SOEN – 6841
Software Project Management

Amin Ranj Bar

Winter, 2021

Agenda

- Project Execution phase
- Testing
- Quality Assurance
- Measuring and Controlling Work Products & Processes
  - Definitions, GQM
  - Success Metrics (Key Performance Indicators (KPI))
  - Examples
- Plan the measurement process

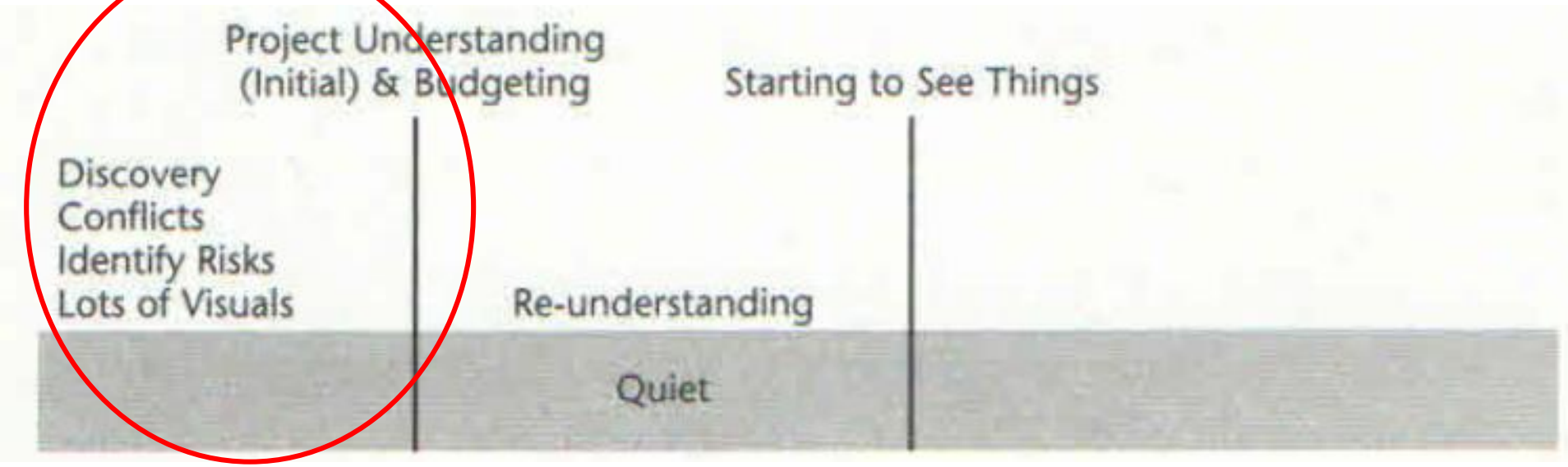
Agenda

- **Project Execution phase**
- Testing
- Quality Assurance
- Measuring and Controlling Work Products & Processes
  - Definitions, GQM
  - Success Metrics (Key Performance Indicators (KPI))
  - Examples
- Plan the measurement process

Review:
The Project First Third

- Roles, team building, and conflict resolution
- Discovery
- Risk identification
- Business requirements
- Budget
- Shared understanding and agreement on what to develop.

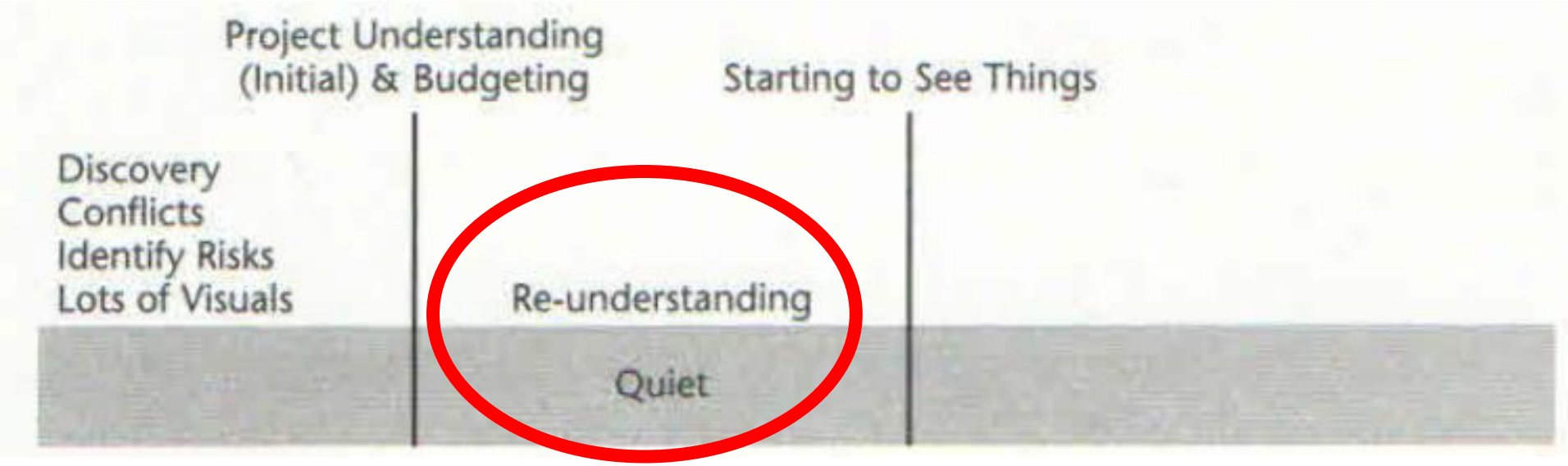
The Project First Third



The Second Third: Project execution phase

- Ongoing discovery: In-depth understanding of the business requirements
- Team key elements: Trust and collaboration
- Decision-making: Communicating and discerning problems
- Best practices: Visuals, demos, iterative deliverables:
  - Bring business stakeholders into the process frequently to give feedback on pieces of software (to be practiced in course the project)
  - Ongoing collaboration between project team and business stakeholders to arrive at final requirements
  - Customer Satisfaction Survey (will be also used in soen6841 demo evaluations)

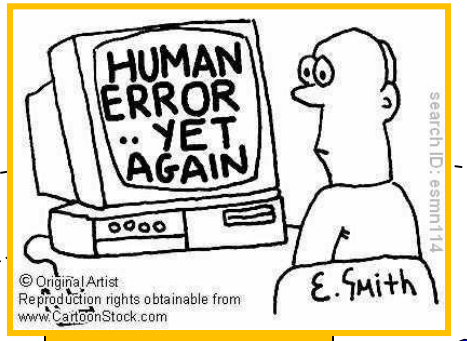
The Second Third: Project execution phase



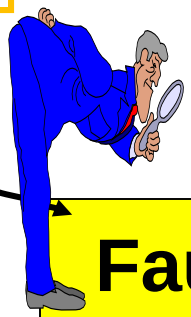
Entering The Final Third: "alpha" stage

- First-complete-look stage
- Use guided walk-throughs to confirm features and functions
- Focus on making sure all "Happy End" scenarios work

Some **BUG** Terminology: error, fault, failure



Error



It

Fault

TESTER

Failure

Observed Incident

Bug life cycle reporting

- Bug tracking systems (Bugzilla…)
- stages: open, fixed, resolved, closed

Agenda

- Project Execution phase
- **Testing**
- Quality Assurance
- Measuring and Controlling Work Products & Processes
  - Definitions, GQM
  - Success Metrics (Key Performance Indicators (KPI))
  - Examples
- Plan the measurement process

Software Testing : Definition

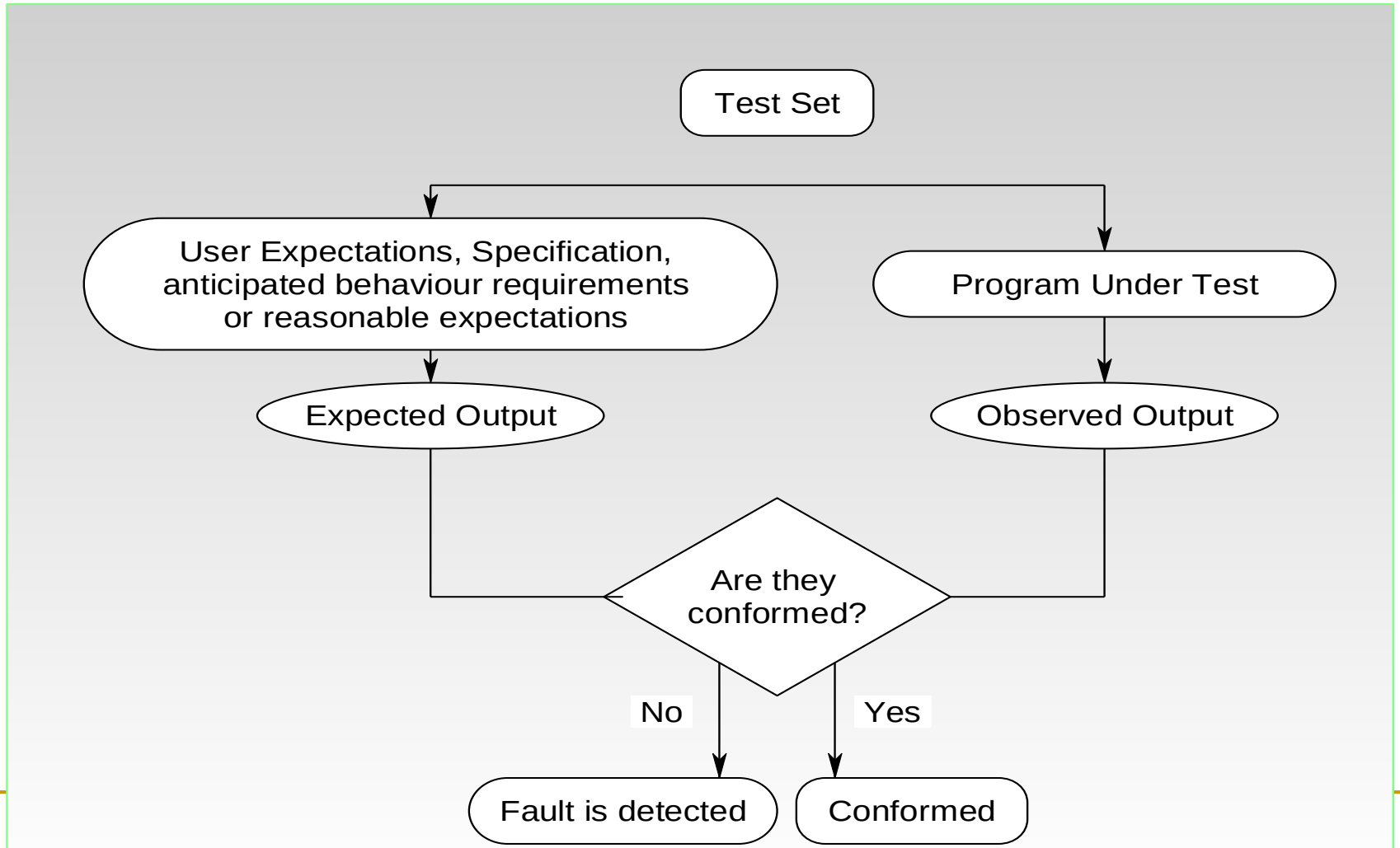
Software testing consists of the dynamic verification of the behaviour of a program on a finite set of test cases, suitably selected from the usually infinite executions domain, against the specified expected behaviour

(SWEBOK definition – ISO TR 19759)

Compare Testing and Debugging

- Testing is concerned with confirming the presence of faults
- Debugging is concerned with locating and repairing these faults

Academic View on Testing: Oracle Methodology



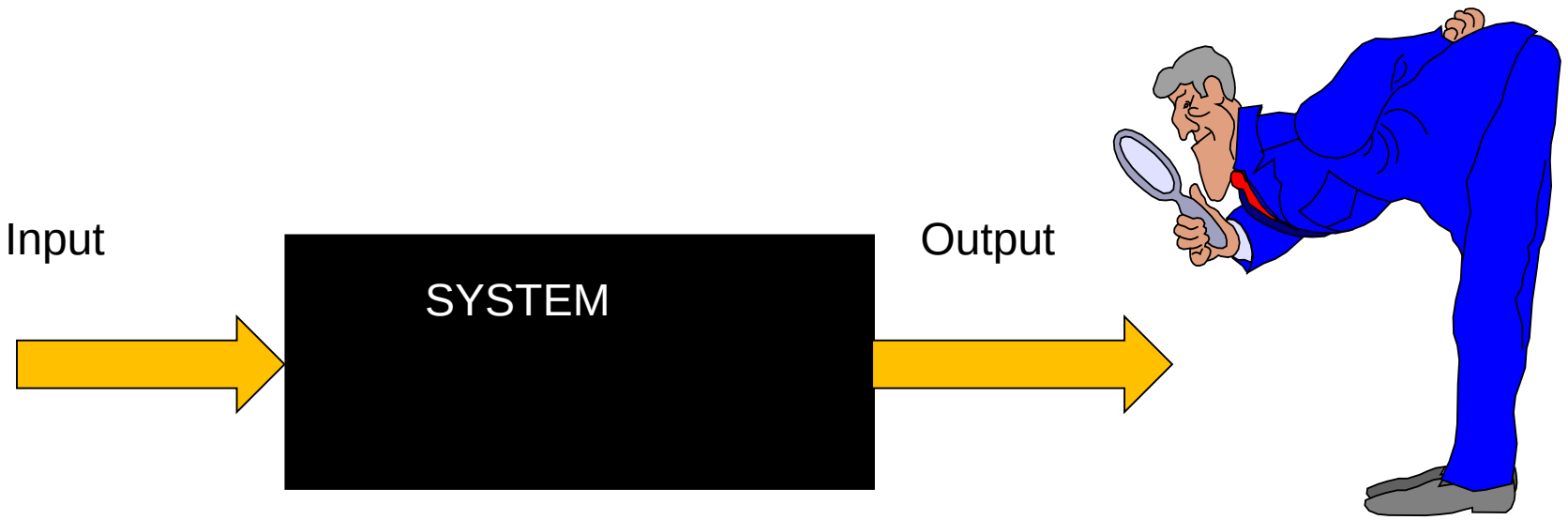
Testing at different levels and stages

- ***Unit testing:*** testing each module in isolation. Involves the creation of a testing environment for providing input/collecting output to the component.
- ***Integration testing:*** testing groups of components incrementally. Defining the order of integration is of prime importance.
- ***System (Function) testing:*** functionalities of the system are tested using the whole system
- ***Performance testing:*** Testing the system to the limits of its operational requirements, and beyond.
- ***Acceptance testing:*** The client is invited to test the software for acceptance (alpha testing)
- ***Installation testing:*** After installation on-site, the software is tested again.
- ***Beta testing:*** the software is given to many customers for them to install and test with no test guidance at all.

Identifying Test Cases

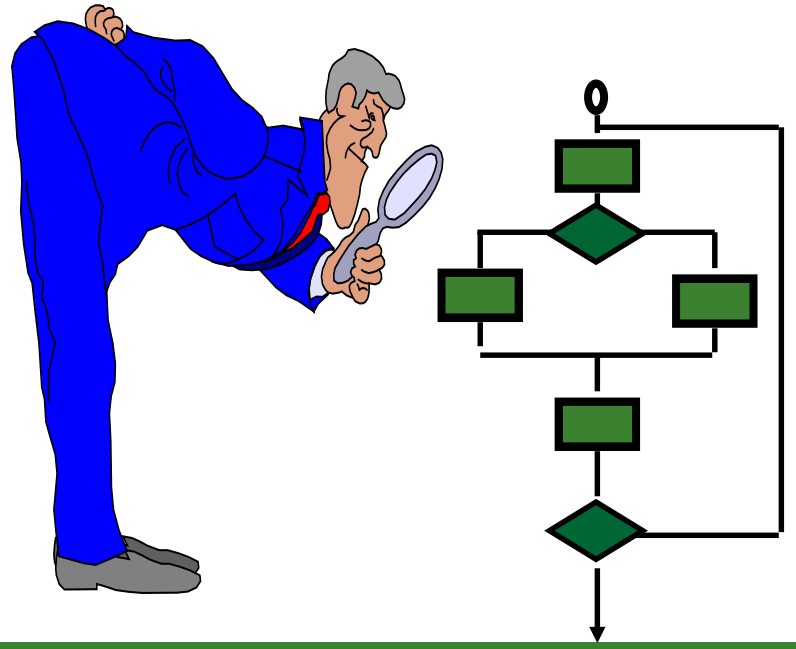
- Testing approaches:
  - **Functional (black-box)**
    - Derived from specs
    - Reusable even if the code changes
  - **Structural (white-box)**
    - Derived from code documentation
    - Test coverage metrics

Functional (Black-Box) Testing



FUNCTION: System (Input) = Output

White-Box Testing



- The structure of the code is “visible”
- Targets the control flow within a Unit of code

Testing Key points

- Detects faults in systems but does not try to recover from the failures caused by them

  Testing can only show the presence of faults, not their absence

  - A **good test case** is one that has a high probability of finding an as-yet-undiscovered bug

  - A **successful test** is one that uncovers an as-yet-undiscovered error

Progression vs. Regression Tests

- **Regression tests:** Ensure that nothing had has happened after a fix.
  - The most common approach to regression testing is to simply repeat the system tests.
  - We can refine this (and drastically reduce the effort) by choosing test threads with respect to the goals of regression and progression testing.
- With progression testing, we are testing new territory, so we expect a higher failure rate than with regression testing.

User Acceptance Testing (UAT)

- UAT is a Testing carried on by stakeholders to "confirm" that a user story was implemented correctly.
- Plan for UAT at the beginning of the software development process
- UAT should happen as you present frequent iterative deliverables / demos
- Feedback is collected and integrated (UAT testers need training on how to report feedback, see next slide)
- After receipt of a software correction, the acceptance test team performs regression testing to ensure the defect or issue has been resolved and previously working software is not impacted.

Classifying UAT Feedback

1. Bug
2. Function not working as expected
3. Request for improvement
4. Feature request

Rule of thumb:

- Address all 1. and 2.
- a small number of 3. may be included
- Defer all 4. to next release

Example of UAT in Agile

User story:

As a recurring customer,

I want to reorder items from my previous orders

so that I don't have to search for them each time.

Create the Happy Path:

1. Purchase items from general search
2. Click order history from accounts page
3. Verify that previously ordered items are displayed
4. Add items to cart from the previously ordered list
5. Complete purchase of previously ordered items

Acceptance Regression Testing

- *After receipt of a software correction, the acceptance test team performs regression testing to ensure the defect or issue has been resolved and previously working software is not impacted.*

Some other testing types

- Load testing
- Usability testing
- Security testing

Agenda

- Project Execution phase
- Testing
- **Quality Assurance**
- Measuring and Controlling Work Products & Processes
  - Definitions, GQM
  - Success Metrics (Key Performance Indicators (KPI))
  - Examples
- Plan the measurement process

Defect Density measurement and analysis

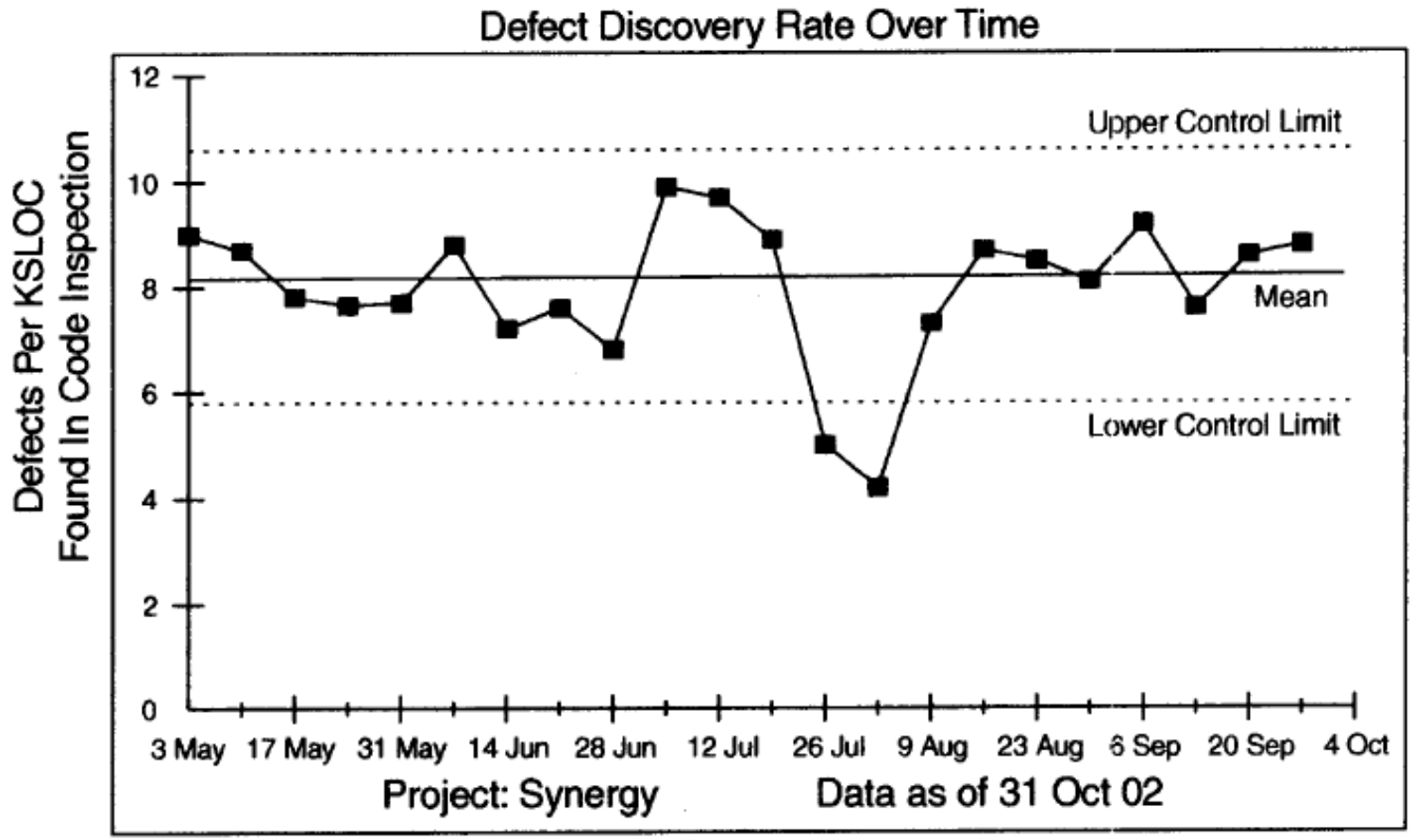
- **Measurement Function**

  defect density for package X =

  (total defects for package X) / (package X size)

- **Define an analysis model**
  - *Compute process centre and control limits using values of defect density*
- **Define rules for interpretation of an indicator (decision-criteria)**
  - *Results outside the control limits require further investigations*

Analysis of design quality



Software Quality Assurance : Definition

- Definition:

"The function of software quality that assures that the standards, processes, and procedures are appropriate for the project and are correctly implemented."

https://sma.nasa.gov/sma-disciplines/software-assurance

- Systematic run-through all use cases in the software
- Do full test prior to launch

SQA Activities

- Prepares an SQA plan for a project
- Participates in the development of the project's software process description
- Reviews software engineering activities to verify compliance with the defined software process
- Audits designated software work products to verify compliance with those defined as part of the software process
- Ensures that deviations in software work and work products are documented and handled according to a documented procedure
- Records any noncompliance and reports to senior management
- Coordinates the control and management of change
- Helps to collect and analyze software metrics

Testing v.s. Software Quality Assurance

- **Testing:** product-oriented
- **SQA**: process-oriented

- Rational:
  - improving the Process leads to an improvement of the resulting Product

Quality Control

- Involves a series of inspections, reviews, and tests used throughout the software process
- Ensures that each work product meets the requirements placed on it
- Requires all work products to have defined, measurable specifications to which practitioners may compare to the output of each process

Formal Technical Review (also called a walkthrough or inspection)

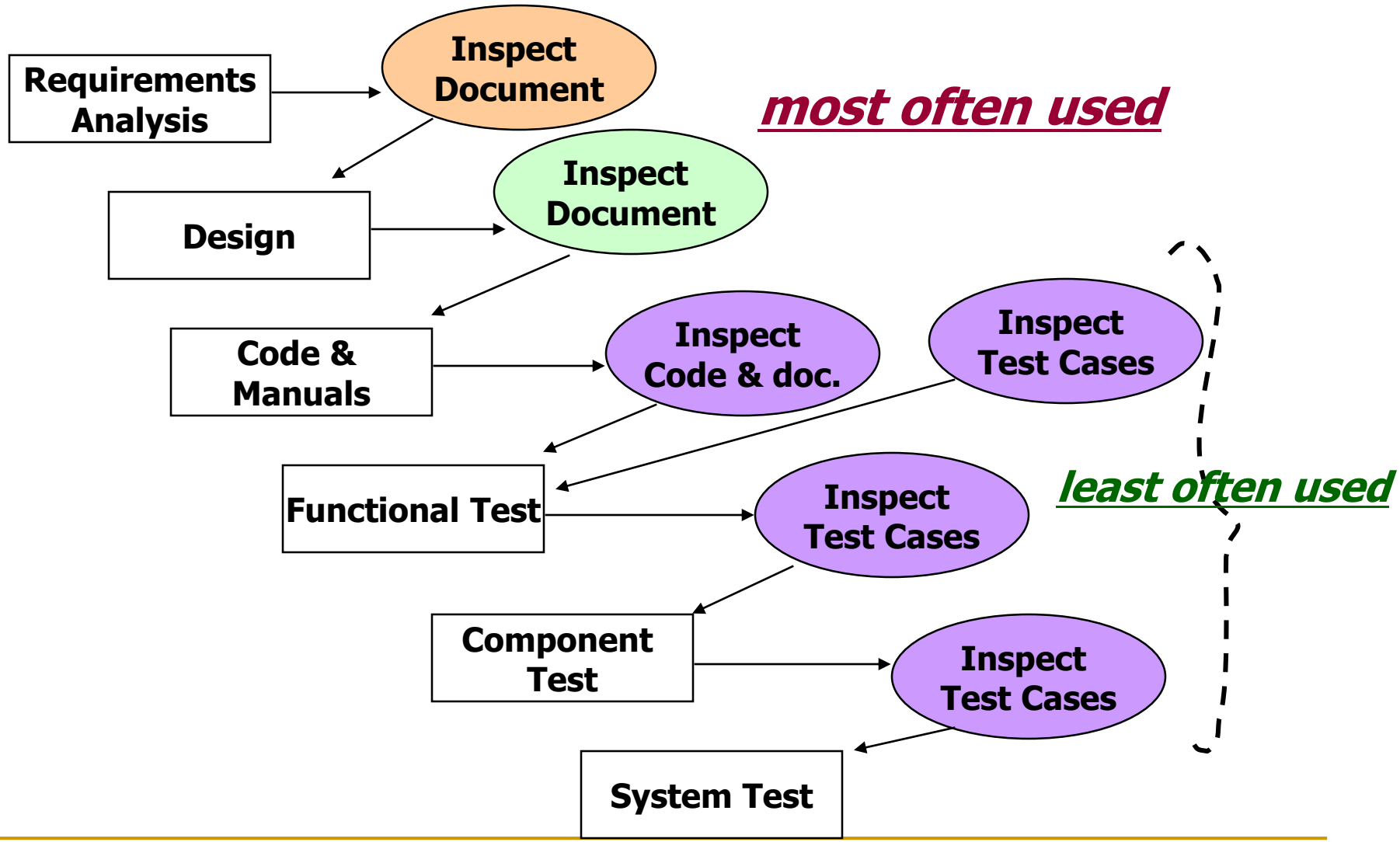
Formal Technical Review (or Inspection)

- The main objective is to detect defects that escape the originator more than other practitioners
- Inspections and reviews are testing of *software artifacts* without the actual execution of code and is especially suited for:
  1. Requirements documents
  2. Design documents
  3. Plans and Tests Cases
- Have been shown to be up to 75% effective in uncovering design flaws (which constitute 50-65% of all errors in software)

Formal Technical Review FTR (or Inspection)

- Objectives
  - To uncover errors in function, logic, or implementation for any representation of the software
  - To verify that the software under review meets its requirements
  - To ensure that the software has been represented according to predefined standards
  - To achieve software that is developed in a uniform manner
  - To make projects more manageable
- Serves as a training ground for junior software engineers to observe different approaches to software analysis, design, and construction
- Promotes backup and continuity because a number of people become familiar with other parts of the software

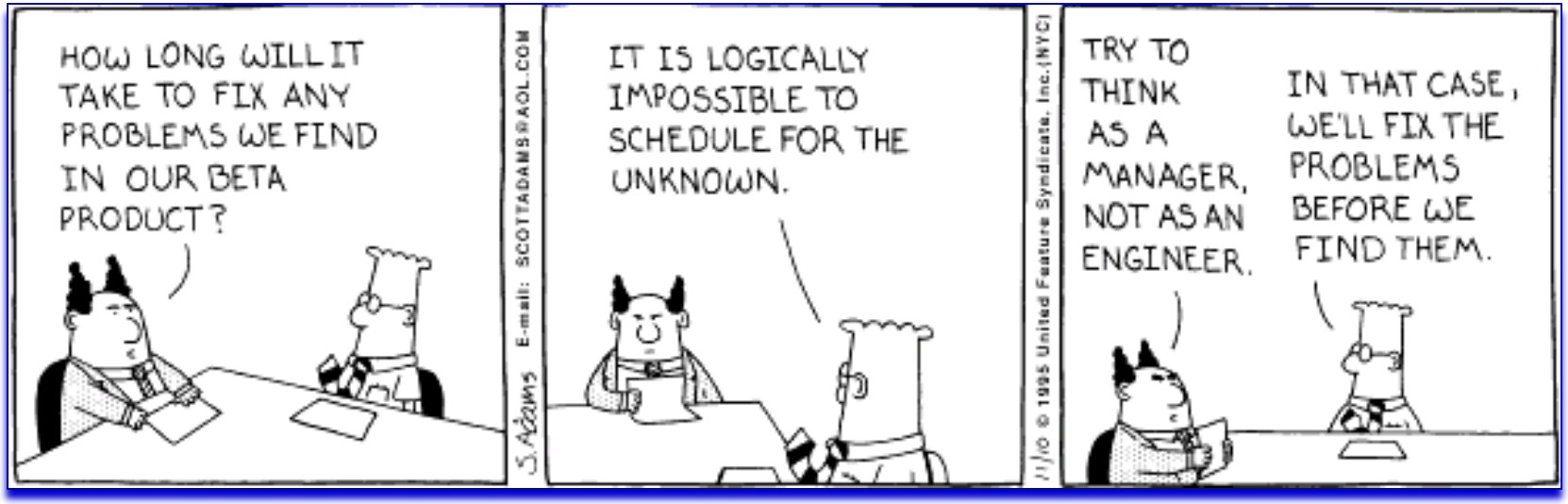
Inspection as *Natural* Part of Software Development



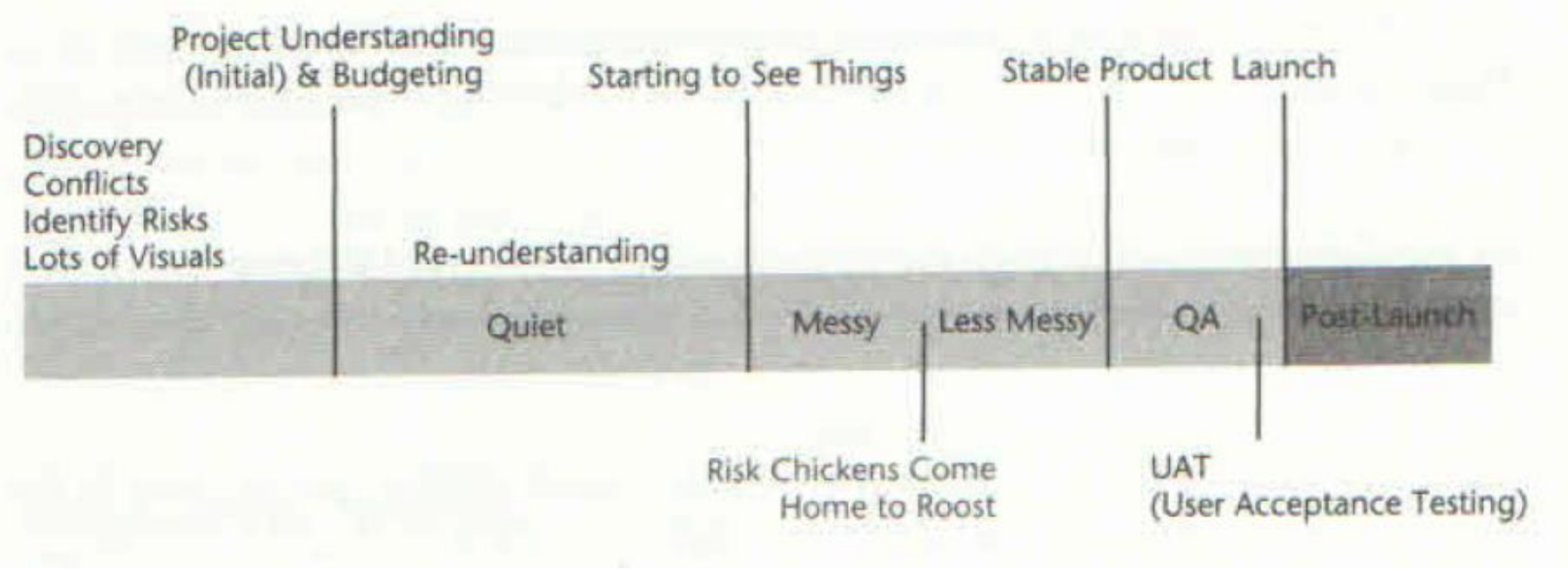
FTR/Inspection Benefits

- Quality : Find and correct defects
- Productivity : Reduce time-to-market
- Efficiency : Correct defects at their point of origin
- Cost :
  - up to 1,000 times cheaper to correct an error during analysis than during test
  - up to 40,000 times if the defect is correct before deployment!

• • •


HOW LONG WILL IT TAKE TO FIX ANY PROBLEMS WE FIND IN OUR BETA PRODUCT?
S. Adams E-mail: SCOTTADAMS@AOL.COM
IT IS LOGICALLY IMPOSSIBLE TO SCHEDULE FOR THE UNKNOWN.
11/10 © 1995 United Feature Syndicate, Inc. (NYC)
TRY TO THINK AS A MANAGER, NOT AS AN ENGINEER.
IN THAT CASE, WE'LL FIX THE PROBLEMS BEFORE WE FIND THEM.

Summary of Final Third



Agenda

- Project Execution phase
- Testing
- Quality Assurance
- **Measuring and Controlling Work Products & Processes**
  - Definitions, GQM
  - Success Metrics (Key Performance Indicators (KPI))
  - Examples
- Plan the measurement process

Four Kinds of Project Management Activities

1. **Plan and Estimate**

   tasks, schedule, budget, resources

2. **Measure and Control**

   work products (quantity & quality)

   schedule, budget, resources, progress

3. **Communicate and Coordinate**

   help people do their work activities

   represent the project to others

4. **Manage Risk**

   Identify and confront risk factors

Measurement Process

the process for establishing, planning, performing and evaluating software measurement within an overall project or organisational measurement structure that meets the specific needs of software organisations and projects. [**ISO/IEC 15939]**

Measuring and Controlling

- Measuring is concerned with
  1) collecting,
  2) validating, and
  3) analyzing

  project status information

Measuring and Controlling

- Controlling is concerned with applying corrective action when actual status does not conform to planned status
  - status of the work products
    - quantity and quality of work products
  - status of the development process
    - schedule, budget, resources, risk factors

What Should be Measured and Controlled?

Some level of measurement and control for each of the following attributes is important to assure a successful outcome for every project:

- **effort:** amount of work expended for various work activities
- **schedule:** achievement of objectively measured milestones
- **cost:** expenditures for various kinds of resources, including effort
- **progress**: work products completed, accepted, and baselined
- **Defects:** quality trends

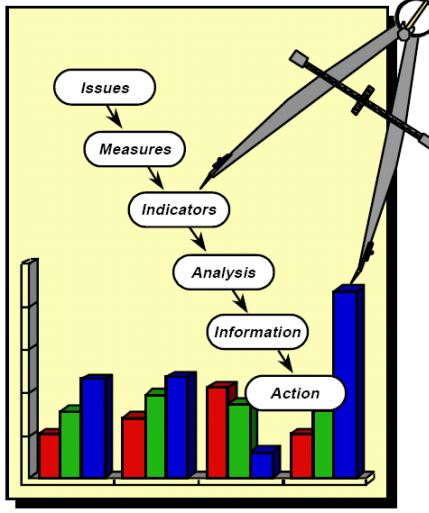
Success metrics

How the company will know if the project is successful?

- Success metrics have to be derived from the business goals and stated in testable terms
- product and process success metrics are, or should be, a byproduct of:
  - The procedures, tools, and techniques used to develop software

Software measurement techniques for establishing success metrics: Goal-Question-(Indicator)-Metric GQM

Goal Question (Indicator) Metrics (GQ(I) M)



GQ(I)M Approach:

Set goals specific to the business needs

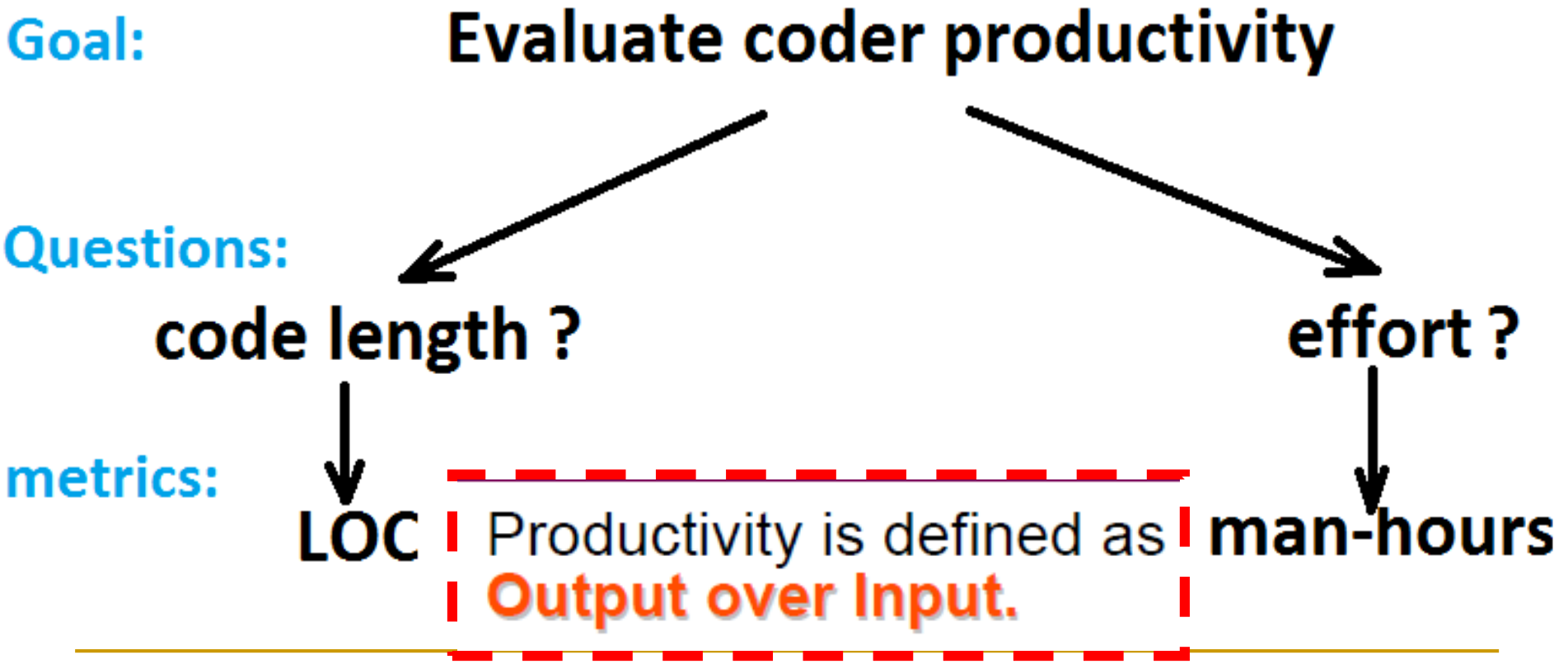
1. Refine the goals into quantifiable questions that are tractable.
2. Deduce the success metrics and data to be collected (and the means for collecting them) to answer the questions.

- GQM ensures that Measurement is Relevant to an Organization.
- GQM helps Managers Focus on Metrics that Relate to Project and Organizational Goals

Goal-Question-Metric Approach

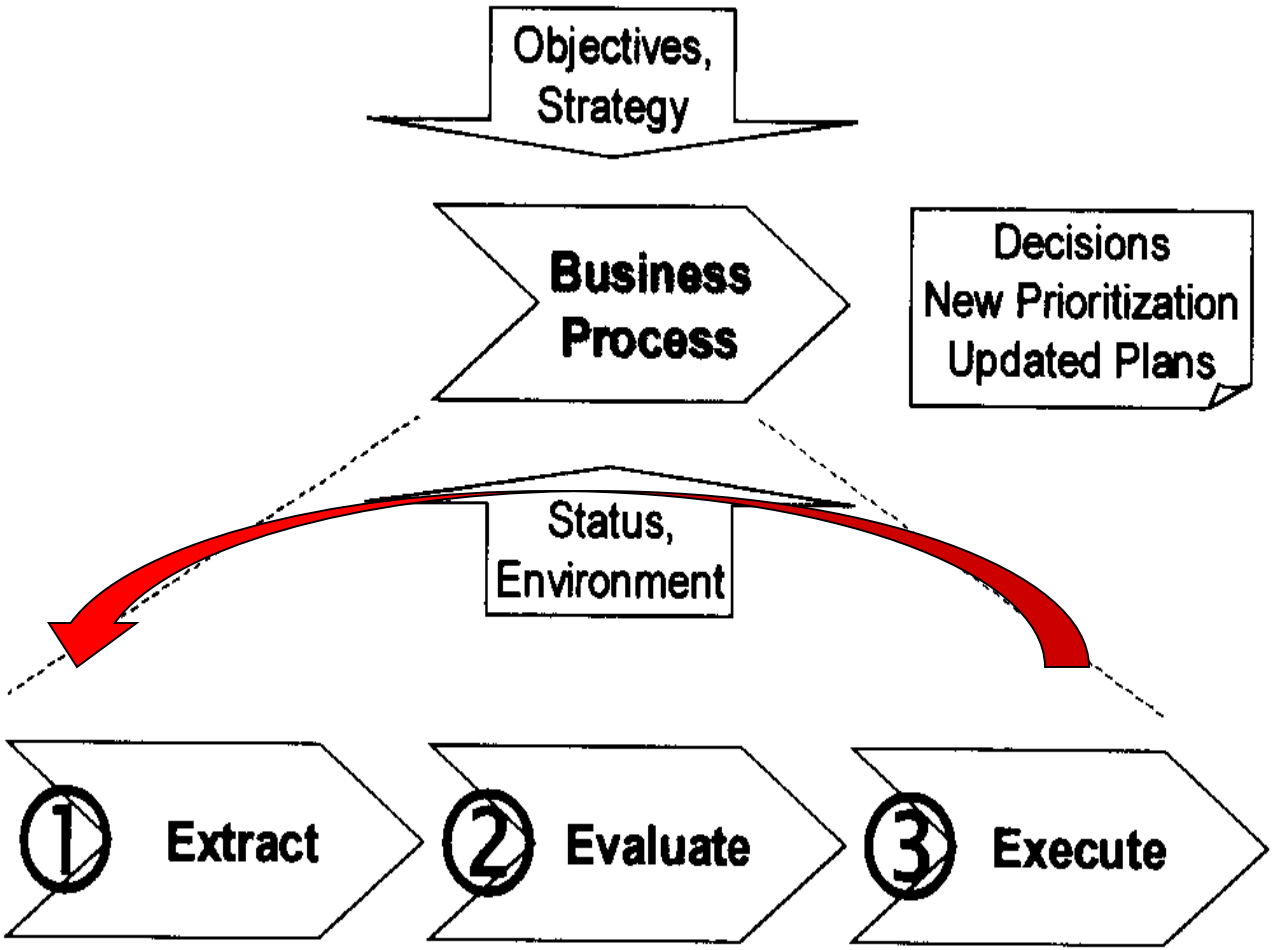
- Primary question in goal-based measurement: *"What do we want to know or learn?"* instead of *"What metrics should we use?"*
- Because the answers depend on your goals, no fixed set of measures is universally appropriate.

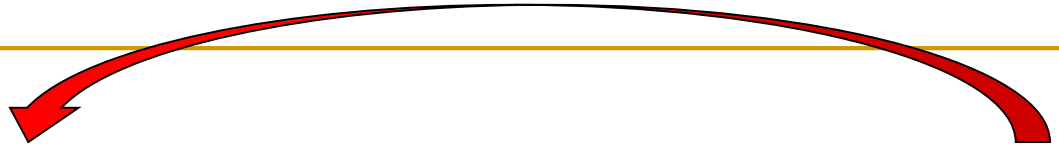
GQM Approach: productivity analysis (RE: ISO15939)



Information Need	Estimate productivity of future project
Measurable Concept	Project productivity
Relevant Entities	1. Code produced by past projects 2. Effort expended by past projects
Attributes	1. C++ language statements (in code) 2. Timecard entries (recording effort)
Base Measures	1. Project X Lines of code 2. Project X Hours of effort
Measurement Method	1. Count semicolons in Project X code 2. Add timecard entries together for Project X
Type of Measurement Method	1. Objective 2. Objective
Scale	1. Integers from zero to infinity 2. Real numbers from zero to infinity
Type of Scale	1. Ratio 2. Ratio
Unit of Measurement	1. Line 2. Hour
Derived Measure	Project X Productivity
Measurement Function	Divide Project X Lines of Code by Project X Hours of Effort
Indicator	Average productivity
Model	Compute mean and standard deviation of all project productivity values
Decision Criteria	Computed confidence limits based on the standard deviation indicate the likelihood that an actual result close to the average productivity will be achieved. Very wide confidence limits suggest a potentially large departure and the need for contingency planning to deal with this outcome.

How the company will know if the project is successful?





measurement feedback

- **Extract**
  - collect measurement data for a specific need and record it for potential further usage. The collected data shall be stored, including any context information necessary to verify, understand, or evaluate the data.
- **Evaluate**
  - The collected data shall be analyzed statistically. The data analysis results shall be interpreted.
- **Execute**
  - Communicate results
  - Execute decisions and actions to reduce the differences between status and objectives.

Management decisions are directly linked to the specific needs of the stakeholders!

GQ(I)M approach: Quality example (RE: ISO15939)

A.3.2 A quality example

The decision-maker in this example (Figure A.3) needs to evaluate detailed design quality as the design is being produced. The measurable concept is that design quality is related to the amount of design produced and the number of defects found. Thus, the design packages and the lists of defects are the entities of concern. Quality of design packages can be normalized by computing defect rate. Thus, data for the base measures (number entries in table below) is collected and the derived measure computed for each package as it is reviewed.

Since we do not really expect to get exactly the same defect rate for every package, we can compute control limits to determine if the defect rate on any package is different enough from the average to warrant concern.

Illustrating the measurement feedback loop: Defect Density measurement and analysis (revisited)

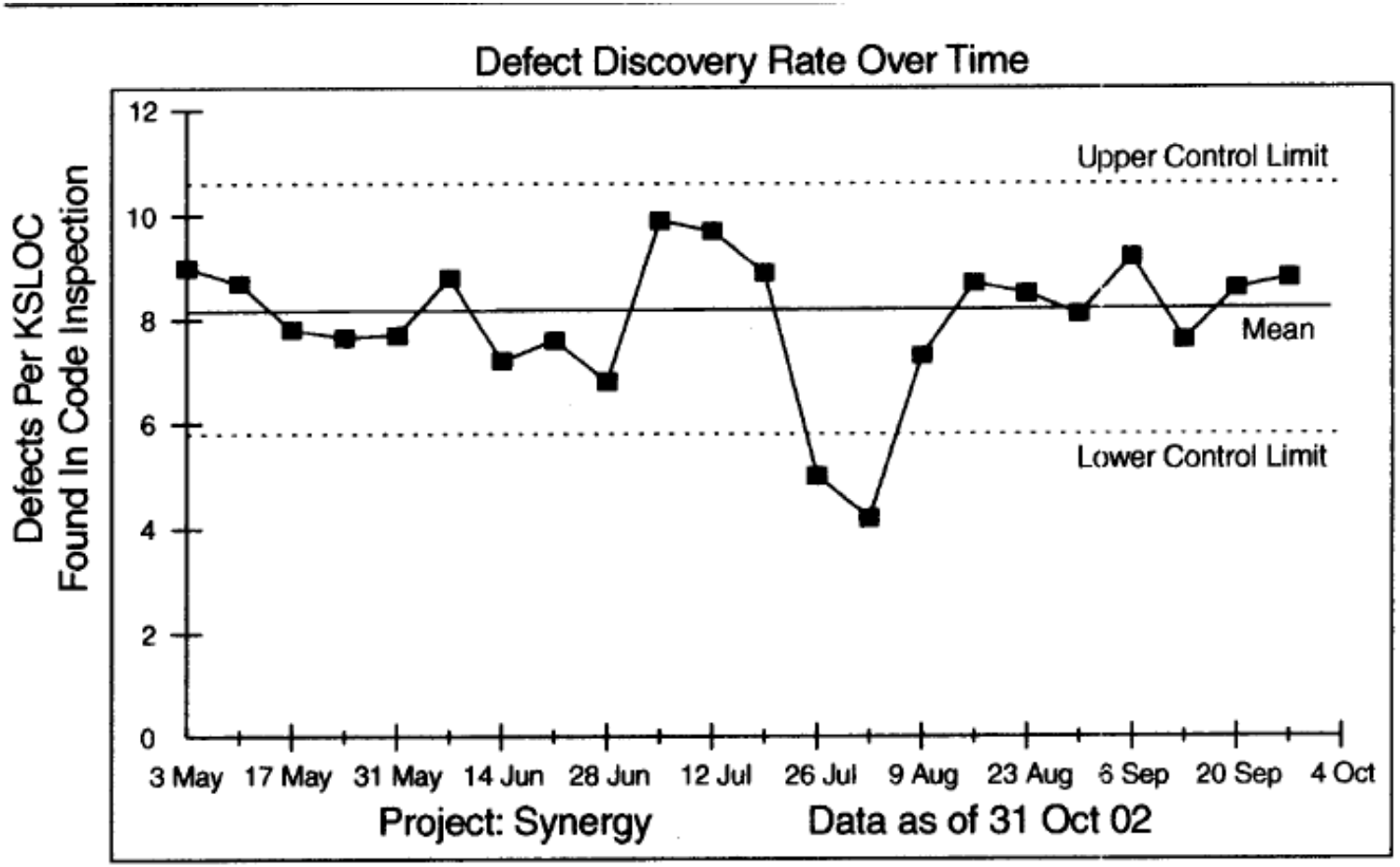
- **Extract data:** total defects for package X, package X size
  - **Calculate defect density:**

defect density for package X =

(total defects for package X) / (package X size)

- **Evaluate (analyze the data):**
  - Compute process centre and control limits using values of defect density
- **Execute (Decision-making):**
  - Results outside the control limits require further investigations

Defect Density Indicator used for decision making



GQ(I)M approach: Quality example (RE: ISO15939)

Information Need	Evaluate product quality during design
Measurable Concept	Product quality
Relevant Entities	1. Design packages 2. Design inspection reports
Attributes	1. Text of inspection packages 2. Lists of defects found in inspections
Base Measures	1. Package X size 2. Total defects for package X
Measurement Method	1. Count number of lines of text for each package 2. Count number of defects listed in each report
Type of Measurement Method	1. Objective 2. Objective
Scale	1. Integers from zero to infinity 2. Integers from zero to infinity
Type of Scale	1. Ratio 2. Ratio
Unit of Measurement	1. Lines 2. Defects
Derived Measure	Inspection defect density
Measurement Function	Divide Total Defects by Package Size for each package
Indicator	Design defect density
Model	Compute process center and control limits using values of defect density
Decision Criteria	Results outside the control limits require further investigations

Key point: Defects and Quality

- A defect that produces a failure results in loss of product quality, which may be manifest as a:
  - safety failure
  - security failure
  - reliability failure
  - availability failure
  - usability failure
  - maintainability failure

An Example of Defect Tracking

defect kind: \ phase in which defects found:	Rqmts	Design	Imple.	Verif.	Valid.	Ops	Totals
Rqmts	50	25	13	6	3	3	100
Design		60	30	15	8	7	120
Imple.			80	40	20	10	150
Verif.				6	3	0	9
Valid.					7	0	7
Totals:	50	85	123	67	41	20	386

Agenda

- Project Execution phase
- Testing
- Quality Assurance
- Measuring and Controlling Work Products & Processes
  - Definitions, GQM
  - Success Metrics (Key Performance Indicators (KPI))
  - Examples
- **Plan the measurement process**

Plan the measurement process
(Chapter 7, section 6.2, SWEBOK)

- Characterize the organizational unit
  - Application domain, org. structure and processes, technology…
- Identify information needs
  - Goals, problems, constraints
- Select success metrics
- Define data collection, analysis, and reporting procedures
  - Storage, scheduled, verification, stats, reporting
- Provide resources for measurement tasks

SIMPLE SOFTWARE MEASUREMENT PLAN EXAMPLE

Step 1 summary

- Commonwealth Software, Inc. (CSI)
  - They use Agile methodology.
  - There are 6 sprints per release, 2 weeks/each.
- GQM approach
- Perspective:
  - Project Manager for Development
- Business Goal for the project:

Deliver high quality products in each release.

Derived Measurement goal:

"Increase the quality of the deliverables in each sprint"

Quantifiable questions:

1. What is the defect density trend from sprint to sprint?
2. What is the percentage of issues which is not fixed after each sprint?

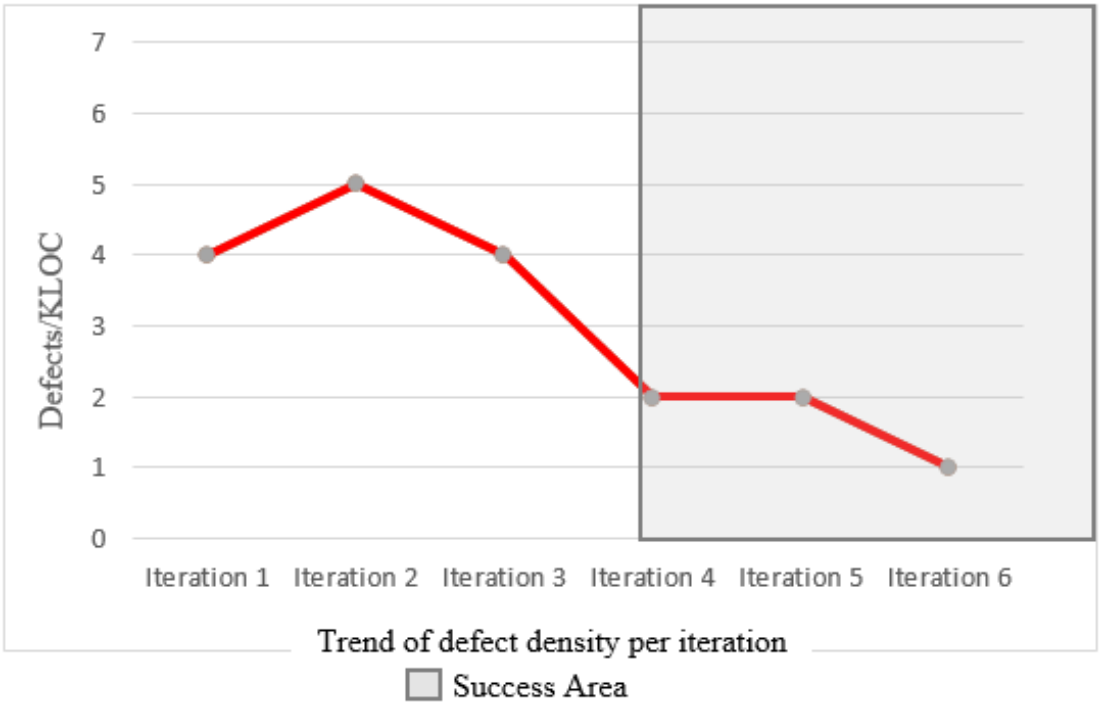
Step 2. Success criteria & Indicators

Q2. What is the defect density trend from from sprint to sprint?

Success criteria 1 & Indicator 1:

Success criteria 1: Defect Density (DD) should be less than two defects in each sprint

Indicator 1: (IN1)



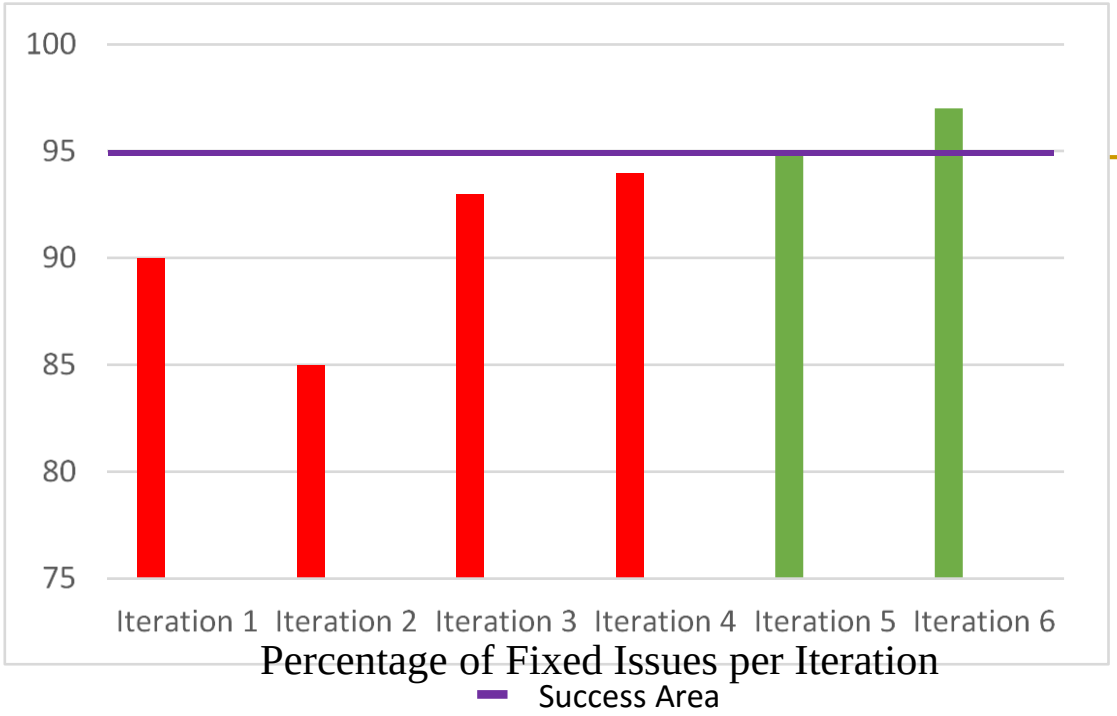
Step 2. Success criteria & Indicators

Q3. What is the percentage of issues which are not fixed after each sprint?

Success criteria 2 and Indicator 2:

Success criteria 2: The percentage of fixed issues per sprint should be more than 95%

Indicator 2: (IN2)



Step 3. Identifying the Data Elements to be Collected & their Availability

Data Elements Required	Indicators		Data Element ID
	IN1	IN2	
# SLOC (source line of code) per iteration.	X		D1
Total number of defects per iteration.	X		D2
Total number of fixed issues per iteration.		X	D3
Total number of open issues per iteration.		X	D4

Step 3. Identifying the Data Elements to be Collected & their Availability

Data Element	Availability	Source
# SLOC (source line of code) per iteration (D1)	Yes	Source code in software repository
# Total Defect per iteration (D2)	Yes	Inspection reports, issue tracker, testing reports.
Total number of fixed issues per iteration (D3)	Yes	Issue Tracker
Total number of open issues per iteration (d4)	Yes	

Step#4. Planning Tasks

Work Breakdown Structure (WBS) of the Measurement Tasks and their integration with the planned iterations			
Iteration No	Tasks in each iteration	Start	Finish
1	1. Collect data 2. Analyze data 3. Prepare reports	21/02/2020	21/02/2020
2	1. Collect data 2. Analyze data 3. Prepare reports	03/03/2020	03/03/2020
3	1. Collect data 2. Analyze data 3. Prepare reports	17/03/2016	17/03/2020
4	1. Collect data 2. Analyze data 3. Prepare reports	31/03/2020	31/03/2020
5	1. Collect data 2. Analyze data 3. Prepare reports	14/04/2020	14/04/2020
6	1. Collect data 2. Analyze data 3. Prepare reports	28/04/2020	28/04/2020

Summary

- Project Execution phase
- Testing
- Quality Assurance
- Measuring and Controlling Work Products & Processes
  - Definitions, GQM
  - Success Metrics (Key Performance Indicators (KPI))
  - Examples
- Plan the measurement process