

# A Goal-Based Framework For Software Measurement

# Objective

- Classification of measures as process , product or resource ,and identify the attribute is internal or external.
- Goal-Question-Metric(GQM)-measurement to the overall goals of the project and process.

# Goal Based Measurement

- ■ The 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 metrics is universally appropriate.
- ■ Instead of attempting to develop general-purpose measures, one has to describe an adaptable process that users can use to identify and define measures that provide insights into their own development problem.

# GBM Process

- **Determining what to measure**
  - Identifying entities
  - Classifying entries to be examined
  - Determining relevant goal
- **Determining how to measure**
  - Inquire about metrics
  - Assign metrics

## GQM Metrics Definition

### Conceptual Level

Goals identify what we want to accomplish relative to products, processes or resources

### Operational Level

Questions help us understand how to meet the goal. They address the context of a quality issue from a particular viewpoint

### Quantitative Level

Metrics identify the measurements that are needed to answer the questions.



# Identifying Entities

- Process
- Product
- Resources

# Classifying Software Measures

In software there are three such classes

- **Processes:** Collection of software-related activities.
- **Products:** Artifacts, deliverables or documents that result from a process activity.
- **Resources:** Entities required by a process activity.

# Internal and External Attributes

- An internal attribute can be measured by examining the **product, process, or resource** on its own, separate **from its behavior**. (Program size, complexity, dependencies).
- External attributes are those that can be measured only with respect to how the product, process or resource **relates to the environment**.  
(Experienced failures, timing and performance)



# Processes

- We often have questions about our **software-development activities and processes that measurement** can help us to answer.
- We want to know how long it takes for a process to **complete**, how much it will **cost**, whether it is **effective or efficient**, and **how it compares with other processes** that we could've chosen.

Example: AT&T developers wanted to know the effectiveness of their software inspections. In particular, managers needed to evaluate the cost of inspections against benefits received. To do this, they measured the average amount of effort expended per thousand lines of code reviewed. This information combined with measures of the number of faults discovered during the inspections, allowed managers to perform a cost-benefit analysis.

# Internal and External Attributes

## Internal

- Size, Effort, Cost
- Code Complexity
- Functionality
- Modularity
- Redundancy
- Syntactic Correctness
- Reuse

## External

- Usability
- Integrity
- Efficiency
- Testability
- Reusability
- Portability
- Interoperability

# Internal Process Attribute Directly Measured by

- Internal process attribute directly measured by
  - the **duration** of a process or one of its activities;
  - the **effort** associated with the process or one of its activities;
  - the **number of incidents** of a specified type arising during the process or one of its activities.
    - Eg:-Ave. cost of error= $\text{cost}/\text{\#errors\_found}$ .

**Table 3.1:** Components of software measurement

ENTITIES	ATTRIBUTES	
<i>Products</i>	<i>Internal</i>	<i>External</i>
Specifications	size, reuse, modularity, redundancy, functionality, syntactic correctness, ...	comprehensibility, maintainability, ...
Designs	size, reuse, modularity, coupling, cohesiveness, functionality, ...	quality, complexity, maintainability, ...
Code	size, reuse, modularity, coupling, functionality, algorithmic complexity, control-flow structuredness, ...	reliability, usability, maintainability, ...
Test data	size, coverage level, ...	quality, ...
...	...	...
<i>Processes</i>		
Constructing specification	time, effort, number of requirements changes, ...	quality, cost, stability, ...
Detailed design	time, effort, number of specification faults found, ...	cost, cost-effectiveness, ...
...		
Testing	time, effort, number of coding faults found, ...	cost, cost-effectiveness, stability, ...
...	...	...
<i>Resources</i>		
Personnel	age, price, ...	productivity, experience, intelligence, ...
Teams	size, communication level, structuredness, ...	productivity, quality, ...
Software	price, size, ...	usability, reliability, ...
Hardware	price, speed, memory size, ...	reliability, ...
Offices	size, temperature, light, ...	comfort, quality, ...
...	...	...

# Products

- Products are not restricted to the items that management is committed to **deliver to the customer**. Any artifact or document produced during the software life cycle can be measured and assessed. For example developers often build **prototypes** for examination only, so that they can understand requirements or evaluate possible designs; **these prototypes may be measured in some way**.

# Products

- External product attributes depend on both **product behavior and environment**, each attribute measure should take these characteristics into account.
- External attributes:
  - Reliability, maintainability, understandability (of documentation), usability, integrity, efficiency, reusability, portability, interoperability...
- Internal product attributes are sometimes **easy to measure**. We can determine the size of a product by measuring the **number of pages it fills or the number of words** it contains. Other internal product attributes are more **difficult to measure**, because opinions differ as to what they mean and how to measure them. For example the **complexity of codes**.
- Internal attributes
  - Size, effort, cost, functionality, modularity, syntactic correctness.

# Product Measurements

- Direct measure example:
  - Entity: Module design document (D1)
  - Attribute: Size
  - Measure: No. of bubbles (in flow diagram).
- Indirect measure example:
  - Entity: Module design document (D1, D2,...)
  - Attribute: Average module size
  - Measure: Average no. of bubbles (in flow diagram).

### 3.1.2.3.The Importance of Internal Attributes

- Many software engineering methods proposed and developed in the last 25 years provide rules, tools, and any heuristics for providing software products. It is claimed that this structure **makes them easier to understand, and tests.**
- It is assumed that good internal structure leads to a good external quality. This connection has rarely been established.



### 3.1.2.4. Internal Attributes and Quality Control and Assurance

- Major reason **developer want to use internal attribute to predict external ones** to monitor and control the product during development.
- Eg-identify the modules at design stage shows that they are likely to be **error prone or difficult** to maintain or test later on

### 3.1.2.5. Validating Composite Measures

- Measure the control and **quality of ours product** and usually do so by measuring and controlling a number of internal product.

# Resources

- The resources that we are likely to measure include **any input for software production.**
- Resources include **personnel, materials, tools and methods.** Resources are measured to determine their **magnitude, cost and quality.**
- Cost is often measured across all types of resources, so that managers can see how the cost of inputs affects the cost of the outputs.
- Resource measure combines a process measure (input) with a product measure (output).


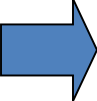
# Resources

- Personnel (individual or team), materials (including office supplies), tools, methods.
  - Resource measurement may show what resource to blame for poor quality.
- Cost measured across all types or resources.
- Productivity:
  - $\text{amount\_of\_output/effort\_input}$ .
  - Combines resource measure (input) with the product measure (output).

## 3.2.Determining what to measure

- A particular measurement is useful only if it helps you to understand the underlying process or one of its resultant products.
- In turn, recognizing improvement of the process and product can occur only when the project has clearly defined goals for process and products.

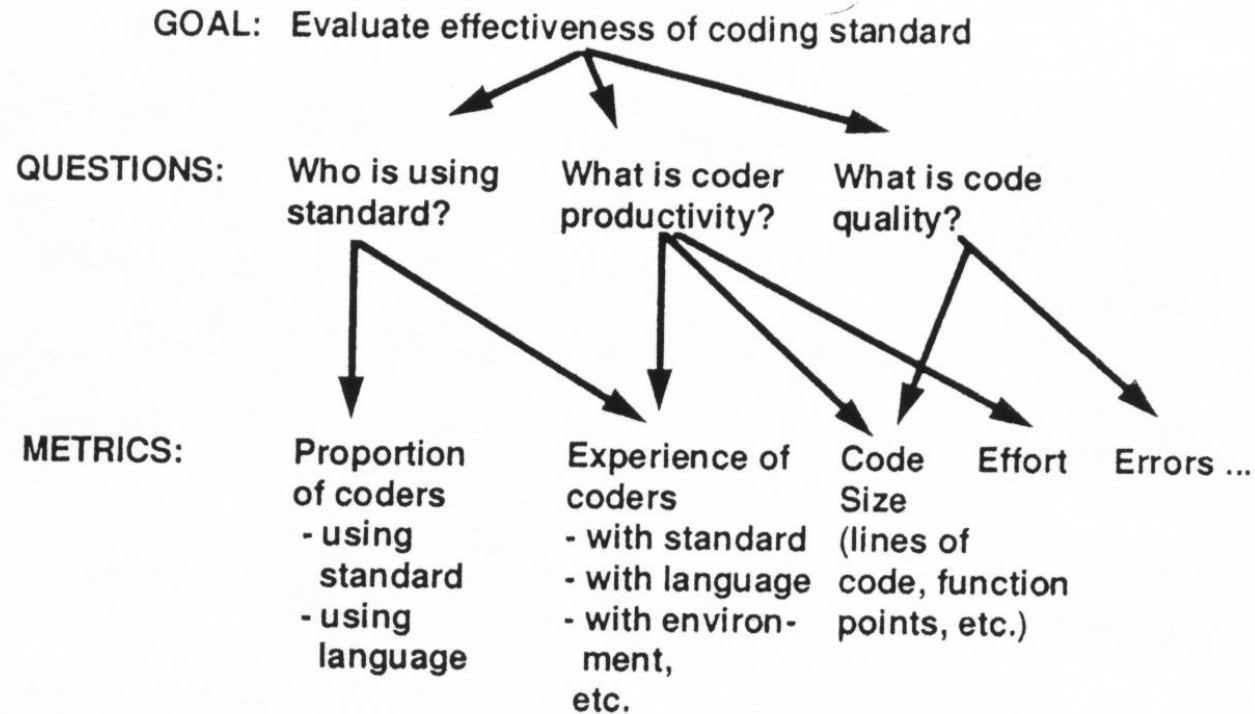
# Goal-Question-Metric

- The GQM approach to process and metrics has proven to be a particularly effective approach to **selecting and implementing the metrics**.
- To use GQM, You express the overall goals of your organization  Ask relevant questions Measure. 

# GQM Paradigm

- Goal-Question-Metric
  - Many metrics begin by what is convenient or easy to measure rather than what is needed-such program not useful for developer and maintainer.
  - SO GQM approach provides framework involves 3steps
- Steps:
  - List the **major goals** of the development effort.
  - Derive **from each goal questions** that must be answered to determine if goals are being met.
  - Decide what must be measured to **answer the questions adequately.**

# Generic GQM Example



**Figure 3.2:** Example of deriving metrics from goals and questions



# Templates for goal definition

- Purpose:
  - To (**characterize**, evaluate, predict...) the (**process**, model, metric...) in order to (**understand**, assess, manage, learn, improve...)
- Perspective
  - Examine the (**cost**, correctness, defects, changes...) from the viewpoint of (**developer**, manager, customer...)
- Environment
  - The environment consists of process factors, **people factors, methods, tools, etc.**

# GQ(I)M Process

- 10 steps of the **GQIM** process

A home brewed tool to automate this process  
exists(ISMS)

# 1. Identify Business goal

- The business goal can be initiated by any **organizational level**.  
Eg: Improve customer satisfaction  
Improve quality of the code

## 2. Identify What to Know

- Identifying what is needed to be known in order to **understand, assess, predict, or improve the activities** related to achieving goals by asking questions such as:
  - **“What activities do we manage or execute?”**  
**“What do we want to achieve or improve?”**
- Repeat these questions several times and break top-level goals down into specific things should be accomplished and issues that are needed to be addressed.
- Eg:- understand , predict the material

# 3. Identify Sub goals

- Grouping related questions helps identify sub goal . In Step 3, you identify the questions that you have about the entities, **then group them and identify the issues they address.**
- Then groupings of issues and questions translate naturally into candidate sub goals.
- Eg:-
  - Improve performance of the staff
  - Improve code development process
  - Improve quality assurance

# 4. Identify Entities & Attribute

- Once having a list of questions, you should examine each question and identify entities implicit in it. Then list pertinent attributes associated with each entity.
- Eg :- **Subgoal1:** Improve the performance of our staff  
**Question1 :** How is the overall office's morale(confidence&courage)?  
**Entity:** Working Environment
- Attributes: Accommodation ,Incentives,Hardware and software used, Workspace(room ,desk etc)

# Subgoal

- **Improve quality assurance**
- **Question 1:**
  - Are we using manual testing or automated testing?
- **Entity:**
  - Testing method
- **Attributes:**
  - Type Name of the type
  - Is used? Ys/no

# 5. Formalize Measurement

- A measurement goal (or sub goal) is a **semi-formal** representation of a business goal (or sub goal), **composed of 4 components**
  - An object of interest (entity)
  - A purpose
  - A perspective
  - A description of environment & constraints



# Active & Passive Measurement

- Active Goals(controlling and causes changes)
  - Meet the scheduled completion date
  - Reduce variability
  - Improve product reliability
  - Improve time-to-market
  - Reduce employee turn over
- Passive Goals(Learning and Understanding)
  - Understand the current development process
  - Identify root causes
  - Assess product maintainability
  - Identify capabilities and trends , so that we can predict the better future performance.

# 6. Identify Indicators

- What is an indicator?
  - Indicator is a display of one or more measurement results that is designed to communicate or explain the significance of those results to the user

## Why it is useful?

- ■ Seeing how measurement data will be displayed helps clarify exactly what must be measured.

# 7. Identify Data Elements

- Data elements that must be **collected to construct the indicators** identified in Step 6.
- Preparing list of data item(attribute)
- Definitions for data items including **scales, ranges and precision** will be added in the next step.

What data are required?	a	b	c	d	e
Total Money Available for Project	X				
Total Money Available for CASE Tools	X				
Lines of Source Code		X		X	
Development Effort		X		X	
Training Effort			X		
Type of CASE Tool			X		X
Average Cost per Line of Code				X	
Cost of CASE Tool				X	X

# 8. Define Measures

- **What is a measure definition?**

A measure definition is a semi-formal specification for the **object to be measured**.

- Why it is useful?

It is extremely useful to clarify the implicit assumptions, **what is included and what is not in the measurement**.

- Definitions must indicate:-

- Name and short description
- Scale
- Range of variation
- Precision required

- The same applies to definitions such as:

- The measure for **software size** is the number of non-commented, nonblank, executable source statements.

# 9. Identify Actions to Implement Measures

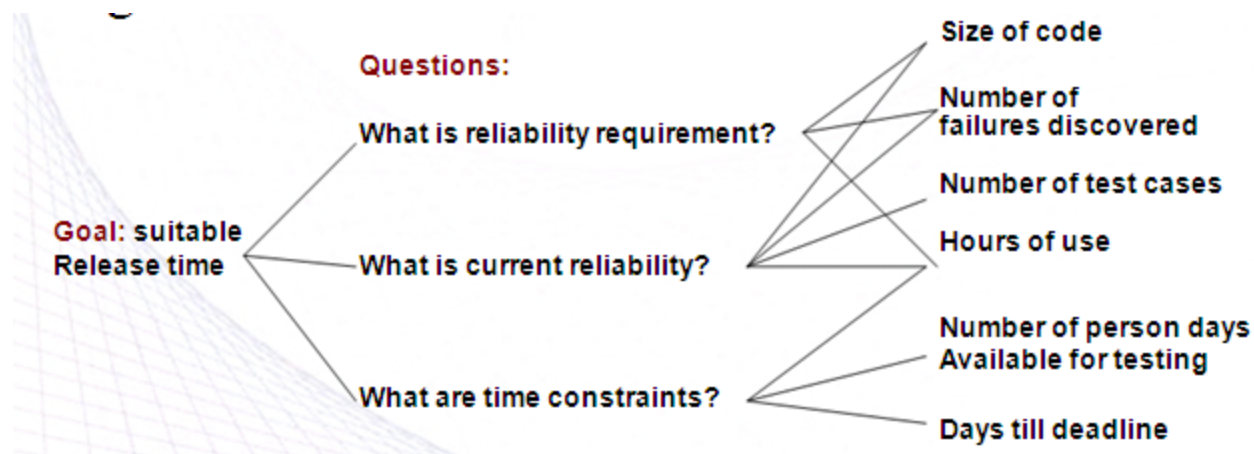
- This step is to assemble information about the **current status and use of the measures**, so as to prepare a **plan for implementing the measures** defined, through:
- Analysis-**identifying the measures that the organization is using** now and understanding how its collected
- Diagnose-**evaluate the data element** , determine how well they meet the needs of new requirement
- Actions-**translating the results of the analysis and diagnosis** into implementable steps.

# 10.Prepare a Plan

- Measurement implementation plan is based on analysis , diagnosis and actions(Step 9)
- Measurement plan                      template includes:
  - 1.Objective
  - 2.Description
  - 3.Implementation-action taken to implement the measures
  - 4.Sustained Operations

# GQM Example

- You are manager for a software development team and you have to decide upon the release time of your product. Construct a GQM tree related to this.



## 3.2.2.Measurement and Process Improvement

- Measurement can helps us to answer about the effectiveness of techniques or tools, and productivity , quality of products and more.
- It allows manager and developer to monitor the effects of activities and changes on development. So that we can control the final outcome . Its useful for
  - Understanding,
  - Establishing a baseline
  - Assessing and predicting



## 3.2.3. Combining GQM with process maturity

- High level goals
  - Improving productivity
  - Improving quality
  - Reducing risk
- Improving productivity as several sub goals affecting resources.
  - Ensuring adequate managerial skills.
  - Ensuring adequate staff skills.
  - Ensuring adequate host software engineering technology.
- Improve productivity using products can mean
  - Identifying problems early in the life cycle
  - Using appropriate technology
  - Reusing previously built products.

## 3.3.Applying the framework

- Product , process and resources are relevant in each case , which attribute are measuring (internal or external) and whether the focus is on assessment or prediction.

## 3.3.1. Cost and effort estimation

- It focus on prediction the attribute of cost and effort.
- Calculation involves based on past history , getting advice from expert judgements , etc...
- Prediction

## 3.3.2. Productivity measures and models

- We measure a resource attribute , such as personnel during particular processes.

### 3.3.3.Data collection

- Work involved in data collection concerned with how to set in place for gathering accurate and insistent measures of process and resources attributes.

### 3.3.4. Quality models and measures

- **Cost and productivity** depend on quality of product during various processes

## 3.3.5. Reliability models

- **Reliability** is a likelihood of successful operation during a given **period of time**.
- Reliability is most important **quality attribute**.

### 3.3.6. Performance evaluation and models

- Measuring **efficiency and product attribute**(time efficiency , space efficiency).
- Algorithmic algorithm:-high level algorithm determine **good predictor of efficiency of the implemented code , time efficiency**
- **predictor**



### 3.3.7. Structural and complexity metrics

- Single internal attribute representing complexity can be **accurate predictor** of many **external quality** attribute as well as process attribute like cost and effort.
- Predictor

### 3.3.8.Capability-maturity assessment

- Maturity score is viewed as an attribute of the contractor , rather than of the contractor's process.

# Determining what to measure

The SEI has suggested that there are five levels of process ranging from ad hoc, repeatable, defined, managed and optimizing.

The SEI distinguishes one level from another in terms of key process activities going on at each level.

## Overview of process maturity and measurement

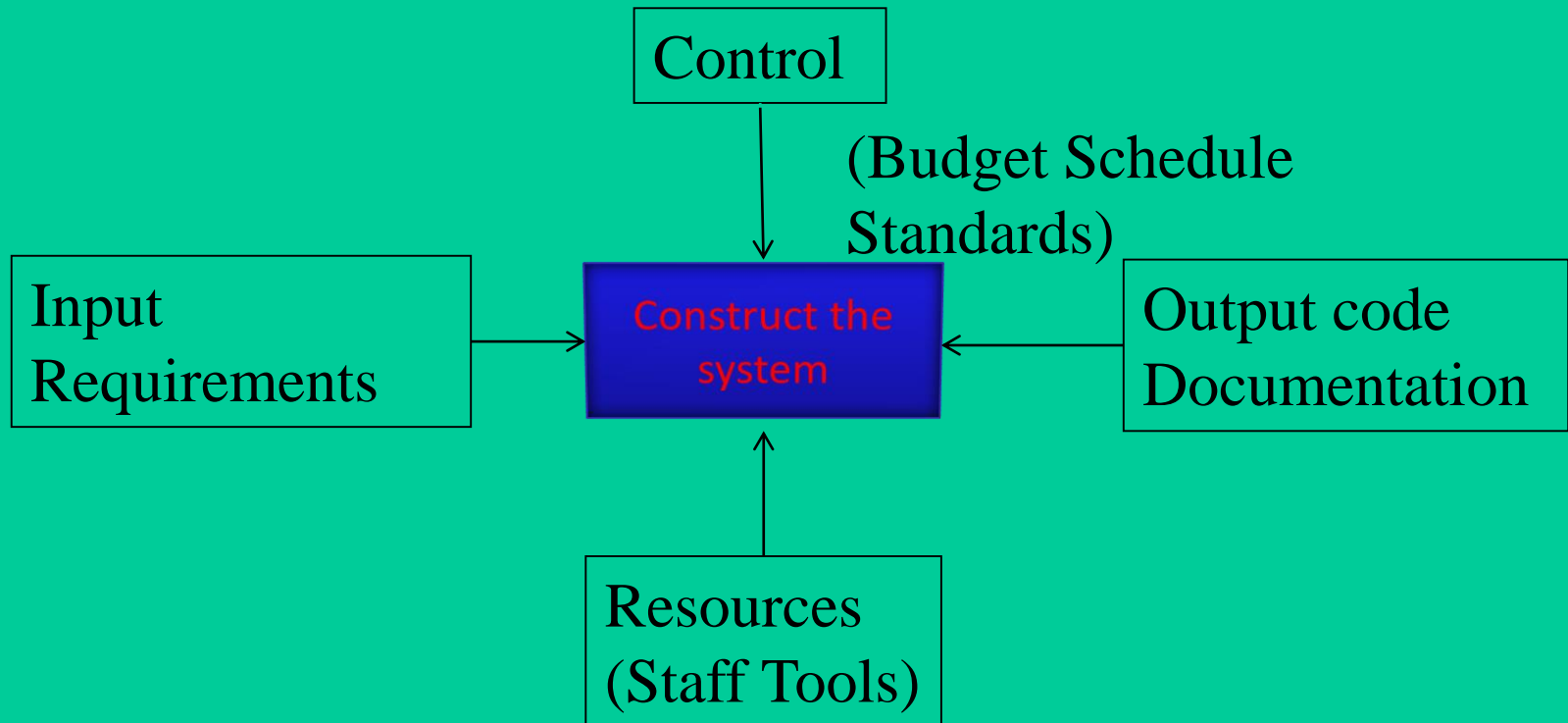
Maturity level	Characteristics	Type of metrics to use
5. Optimizing	Improvement feedback to the process	Process plus feedback for changing the process
4. Managed	Measured process	Process plus feedback for control
3. Defined	Process defined and institutionalized	Product
2. Repeatable	Process dependent on individuals	Project management
1. Initial	Ad hoc	Baseline

## **Level 1 : Initial**

- Inputs to the process are ill-defined; while outputs are expected, the transitions from inputs to outputs is undefined and uncontrolled.
- Similar projects may vary widely in their productivity and quality characteristics because of lack of adequate structure and control.
- Baseline measurements are needed to provide starting point for measuring improvement as maturity increases.

## Level 2: Repeatable

- It identifies the inputs and outputs of the process, the constraints and the resources used in the final product.
- The process is repeatable: proper inputs, proper outputs, but there is no visibility into how the outputs are produced

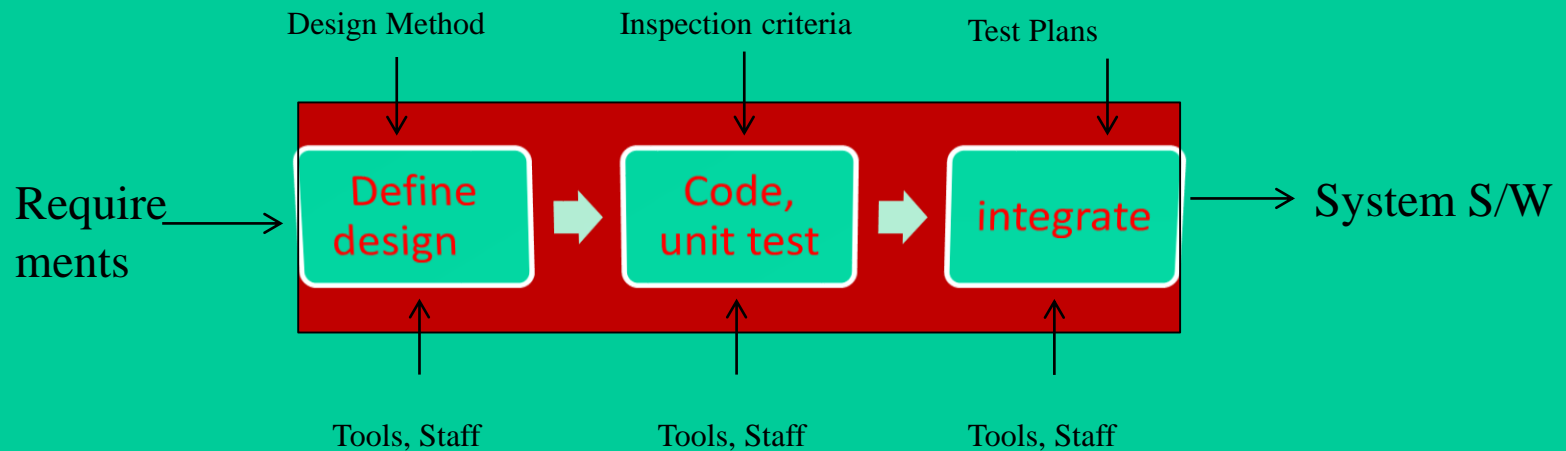


**A Repeatable process**

### Level 3 : Defined

- The defined process provides visibility into the “construct the system” box
- In this level the intermediate activities are defined and their inputs and outputs are known and understood.
- All process can be examined, measured and assessed.

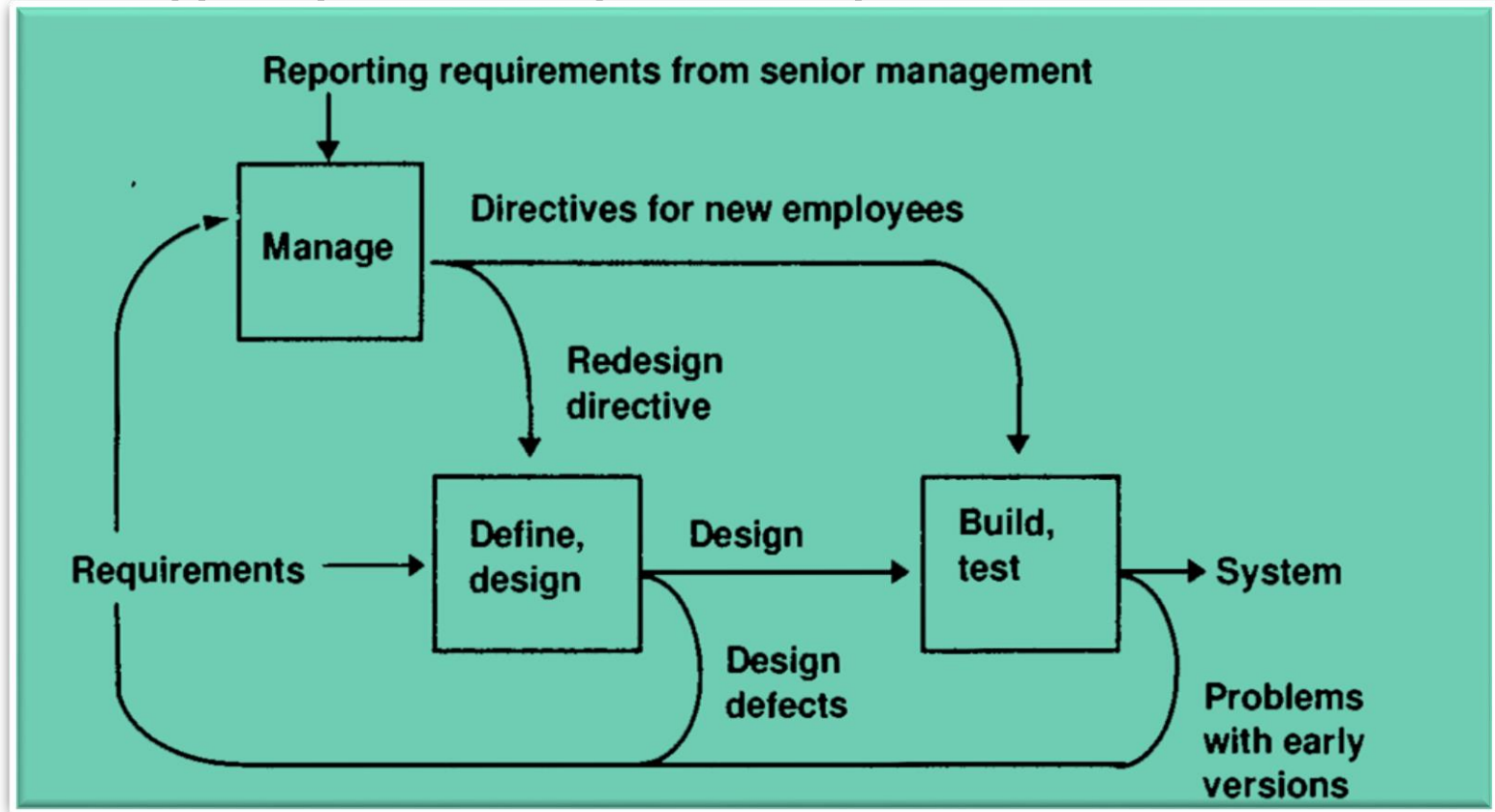
### A defined process



## Level 4: Managed

- A managed process adds management oversight to a defined process.
- You can compare contrast, the effects of changes in one activity can be traced in the others.
- The feedback determines how resources are deployed.
- You can evaluate the effectiveness of process activities: how effective are reviews? Configuration management? Quality assurance?.
- A significant difference between level 3 and 4 is that level 4 measurements reflects characteristics of the overall process and of the interaction among and across major activities.
- Management oversight relies on a metrics database that can provide information about such characteristics as distribution of defects, productivity and effectiveness of tasks, allocation of resources, and the likelihood that planned and actual values will match.

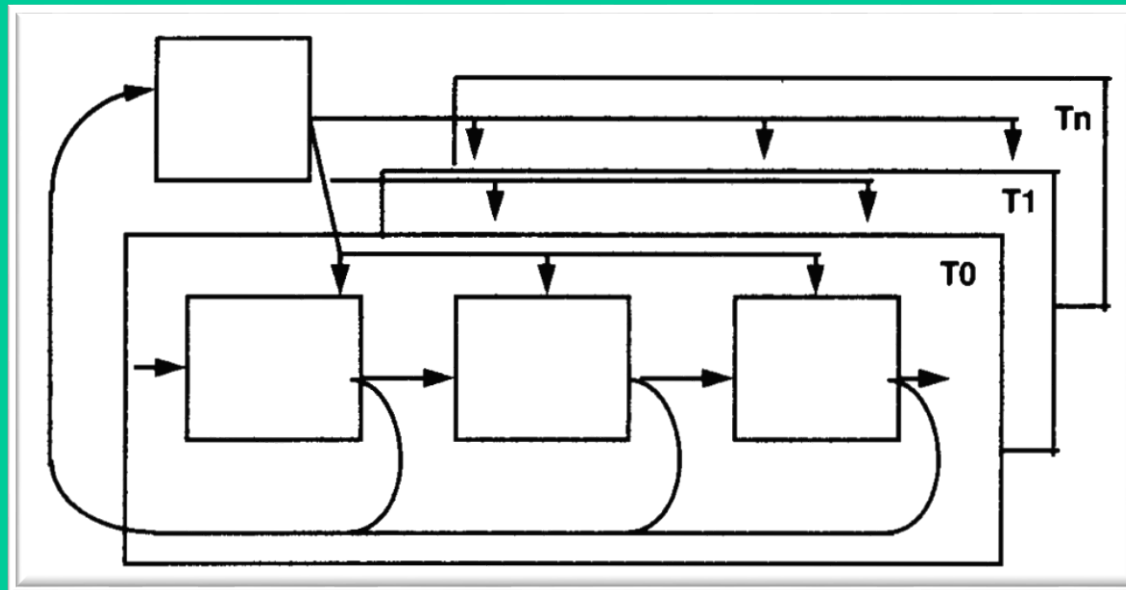
## A managed process (Level 4)





## Level 5 : Optimizing

- Here measure from activities are used to improve process, possibly by removing and adding process activities, and changing the structure dynamically in response to measurement feedback.



**An optimizing process (Level 5)**

## 3.9.Measurement by metrics

- Management use metrics to set targets for their development projects.
- Target derived from best practice.
- E.g.->Existing project

## 3.3.10.Measurement of Methods and Tools

- Investing new method or tools but hesitate because of uncertainty about cost , utility, effectiveness.
- Proposed tool tried in small project and result are evaluated and used for later project

## 3.11.A Mathematician View of Metrics

- In mathematical analysis ,a metric has a very specific meaning: it is a rule use to describe how far two points
- Metric is a function  $m$  defined on pairs of objects  $x$  and  $y$  such that  $m(x,y)$
- It should satisfy some properties:
  - $M(x,y)=0$
  - $M(x , y)=m(y , x)$
  - $M(x,z)<=m(x,y)+m(y,z)$

# Software Measurement Validation

Even when you know which entity and attribute you want to assess, there are many measures from which to choose.

- **Measures or measurement systems** are used to assess an existing entity by numerically characterizing one or more of its attribute.
- **Prediction systems** are used to predict some attribute of a future entity, involving a mathematical model with associated prediction procedures.

## 3.4.1. Validating prediction systems

- In a given environment is the process of establishing the accuracy of the prediction system by empirical means
- Two prediction system
  - Deterministic-get the same o/p for the given i/p
  - Stochastic-output for the a given input will vary probabilistically.

# Measurement vs. Prediction

- **Measurement systems** are used to assess existing entities by numerically characterizing one or more of its attributes.
- **Prediction systems** are used to predict some attributes of a future entity, involving a mathematical model with associated prediction procedures.
  - **Deterministic prediction system:** The same output will be generated for a given input.
  - **Stochastic prediction system:** The output for a given input will vary probabilistically. E.g., weather forecast.

# Measurement Validation

- Validation question: Does the measure used capture the information that it was intended for?
  - A measure is valid if it accurately characterizes the attribute it claims to measure.
  - A prediction system is valid if it makes accurate predictions.



# Validating Measures

- **Definition:** The process of ensuring that the measure is a proper numerical characterization of the claimed attribute by showing that the representation condition is satisfied.
  - **Example: Measuring program length**
    - Any measure of length should satisfy conditions such as:  
Length of joint programs should be

$$m(p_1 ; p_2) = m(p_1) + m(p_2)$$

If length of  $p_1$  is greater than  $p_2$ , any measure of length should satisfy  $m(p_1) > m(p_2)$

# Validating Prediction Systems

- **Definition:** The process of establishing the accuracy of the prediction system by empirical means, i.e., by comparing model performance with known data in the given environment.
  - **Example: Software Reliability**
    - Using tools such as CASRE to fit the failure data into the reliability model.
    - Accuracy of estimation of the failure intensity  $\lambda$  depends on the number of failures experienced (i.e., the sample size).
    - Good results in estimating failure intensity are generally experienced for programs with 5,000 or more developed source lines.

# Classes of Prediction Systems

- Class 1: measures of internal attributes of early **life-cycle products** predict measures of internal attributes of **later life-cycle products**
  - Example: Reuse of specification are used to predict size and final code.

# Classes of Prediction Systems

- Class 2: Measures of early life-cycle process attributes predict measures of later life-cycle processes and resources.
  - Example: number of faults found during design review predict cost of implementation.
  - Number of faults(old shirt damage)-predict cost of implementation(predict the cost)

# Classes of Prediction Systems

- Class 3: Measures of internal product attributes predict process attributes.
  - Example: Measures of design structure to predict maintenance effort, or the number of faults found during testing.
  - Measure the design structure predict the number of faults found during testing

# Classes of Prediction Systems

- Class 4: Measures of process attributes predict later product attributes.
  - Example: 1. Measure of failure during one operational period are used to predict likely failure occurrence in the subsequent operational period.
  - 2. Failures during testing predict reliability in the field.

# Accuracy of Measurement /1

- In an ongoing project, it is possible to validate the estimate by comparing actual values with predicted values. If  $E$  is a estimate of a value and  $A$  is the actual value, the **error** (RE) in the estimate is:

$$RE = \frac{A-E}{A}$$

- Relative error will be negative (if the estimate is greater than the actual value) or positive (if the estimate is less than the actual value).
- Inaccuracies can result from either statistical variation or estimating bias.
- Estimating inaccuracies must be compensated for throughout the project.

# Accuracy of Measurement /2

- *Relative Error (RE)* can be calculated for a collection of estimates, e.g., it is useful to determine if predictions are accurate for a group of projects.

■ The **mean relative error (MRE)** is calculated as the sum of *RE* for  $n$  projects divided by  $n$ . It is possible for small relative errors to balance out large ones, so the magnitude of the error can also be considered. (Sum of  $RE(n)/n$ )

- The **mean magnitude of relative error (MMRE)** is calculated using the *absolute value* of *RE*.



# Accuracy of Measurement /3

- MMRE is used to define prediction quality.
- On a set of  $n$  projects, if  $k$  is the number of projects whose  $RE$  is less than or equal to  $q$ , then prediction quality is calculated as:

$$PRED(q) = k/n$$

- **Example:**

- $PRED(0.25) = 0.47$  means that 47% of the predicted values fall within 25% of their actual values.
  - An estimation technique is (usually) acceptable if  $PRED(0.25) = 0.75$  .

# Exercise

- Assume you want to predict the error proneness of software modules.

a) How would you assess prediction accuracy?

$$RE \text{ (relative error)} = (A - E) / A$$

*with A: actual, E: estimated*

$$MRE = \frac{1}{n} \sum_{i=1}^n RE_i$$

$$MMRE = \frac{1}{n} \sum_{i=1}^n RE_i$$

b) How would you assess prediction quality?

$$PRED(q) = k/n$$

*where k is the number of predictions with  $|RE| \leq q$*