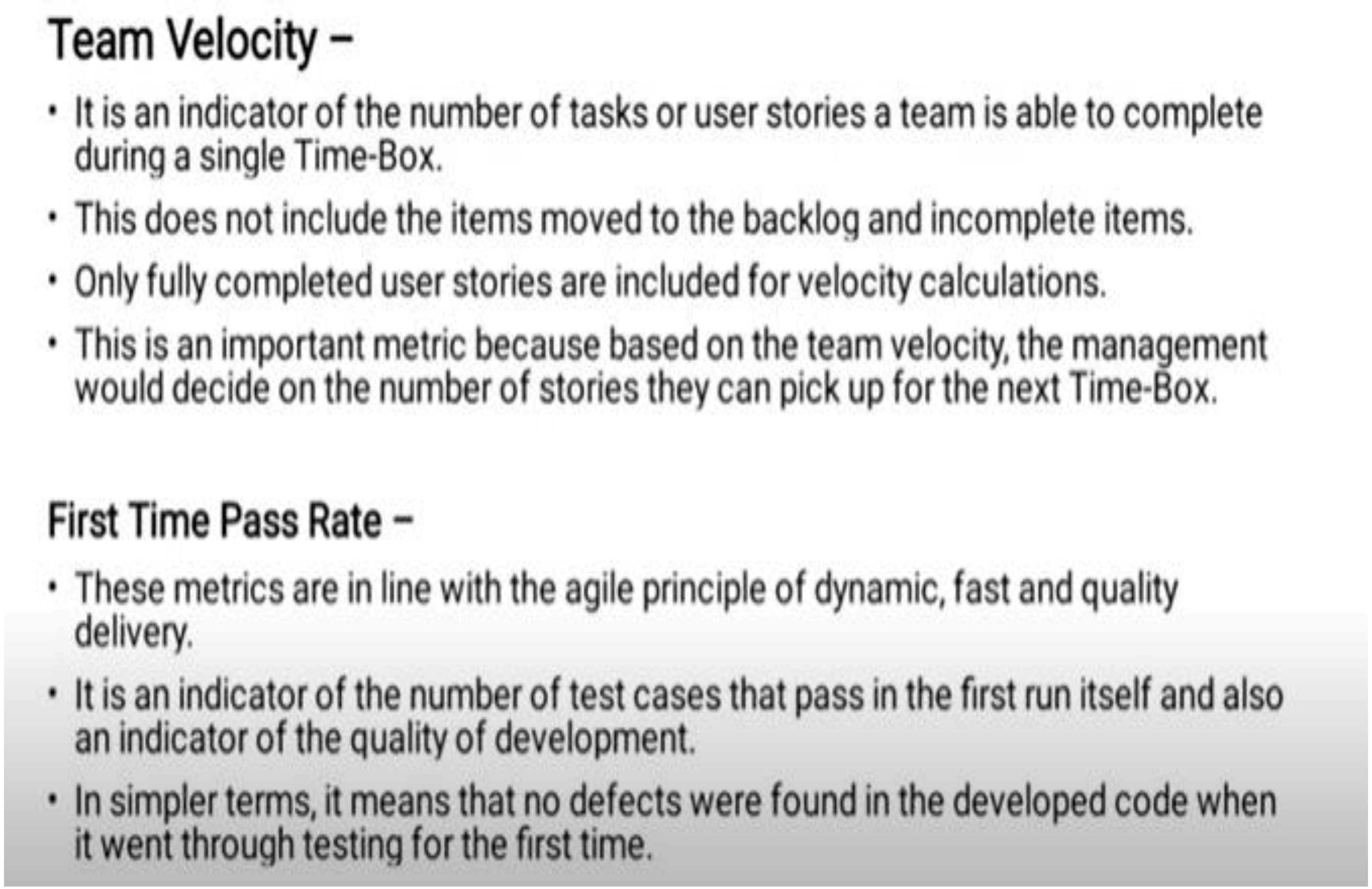


Lead Time –

- Lead time is defined as the time it takes from the time of project or Time-Box kick-off to the completion.
- In an agile process, we normally pick up user stories that will be delivered at the end of the Time-Box.

Cycle Time –

- Cycle time is similar to the lead time with a difference that leads time is measured per user story, while cycle time is measured per task. For eg, if database creation is part of the user story related to client data.
- Then time taken to create the database would be the cycle time, while the time taken to have the complete database ready would be the lead time.



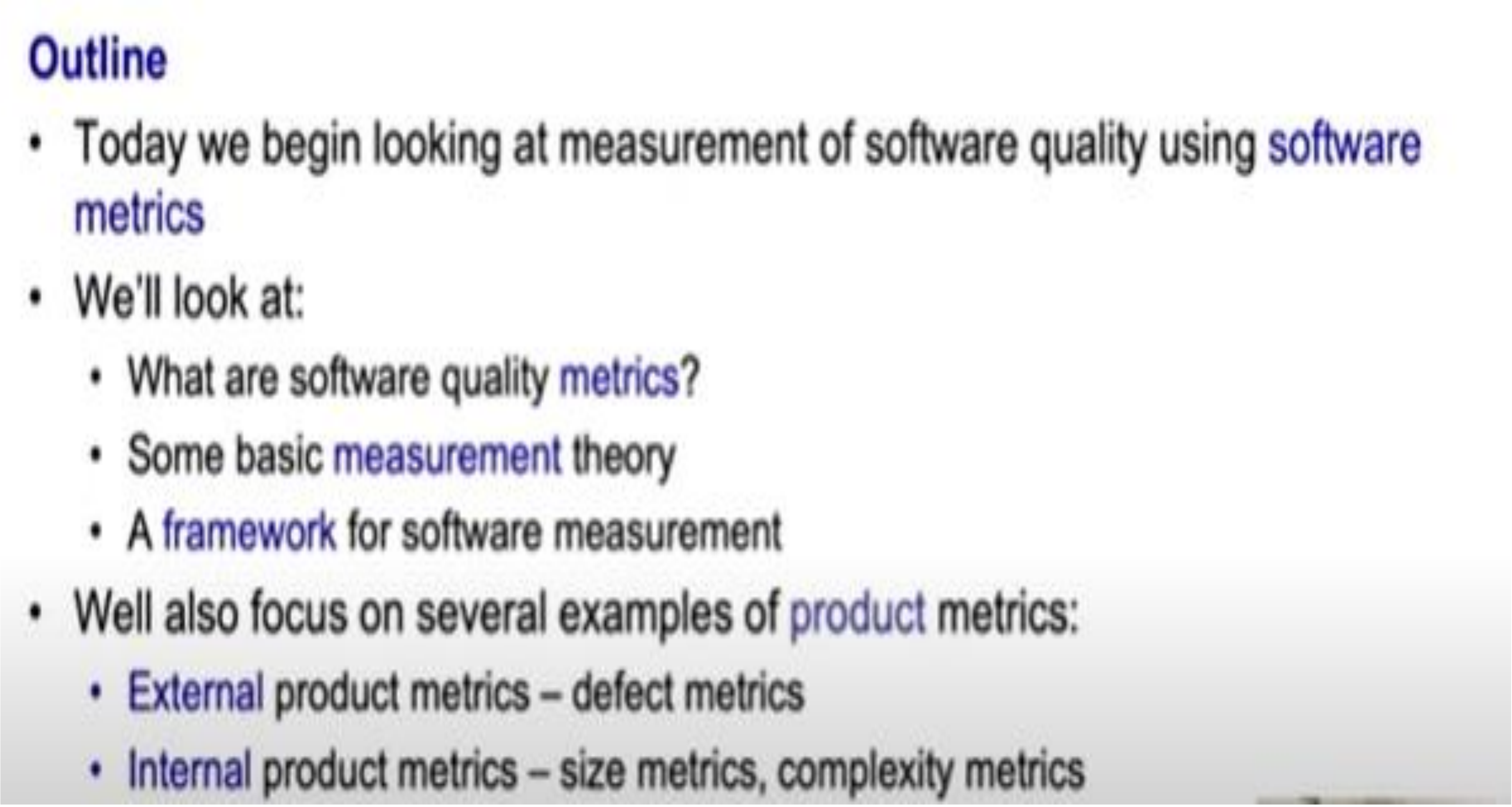
Team Velocity –

- It is an indicator of the number of tasks or user stories a team is able to complete during a single Time-Box.
- This does not include the items moved to the backlog and incomplete items.
- Only fully completed user stories are included for velocity calculations.
- This is an important metric because based on the team velocity, the management would decide on the number of stories they can pick up for the next Time-Box.

First Time Pass Rate –

- These metrics are in line with the agile principle of dynamic, fast and quality delivery.
- It is an indicator of the number of test cases that pass in the first run itself and also an indicator of the quality of development.
- In simpler terms, it means that no defects were found in the developed code when it went through testing for the first time.

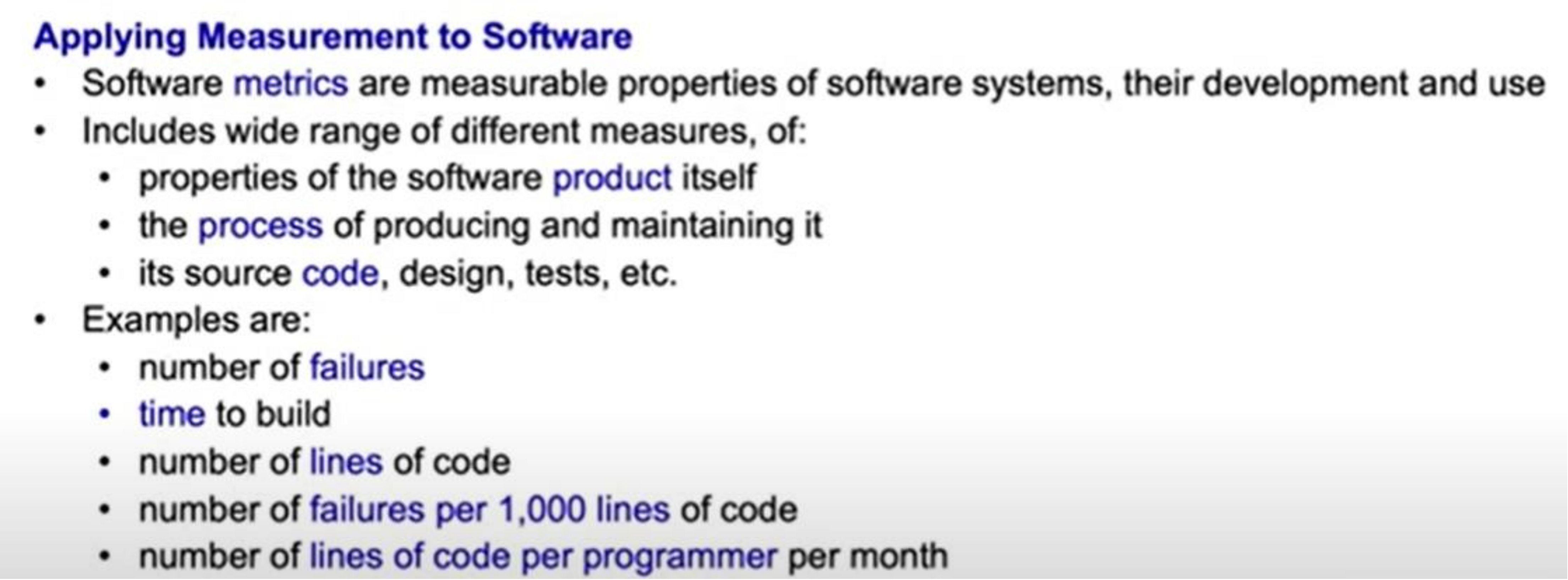
Software Measurement metrics for software quality



Outline

- Today we begin looking at measurement of software quality using software metrics
- We'll look at:
  - What are software quality metrics?
  - Some basic measurement theory
  - A framework for software measurement
- Well also focus on several examples of product metrics:
  - External product metrics – defect metrics
  - Internal product metrics – size metrics, complexity metrics

Software Quality metrics



Applying Measurement to Software

- Software metrics are measurable properties of software systems, their development and use
- Includes wide range of different measures, of:
  - properties of the software product itself
  - the process of producing and maintaining it
  - its source code, design, tests, etc.
- Examples are:
  - number of failures
  - time to build
  - number of lines of code
  - number of failures per 1,000 lines of code
  - number of lines of code per programmer per month

What are metrics Good for

Reliability and Quality Control

- Metrics helps us to predict and control the quality of our software
- Example: By measuring relative effectiveness of defect detection and removal of various testing or inspection methods, we can choose best one for our software products.

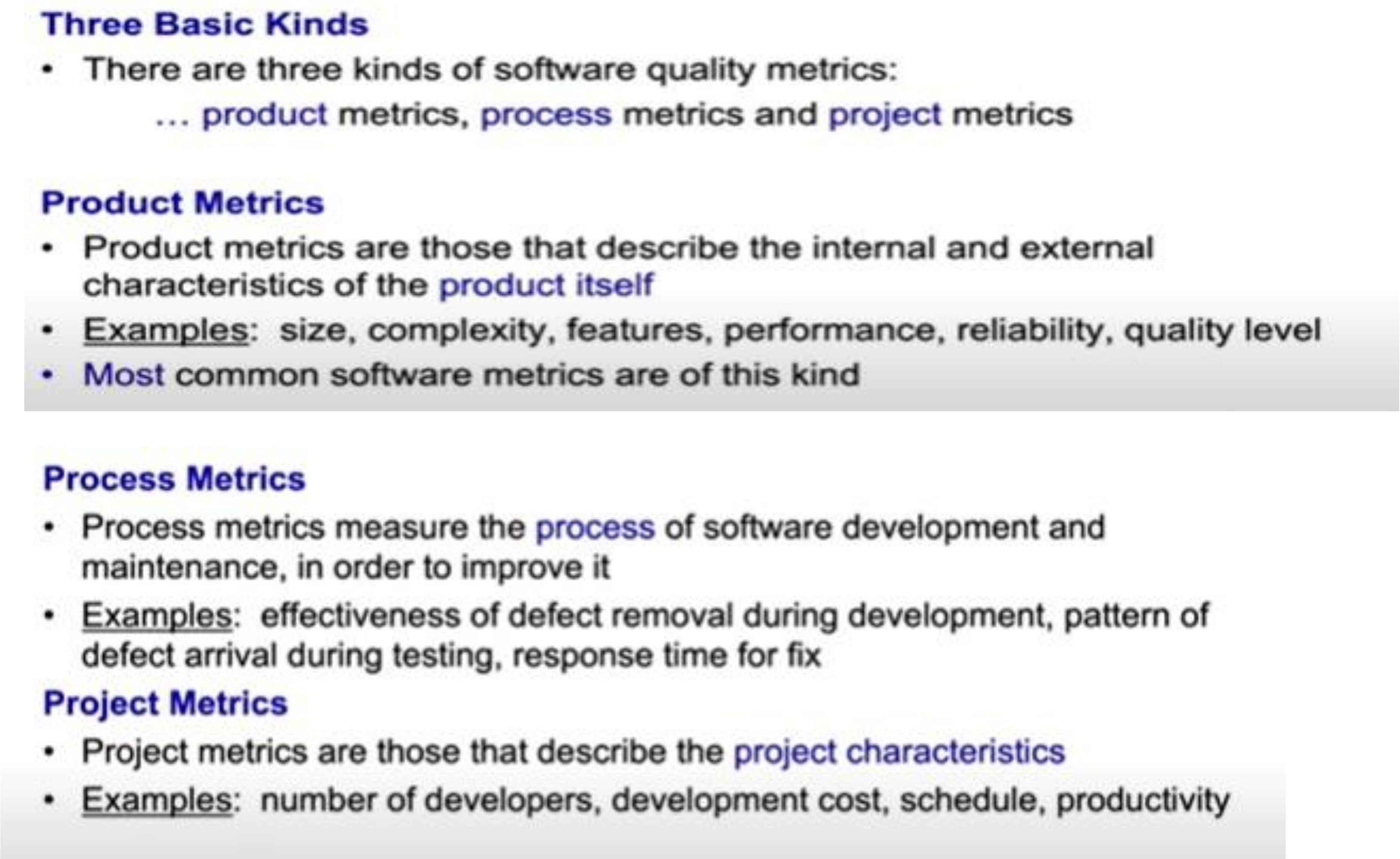
Cost Estimation and Productivity Improvement

- Metrics helps us predict effort to produce or maintain our software, and to improve our scheduling and productivity.
- Example: By measuring code production using different languages or tools, we can choose those that give the best results.

Quality Improvement

- Metrics helps us to improve code quality and maintainability
- Example: By measuring complexity of our program code, we can identify sections of code most likely to fail or difficult to maintain.

Kinds of Metrics



Three Basic Kinds

- There are three kinds of software quality metrics:

  … product metrics, process metrics and project metrics

Product Metrics

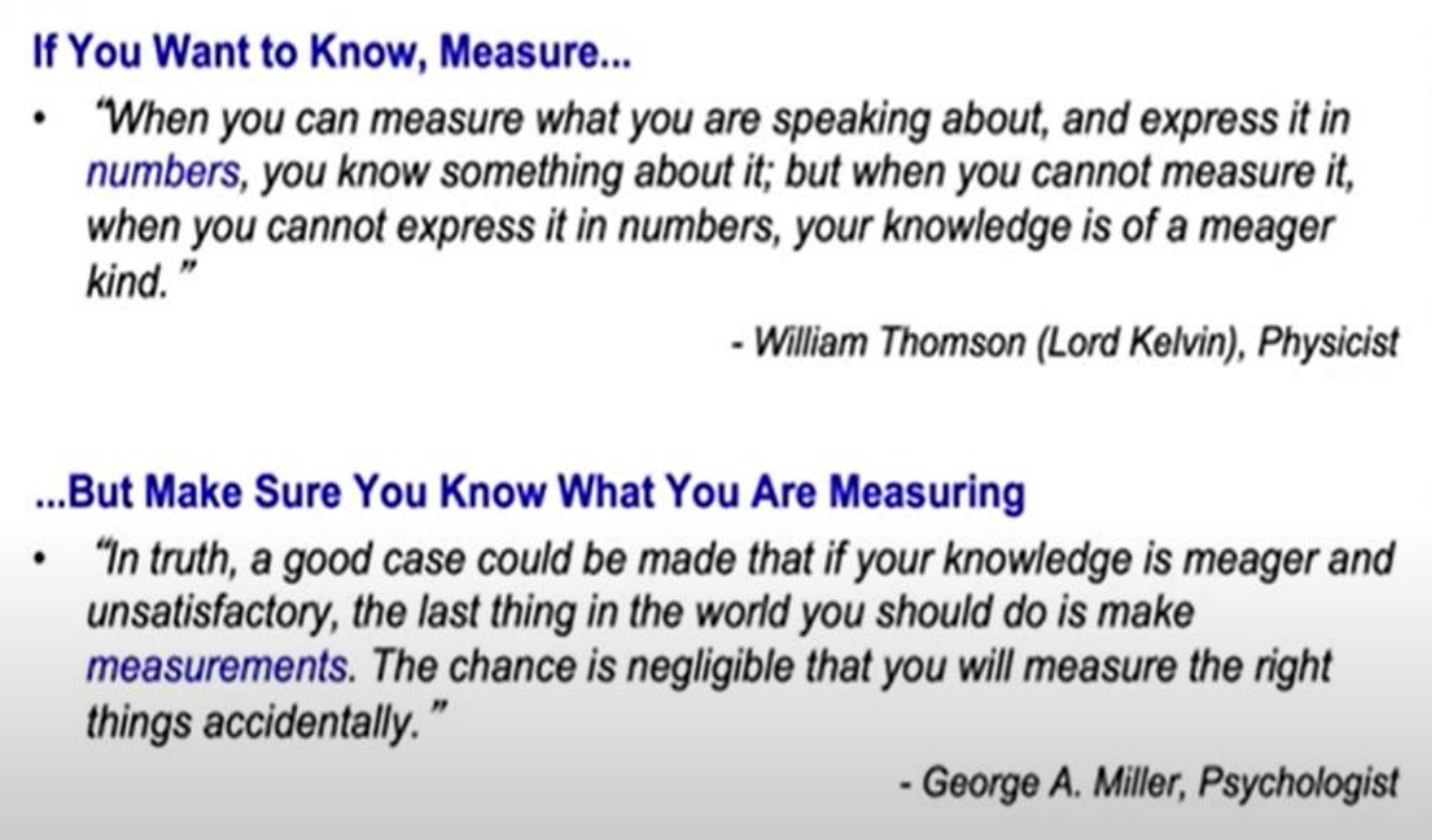
- Product metrics are those that describe the internal and external characteristics of the product itself
- Examples: size, complexity, features, performance, reliability, quality level
- Most common software metrics are of this kind

Process Metrics

- Process metrics measure the process of software development and maintenance, in order to improve it
- Examples: effectiveness of defect removal during development, pattern of defect arrival during testing, response time for fix

Project Metrics

- Project metrics are those that describe the project characteristics
- Examples: number of developers, development cost, schedule, productivity



If You Want to Know, Measure...

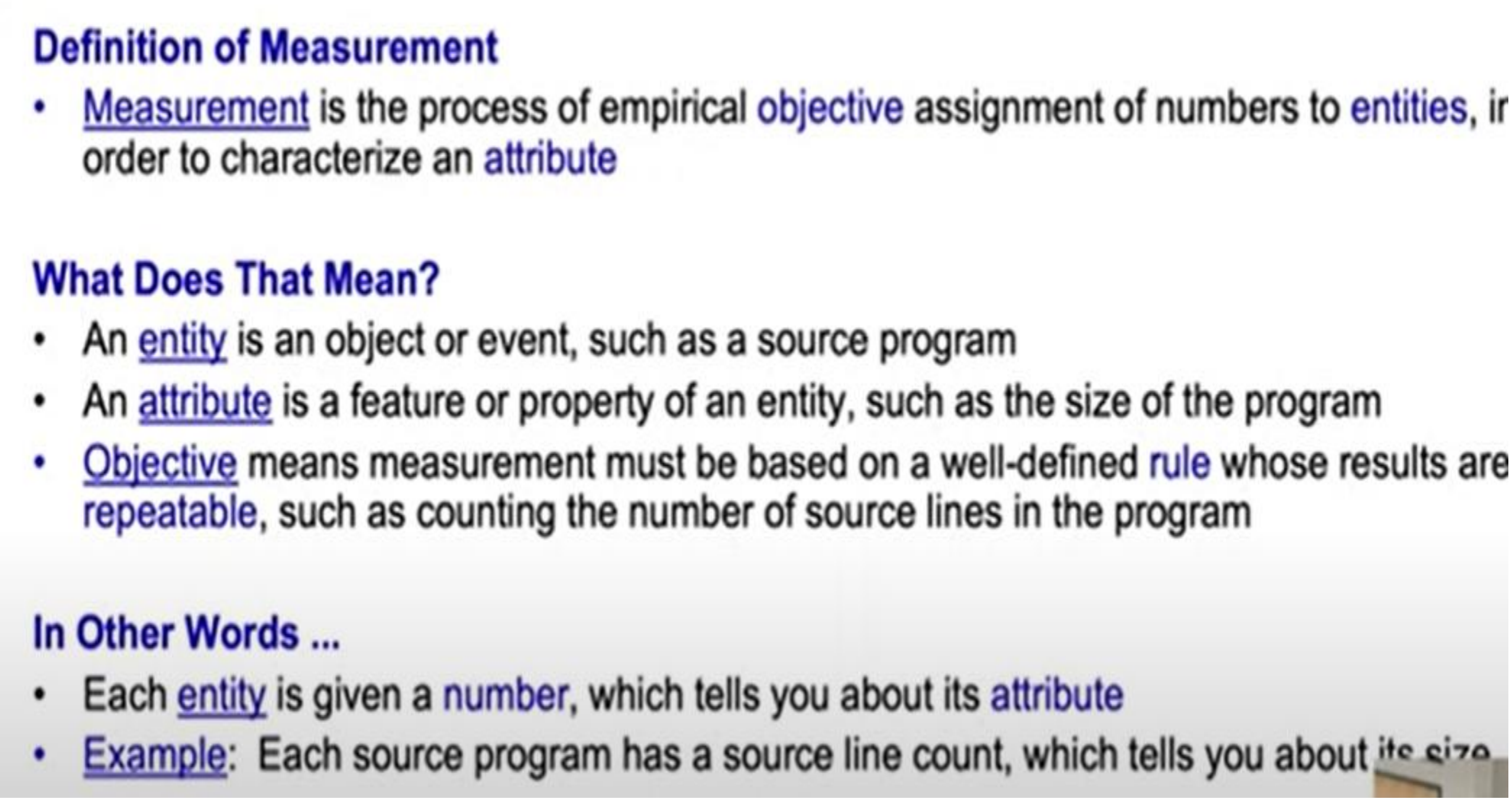
- *"When you can measure what you are speaking about, and express it in numbers, you know something about it; but when you cannot measure it, when you cannot express it in numbers, your knowledge is of a meager kind."*

- William Thomson (Lord Kelvin), Physicist

...But Make Sure You Know What You Are Measuring

- *"In truth, a good case could be made that if your knowledge is meager and unsatisfactory, the last thing in the world you should do is make measurements. The chance is negligible that you will measure the right things accidentally."*

- George A. Miller, Psychologist



Definition of Measurement

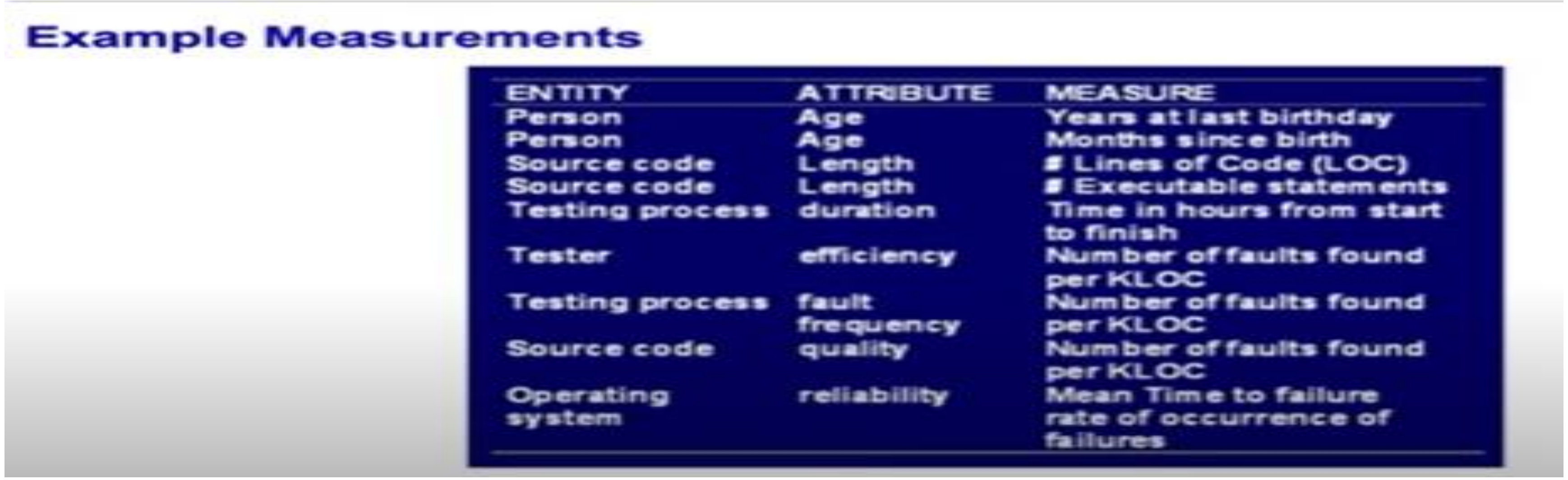
- Measurement is the process of empirical objective assignment of numbers to entities, in order to characterize an attribute

What Does That Mean?

- An entity is an object or event, such as a source program
- An attribute is a feature or property of an entity, such as the size of the program
- Objective means measurement must be based on a well-defined rule whose results are repeatable, such as counting the number of source lines in the program

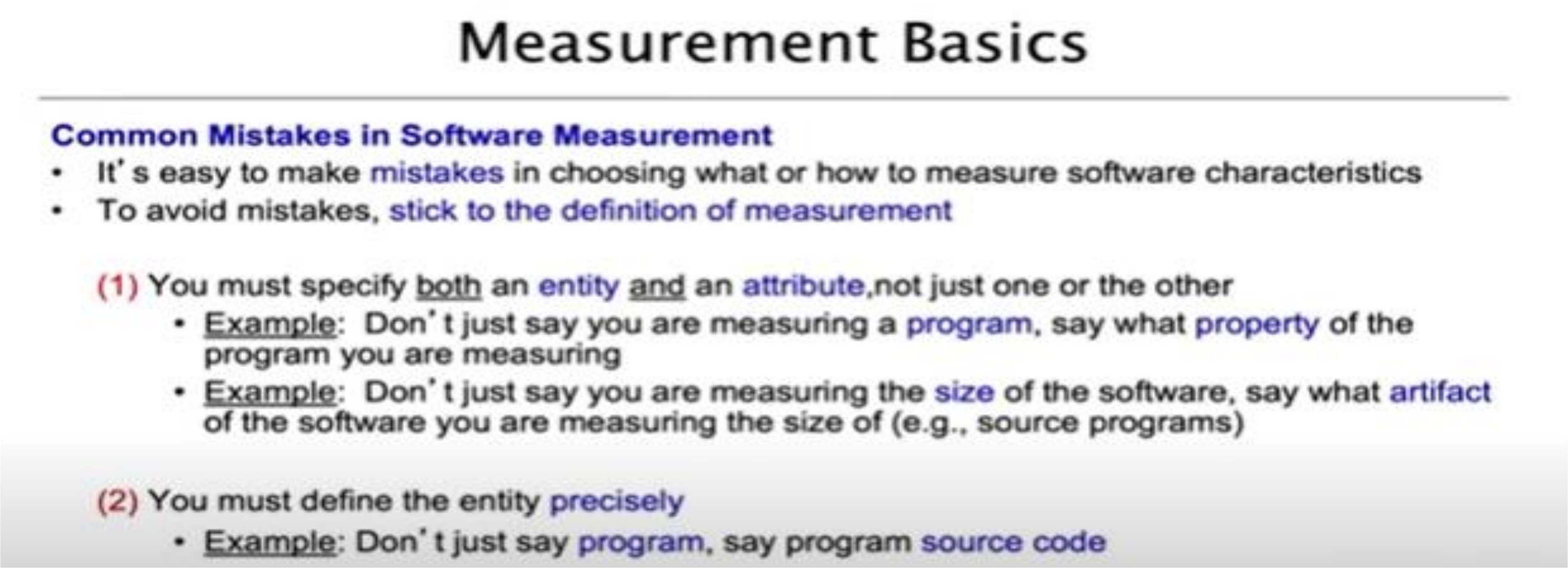
In Other Words ...

- Each entity is given a number, which tells you about its attribute
- Example: Each source program has a source line count, which tells you about its size



Example Measurements

ENTITY	ATTRIBUTE	MEASURE
Person	Age	Years at last birthday
Person	Age	Months since birth
Source code	Length	# Lines of Code (LOC)
Source code	Length	# Executable statements
Testing process	duration	Time in hours from start to finish
Tester	efficiency	Number of faults found per KLOC
Testing process	fault frequency	Number of faults found per KLOC
Source code	quality	Number of faults found per KLOC
Operating system	reliability	Mean Time to failure rate of occurrence of failures



Measurement Basics

Common Mistakes in Software Measurement

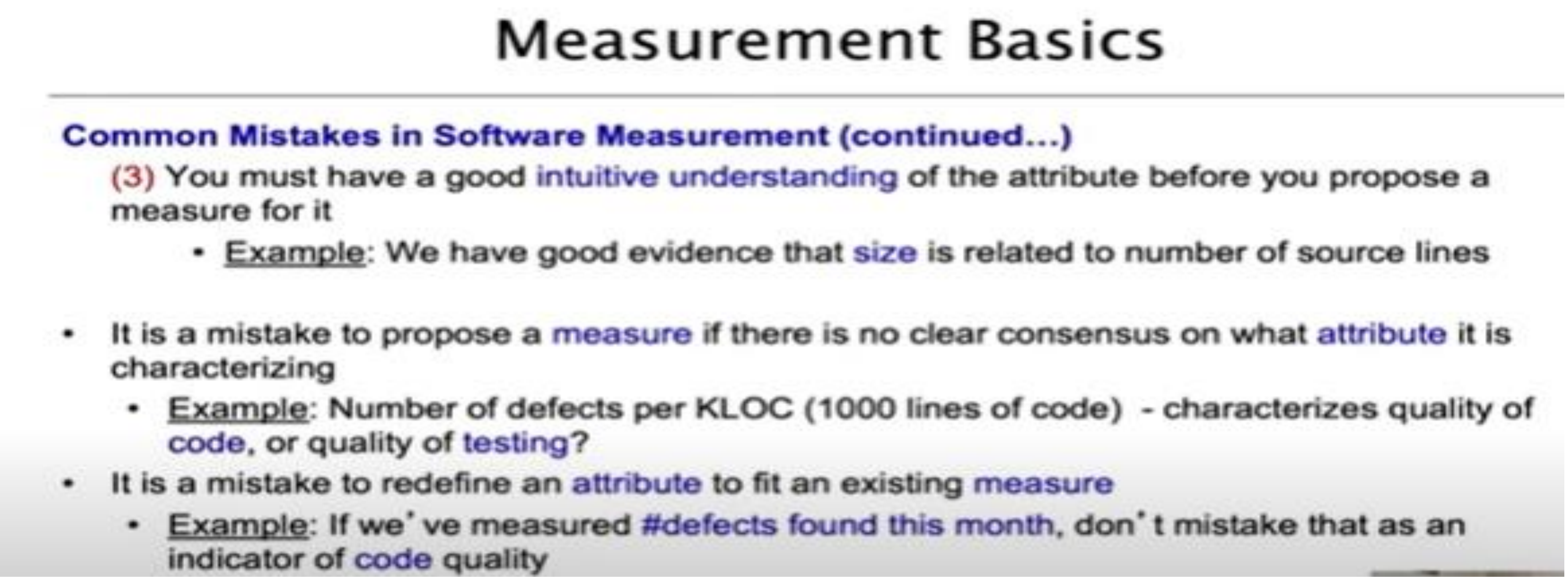
- It's easy to make mistakes in choosing what or how to measure software characteristics
- To avoid mistakes, stick to the definition of measurement

(1) You must specify both an entity and an attribute,not just one or the other

- Example: Don't just say you are measuring a program, say what property of the program you are measuring
- Example: Don't just say you are measuring the size of the software, say what artifact of the software you are measuring the size of (e.g., source programs)

(2) You must define the entity precisely

- Example: Don't just say program, say program source code



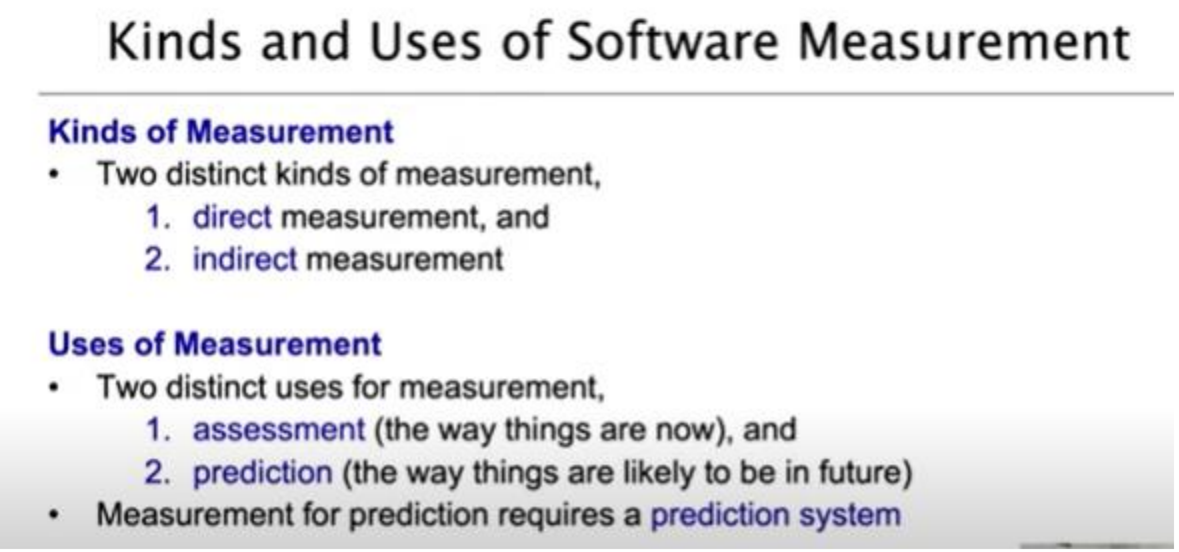
Measurement Basics

Common Mistakes in Software Measurement (continued…)

(3) You must have a good intuitive understanding of the attribute before you propose a measure for it

- Example: We have good evidence that size is related to number of source lines

- It is a mistake to propose a measure if there is no clear consensus on what attribute it is characterizing
  - Example: Number of defects per KLOC (1000 lines of code) - characterizes quality of code, or quality of testing?
- It is a mistake to redefine an attribute to fit an existing measure
  - Example: If we’ ve measured #defects found this month, don’ t mistake that as an indicator of code quality



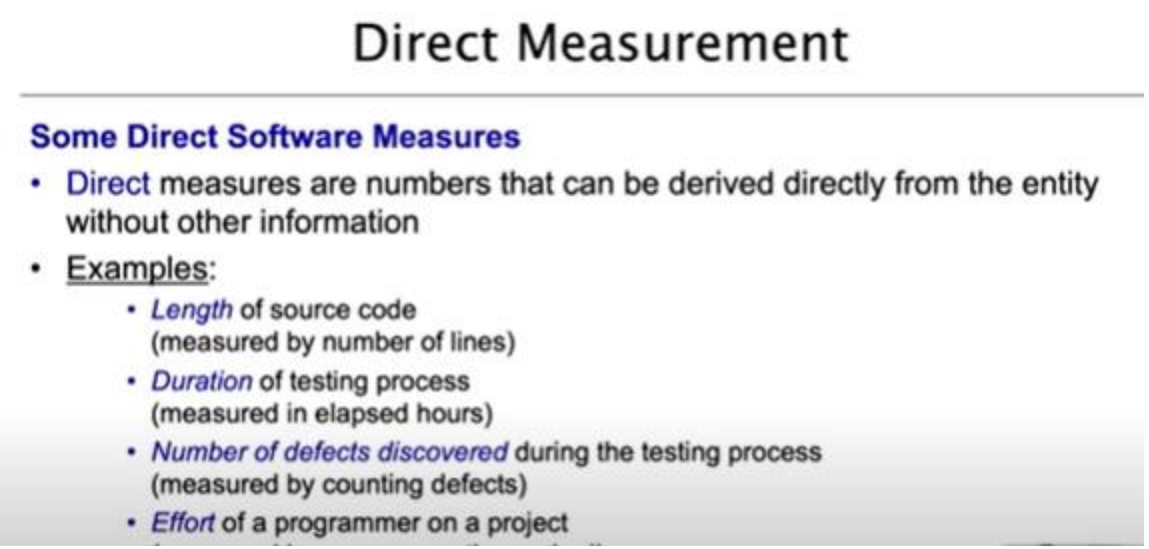
Kinds and Uses of Software Measurement

Kinds of Measurement

- Two distinct kinds of measurement,
  1. direct measurement, and
  2. indirect measurement

Uses of Measurement

- Two distinct uses for measurement,
  1. assessment (the way things are now), and
  2. prediction (the way things are likely to be in future)
- Measurement for prediction requires a prediction system

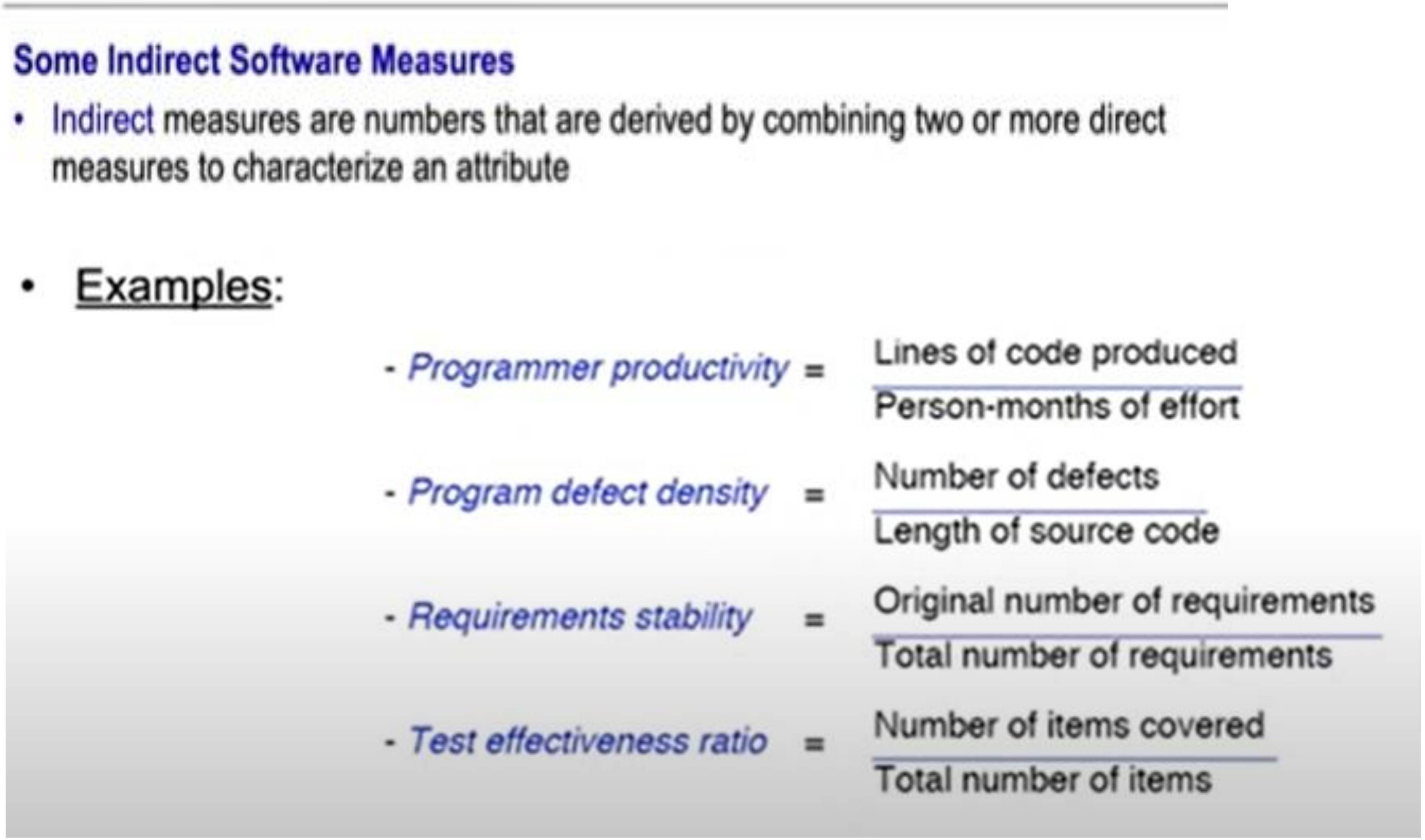


Direct Measurement

Some Direct Software Measures

- Direct measures are numbers that can be derived directly from the entity without other information
- Examples:
  - *Length* of source code
    (measured by number of lines)
  - *Duration* of testing process
    (measured in elapsed hours)
  - *Number of defects discovered* during the testing process
    (measured by counting defects)
  - *Effort* of a programmer on a project

Indirect Measurement



Some Indirect Software Measures

- Indirect measures are numbers that are derived by combining two or more direct measures to characterize an attribute

- <u>Examples</u>:

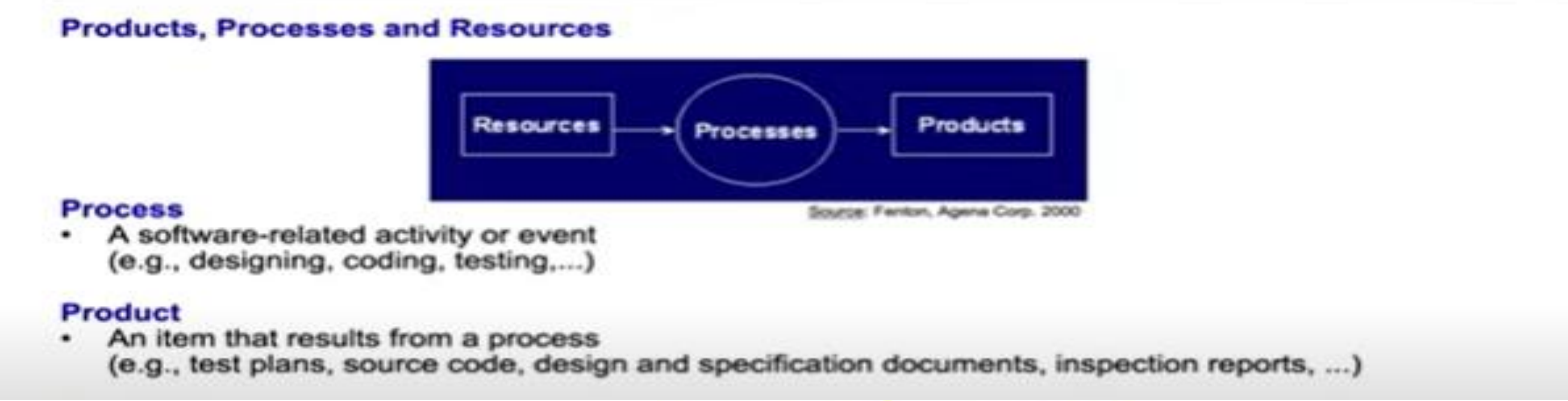
  - *Programmer productivity* $= \dfrac{\text{Lines of code produced}}{\text{Person-months of effort}}$

  - *Program defect density* $= \dfrac{\text{Number of defects}}{\text{Length of source code}}$

  - *Requirements stability* $= \dfrac{\text{Original number of requirements}}{\text{Total number of requirements}}$

  - *Test effectiveness ratio* $= \dfrac{\text{Number of items covered}}{\text{Total number of items}}$

A Framework for Software Measurement



Products, Processes and Resources

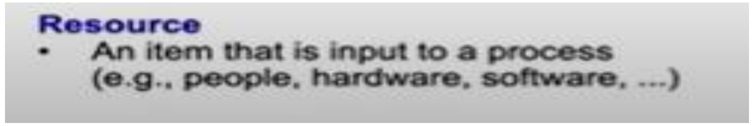
Source: Fenton, Agena Corp. 2000

Process

- A software-related activity or event
  (e.g., designing, coding, testing,...)

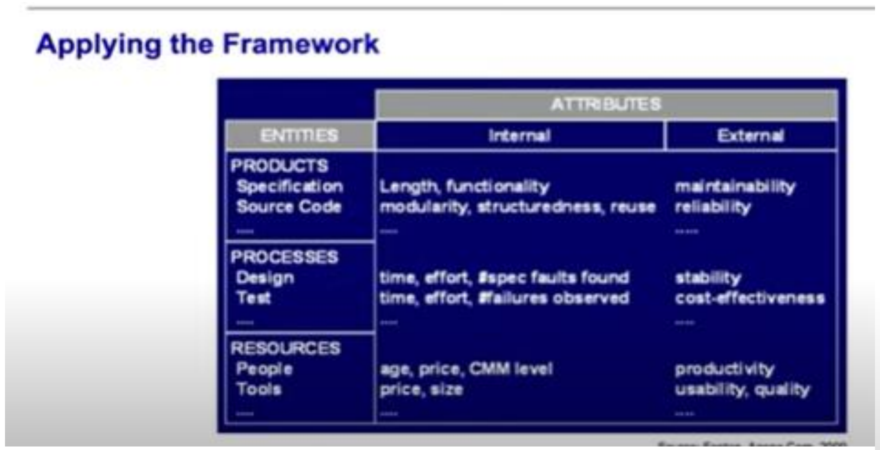
Product

- An item that results from a process
  (e.g., test plans, source code, design and specification documents, inspection reports, ...)



Resource

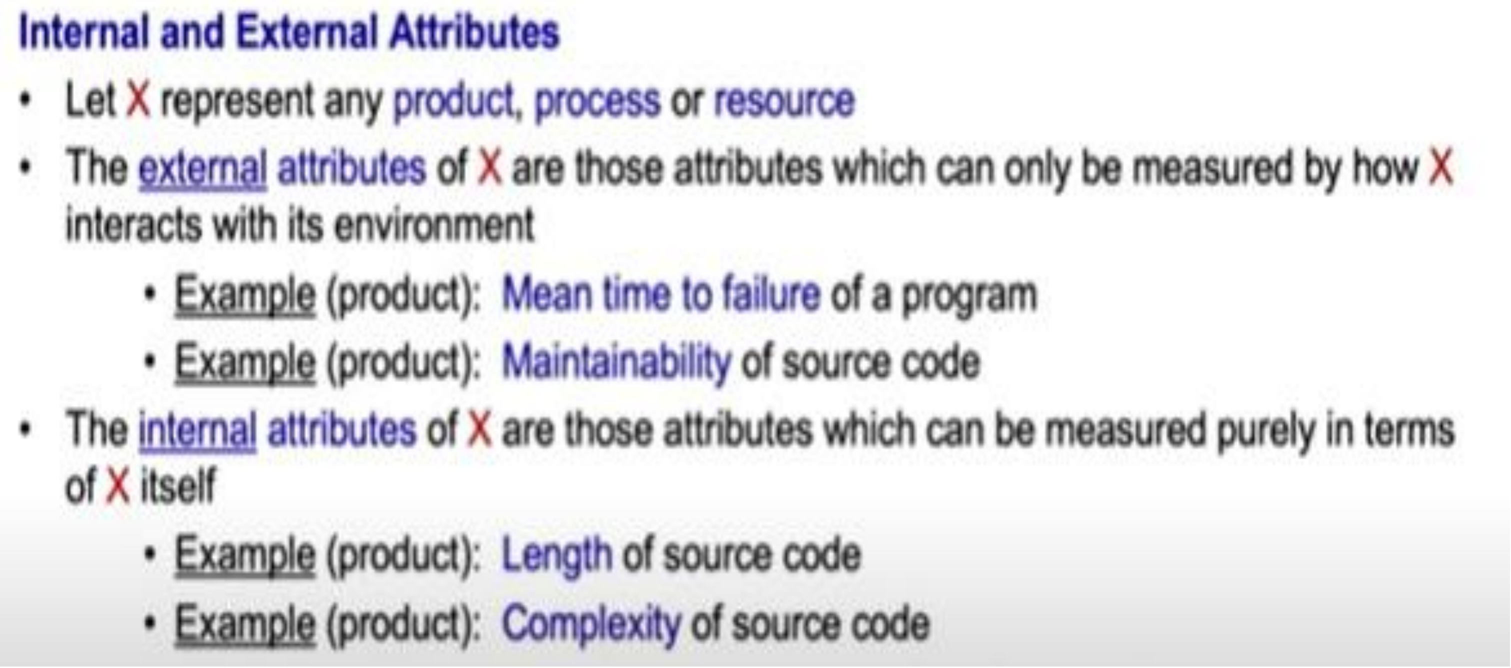
- An item that is input to a process
  (e.g., people, hardware, software, ...)



Applying the Framework

	ATTRIBUTES	
ENTITIES	Internal	External
PRODUCTS Specification Source Code	 Length, functionality modularity, structuredness, reuse	 maintainability reliability .....
PROCESSES Design Test	 time, effort, #spec faults found time, effort, #failures observed	 stability cost-effectiveness .....
RESOURCES People Tools	 age, price, CMM level price, size	 productivity usability, quality .....

Source: Fenton, Agena Corp. 2000



Internal and External Attributes

- Let X represent any product, process or resource
- The external attributes of X are those attributes which can only be measured by how X interacts with its environment
  - Example (product): Mean time to failure of a program
  - Example (product): Maintainability of source code
- The internal attributes of X are those attributes which can be measured purely in terms of X itself
  - Example (product): Length of source code
  - Example (product): Complexity of source code

Classification of software metrics

Functions

Product metrics

Describe the characterstics of the product

Generally include the measurement of

Size

Complexity

Design features

Performance

Quality level

Reliability

Functionality

Process metrics

Used to improve the development process and maintainance activities of software

Generally include the measurement of

Effort Required

Time to produce the product

Number of defects formed

Tools and technology

Quality and Efficiency

Project Metrics

Describe project characterstics and execution

Generally measures

Number of software developers

Cost

Schedule

Productivity

Quality

Assess status of ongoing project