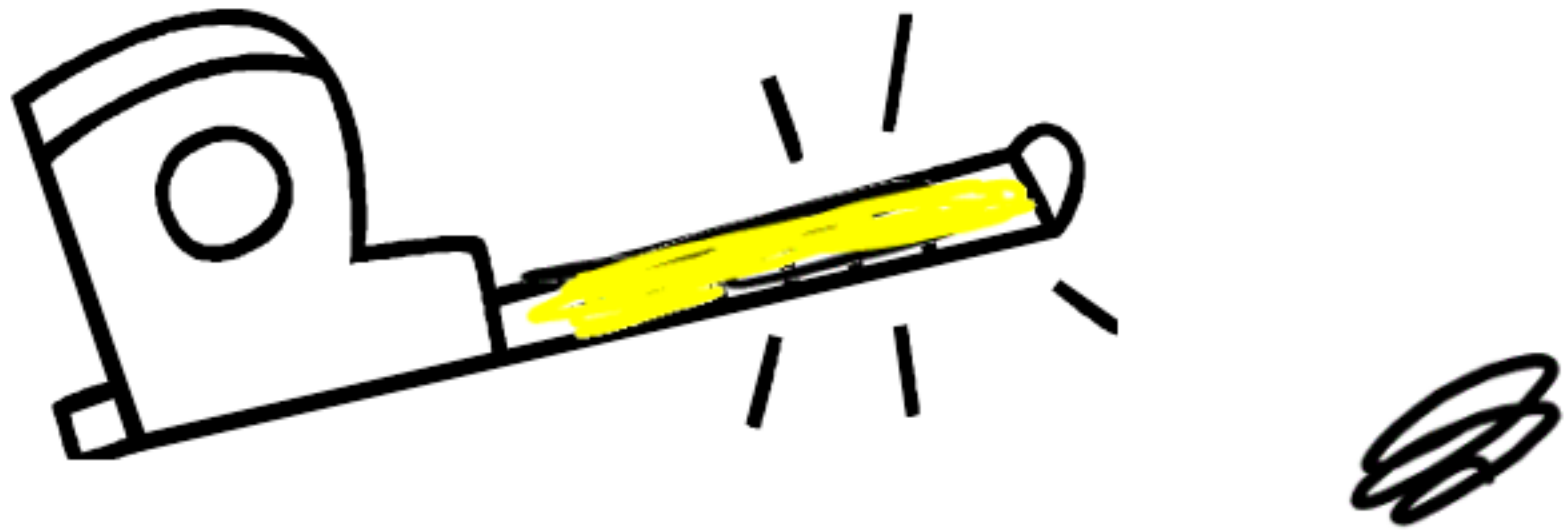
SE4831 : Software Quality Assurance

Metrics



Dr. Walter W. Schilling, Jr.

Instructor

Objectives

- Explain the objectives for software quality metrics
- Compare and contrast the relationship between measures, metric, and indicators
- Explain why trending is important to metrics management
- List Recommended quality indicators and justify the need for different metrics based on project phase
- Draw a diagram showing the process for managing metrics in a project
- List some interesting metrics

Objectives of Software Quality Management

1. **Facilitate management control, planning and managerial intervention.**

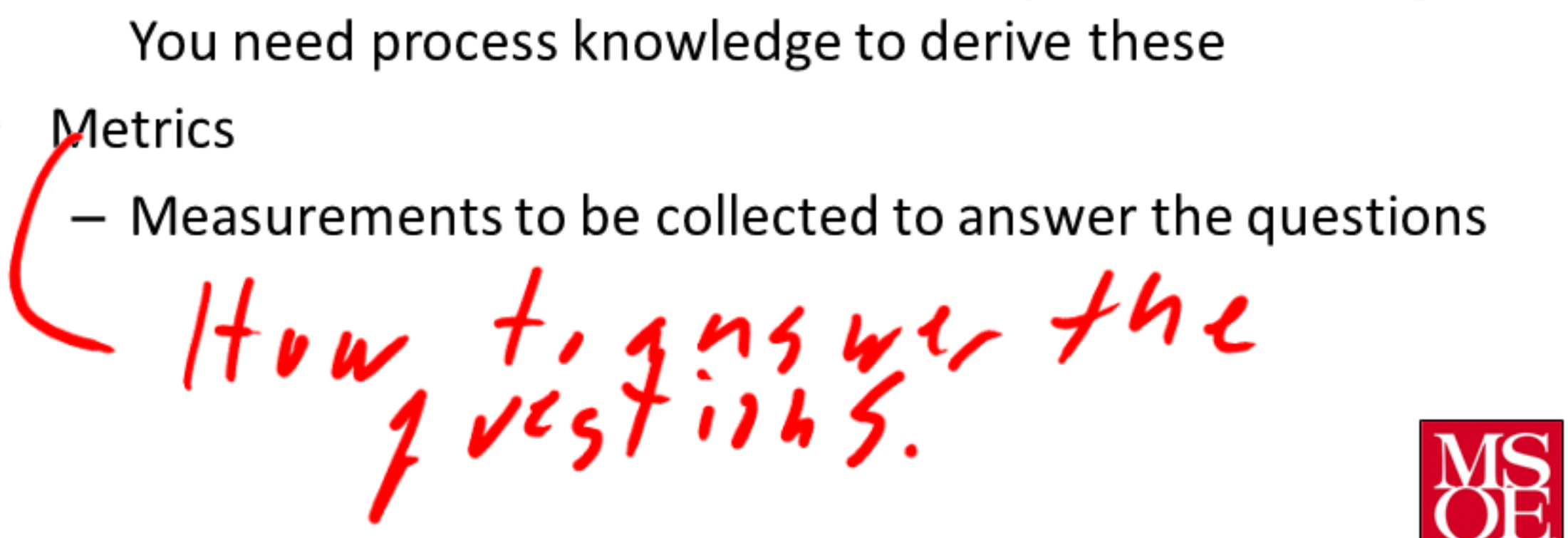
L4 Disciplined management

2. **Identify situations for development or maintenance process improvement (preventive or corrective actions).** Based on:

data

Goal-Question-Metric Paradigm

- Goals
  - What is the organisation trying to achieve? The objective of process improvement is to satisfy these goals
- Questions
  - Questions about areas of uncertainty related to the goals. You need process knowledge to derive these
- Metrics
  - Measurements to be collected to answer the questions



How would updates be applied?

What is the goal?

How is the SW doing?

General

Monitor all
of it

Bug open
Customer satisfaction
Uptime / Downtime.

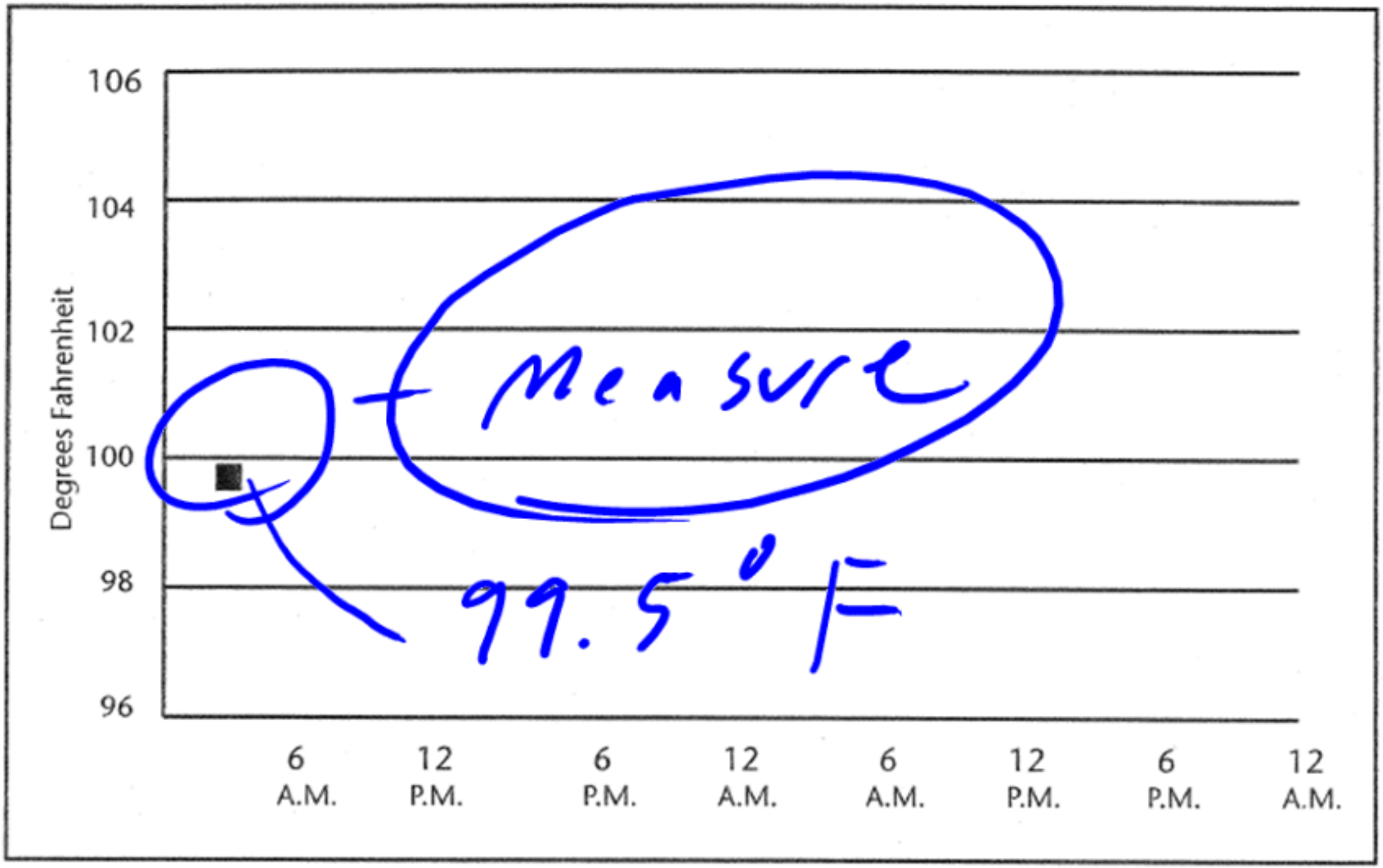
- who is the audience?
- key features?
- how will it effect budget?
- what am I spending my time on?

⇒ Percent time fixing bugs.
⇒ Percent time iterating...

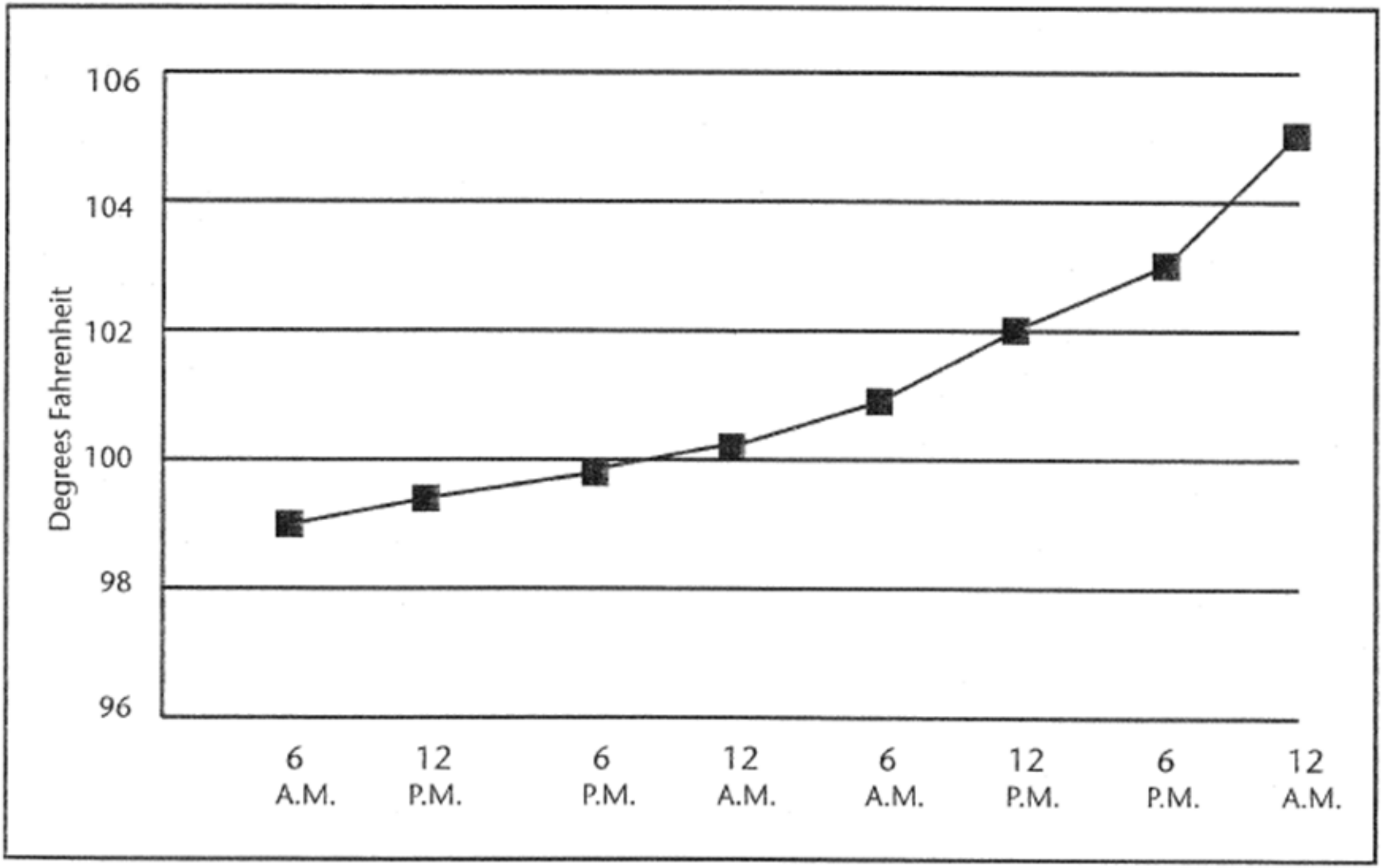
Measures, Metrics, and Indicators

- Measure — a comparison used to ascertain a condition based on a standard unit
- Metric — Qualitative measure of the degree to which a system possesses a given attribute
- Indicator — A variable which can be set to provide a state based on criteria

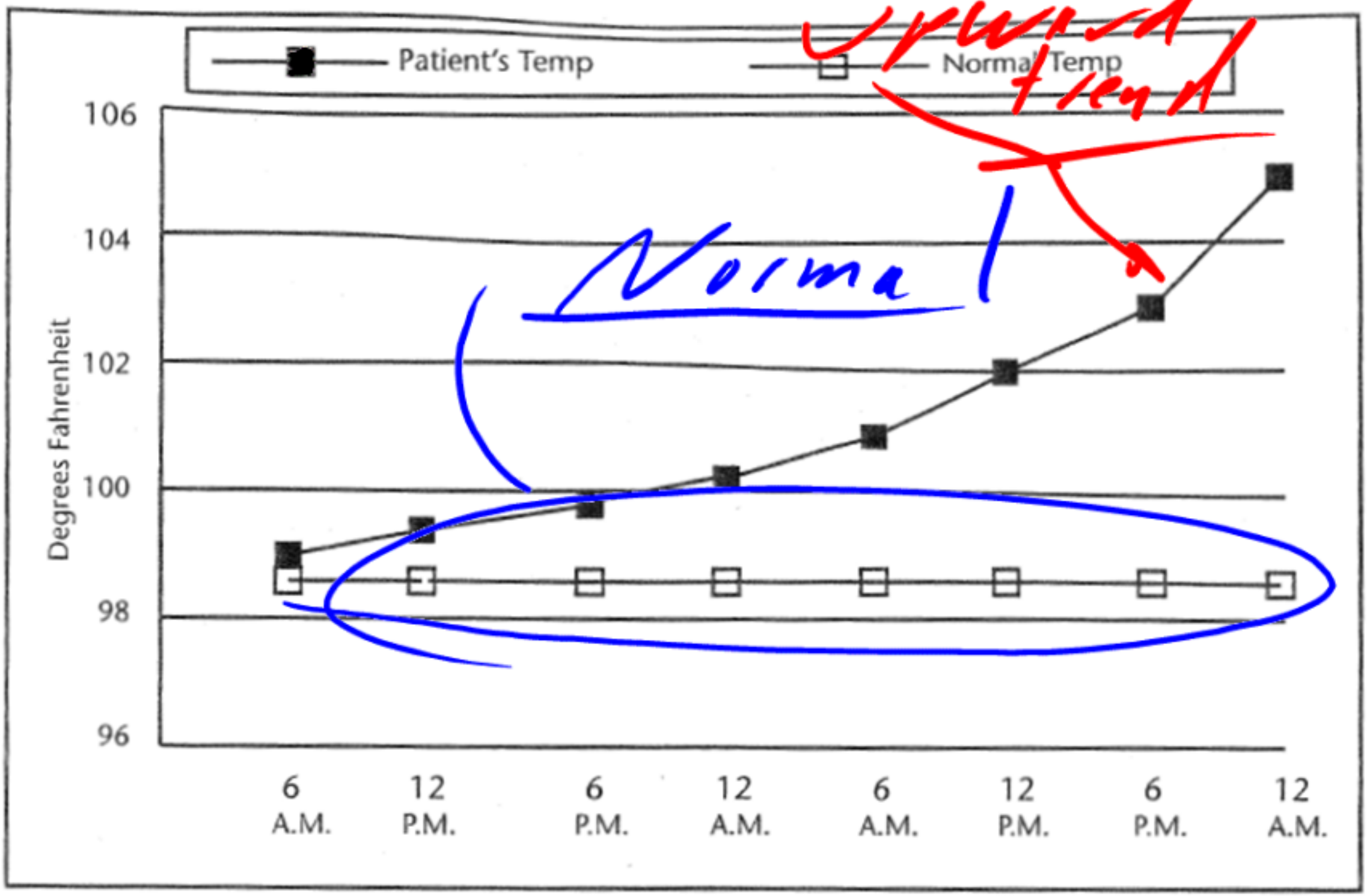
Body Temperature Measure



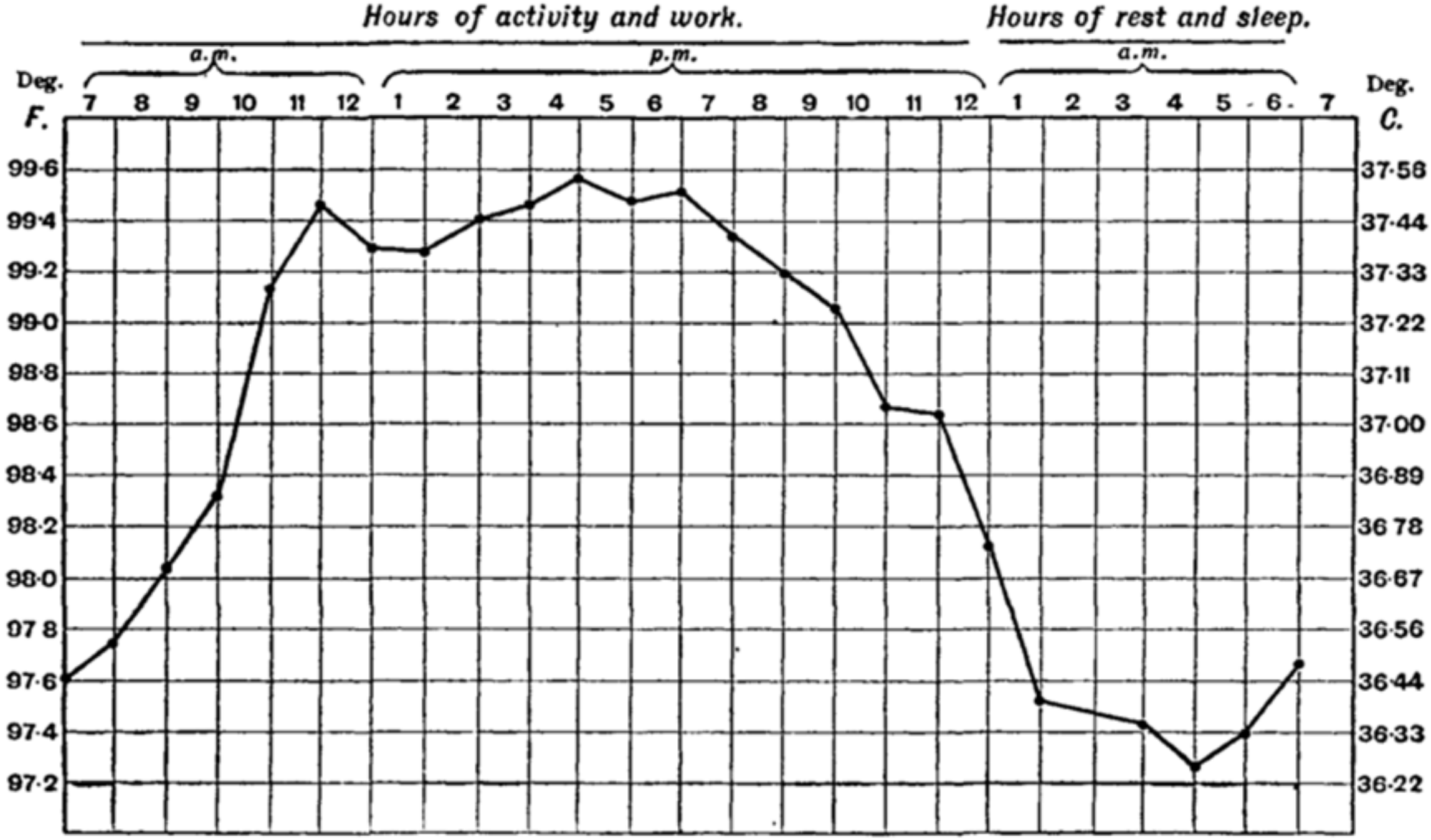
Body Temperature (Metric)



Body Temperature compared with Normal (indicator)



Another body temperature compared with Normal



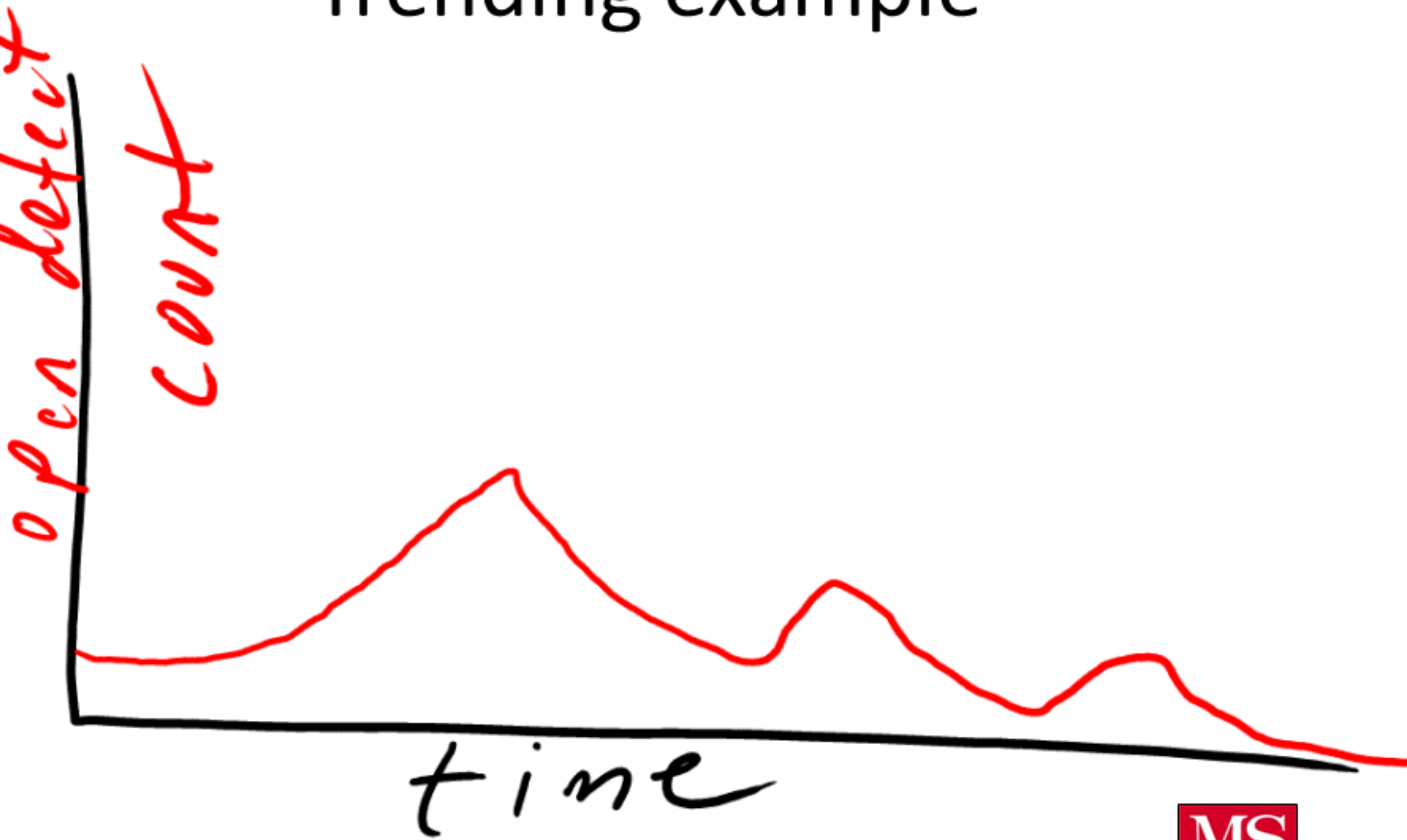
Why is trending important?

Trending tells us what is happening.

⇒ Are we getting

⇒ Are we getting worse?

Trending example



Quality Indicators

- Progress
- Stability
- Process Compliance
- Quality Evaluation Effort
- Test Coverage
- Defect detection efficiency
- Defect removal rate
- Defect age profile
- Defect density
- Complexity

How much are we doing towards the goal?

How stable is the product?

Phases of Development

Table 16.1 Quality Indicators by Development Phase

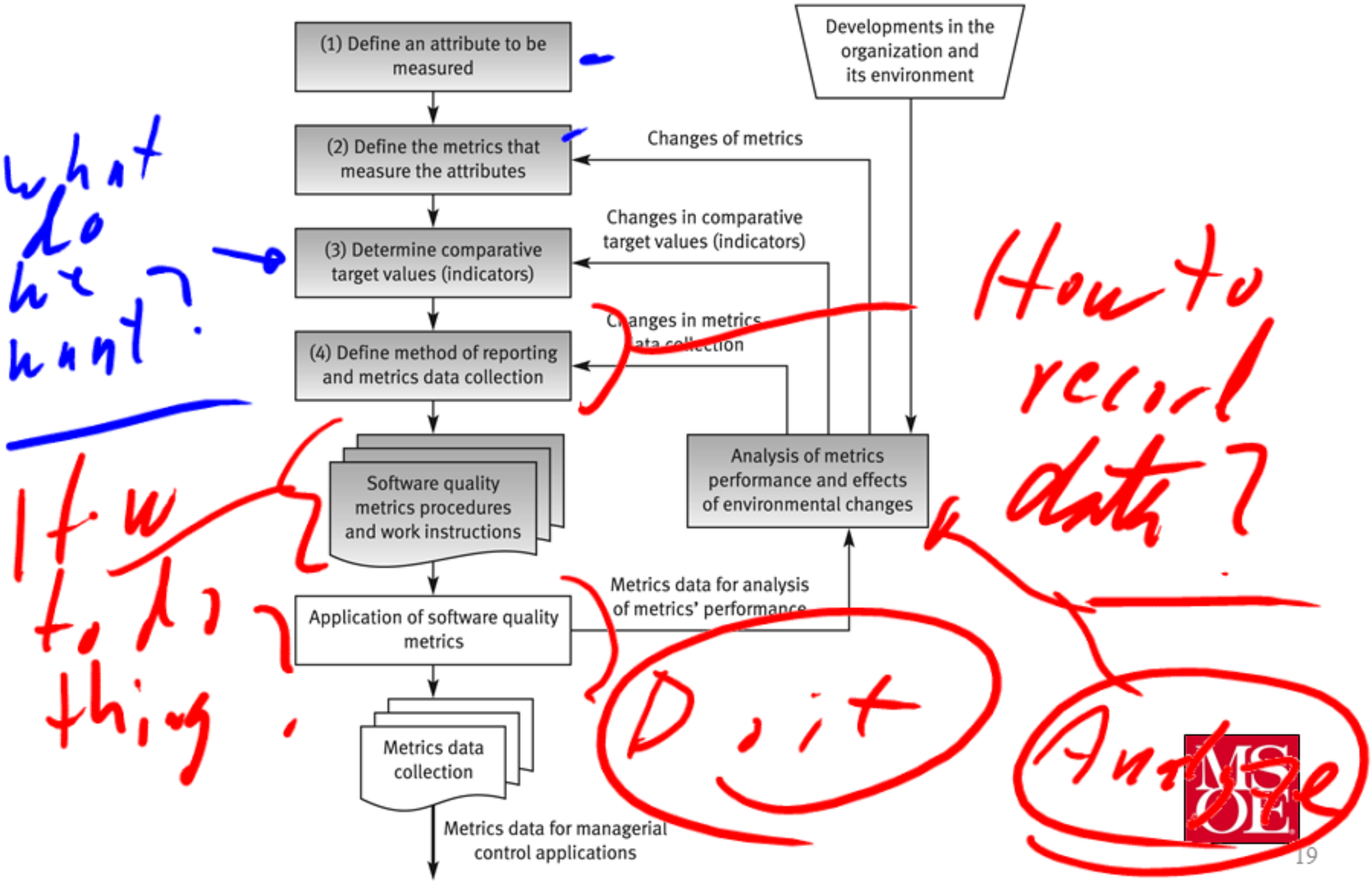
	Software Requirements Analysis	*Preliminary Design*	*Detailed Design*	*Code and Unit Testing*	*CSC Integration and Testing*	*CSCI Testing*
Process Indicators						
Management Concern						
Progress	Requirements volume	Top level design complete	Detailed design complete	Units completed	Tests accomplished	Tests accomplished
Stability	System requirements stability	Software requirements stability	Top level design stability	Detailed design stability	Software stability	Software stability
Compliance	Process compliance	<———	———	———	———	———>
Quality effort	Quality evaluation effort	<———	———	———	———	———>
Defect detection						
1. Test coverage				Percentage of paths executed	Percentage of paths executed	Percentage of functions executed
2. Defect detection efficiency		Defect detection efficiency	<———	———	———	———>

How much change?

	Software Requirements Analysis	*Preliminary Design*	*Detailed Design*	*Code and Unit Testing*	*CSC Integration and Testing*	*CSCI Testing*
Product Indicators						
Completeness	System requirements stability	Software requirements traceability				
1. Defect removal rate	Open and closed problem reports	<----------				---------->
2. Age profile	Problem report age profile	<----------				---------->
3. Defect density	Defect density	<----------				---------->
Complexity	Requirements complexity	Design complexity	Design complexity	Code complexity		

Source: [4].

The process of defining software quality metrics



How can we use metrics?

- Improving software quality
  - Reliability
  - Quality
  - Customer Satisfaction
- Modeling software development process
  - Estimating release times
  - Estimating delivered quality
- Improving Software development

Question

- How do we determine the stability of a software system

Indications of Module Change

- Software Maturity Index
  - Metric developed by IBM

$$SMI = 1 - \frac{CSI}{LOC}$$

  - Characterizes extent of change for individual modules
    - LOC – Lines of Code for the Module
    - CSI – Changed Source Instructions
      - SMI = 0 if a new module is being created
      - SMI = 1 if a module is unchanged
- Must understand both the module one is working on, but also the modules with which they interact.

Question

- How do we determine which modules are more likely to fail in the field?

Looking at test data?

which fail in testing...

Code churn

- Code churn is defined as lines added, modified or deleted to a file from one version to another.

Unstable modules