Chapter-3

A GOAL-BASED FRAMEWORK FOR SOFTWARE MEASUREMENT

Software metrics is a term that embraces many activities , all of which involve some degree of software measurement:

- Cost and effort estimation
- Productivity measures and models
- Data collection
- Quality models and measurement
- Reliability models
- Performance Evaluation and models
- Structural and complexity metrics
- Capability-maturity assessment
- Management by metrics
- Evaluation of methods and tools

- The framework for software measurement is based on three principles
  - Classifying the entities to be examined
  - Determining relevant measurement goals
  - Identifying the level of maturity that the organization has reached

Classifying Software Measures

- The first obligation of any software measurement activity is identifying the entities and attributes that we want to measure
- Software entities can be classified as follows
  - **Processes:** Collection of software-related activities.
  - **Products:** Artifacts, deliverables or documents that result from a process activity.
  - **Resources:** Entities required by a process activity.
- Within each class of entity, we distinguish between internal and external attributes of a product, process, or resource:

Classifying Software Measures (cont.)

- Internal attribute: are those that can be measured by examining the product, process, or resource on its own, separate from its behavior. (e.g. 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. (Ease of Navigation, Number of failures)

Classifying Software Measures (cont.)

Internal

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

External

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

Entities	Attributes	
Products	**Internal**	**External**
Requirements Use case models and scenarios	Size, reuse, modularity, redundancy, functionality, syntactic correctness, etc.	Comprehensibility, maintainability, etc.
Designs, Design models	Size, reuse, modularity, coupling, cohesiveness, functionality, etc.	Quality, complexity, maintainability, etc.
Code	Size, reuse, modularity, coupling, functionality, algorithmic complexity, control-flow structuredness, etc.	Reliability, usability, maintainability, etc.
Test requirements	Size, etc.	Effectiveness, etc.
Test data	Size, coverage,	Fault-finding ability
Test harness	Languages supported, features	Ease of use
...	...	...
Processes		
Constructing requirements	Time, effort, number of requirements changes, etc.	Quality, cost, stability, etc.
Detailed design	Time, effort, number of specification faults found, etc.	Cost, cost-effectiveness, etc.
Testing	Time, effort, number of coding faults found, etc.	Cost, cost-effectiveness, stability, etc.
...	...	...
Resources		
Personnel	Age, price, etc.	Productivity, experience, intelligence, etc.
Teams	Size, communication level, structuredness, etc.	Productivity, quality, etc.
Software	Price, size, etc.	Usability, reliability, etc.
Hardware	Price, speed, memory size, etc.	Reliability, etc.
Offices	Size, temperature, light, etc.	Comfort, quality, etc.
...	...	...

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.

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 have chosen.
- Following are some of the internal attributes that can be measured directly for a process
  - The duration of the 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.

- E.g: AT&T developers wanted to know the effectiveness of their software inspections. In particular, managers needed to evacuate 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.

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.
- External product attributes depend on both product behavior and environment, each attribute measure should take these characteristics into account.

- 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 measured them. For example the complexity of codes.

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).

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.
- The improvement in the process or products can be performed only when the project has clearly defined goals for processes and products.
- A clear understanding of goals can be used to generate suggested metrics for a given project in the context of a process maturity framework.

Goal-Question-Metric

- The GQM approach to process and metrics has proven to be a particular effective approach to selecting and implementing the metrics.
- To use GQM, You express the overall goals of your organization ⟶ Ask relevant questions ⟶ Measure.
- So, GQM approach provides a framework involving the following three steps
  - Listing the major goals of the development or maintenance project
  - Deriving the questions from each goal that must be answered to determine if the goals are being met
  - Decide what must be measured in order to be able to answer the questions adequately

- Typical goals are expressed in terms of productivity, quality, risk, customer satisfaction, etc. Goals and questions are to be constructed in terms of their audience.
- To help generate the goals, questions, and metrics, Basili & Rombach provided a series of templates.
  - **Purpose**: To (characterize, evaluate, predict, motivate, etc.) the (process, product, model, metric, etc.) in order to understand, assess, manage, engineer, learn, improve, etc. **Example**: To characterize the product in order to learn it, maintenance in order to improve it

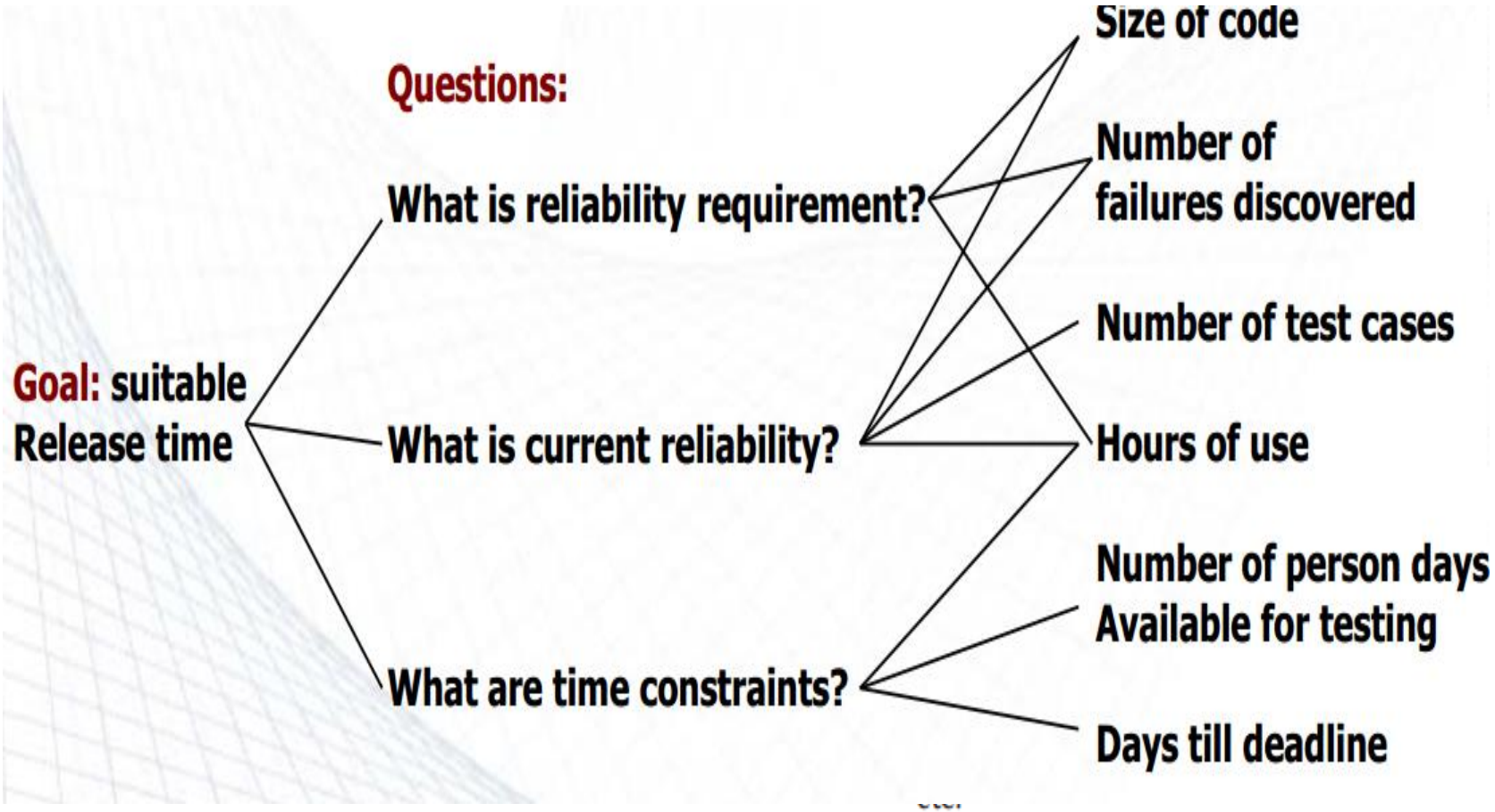
- **Perspective** − Examine the (cost, effectiveness, correctness, defects, changes, product measures, etc.) from the viewpoint of the developer, manager, customer, etc. **Example**: Examine the defects from the viewpoint of the customer.
- **Environment** − The environment consists of the following: process factors, people factors, problem factors, methods, tools, constraints, etc. **Example**: The customers of this software are those who have no knowledge about the tools.

Examples Of AT&T goals, questions and metrics

Table 3:3: Examples of AT&T goals, questions and metrics (Barnard and Price, 1994)

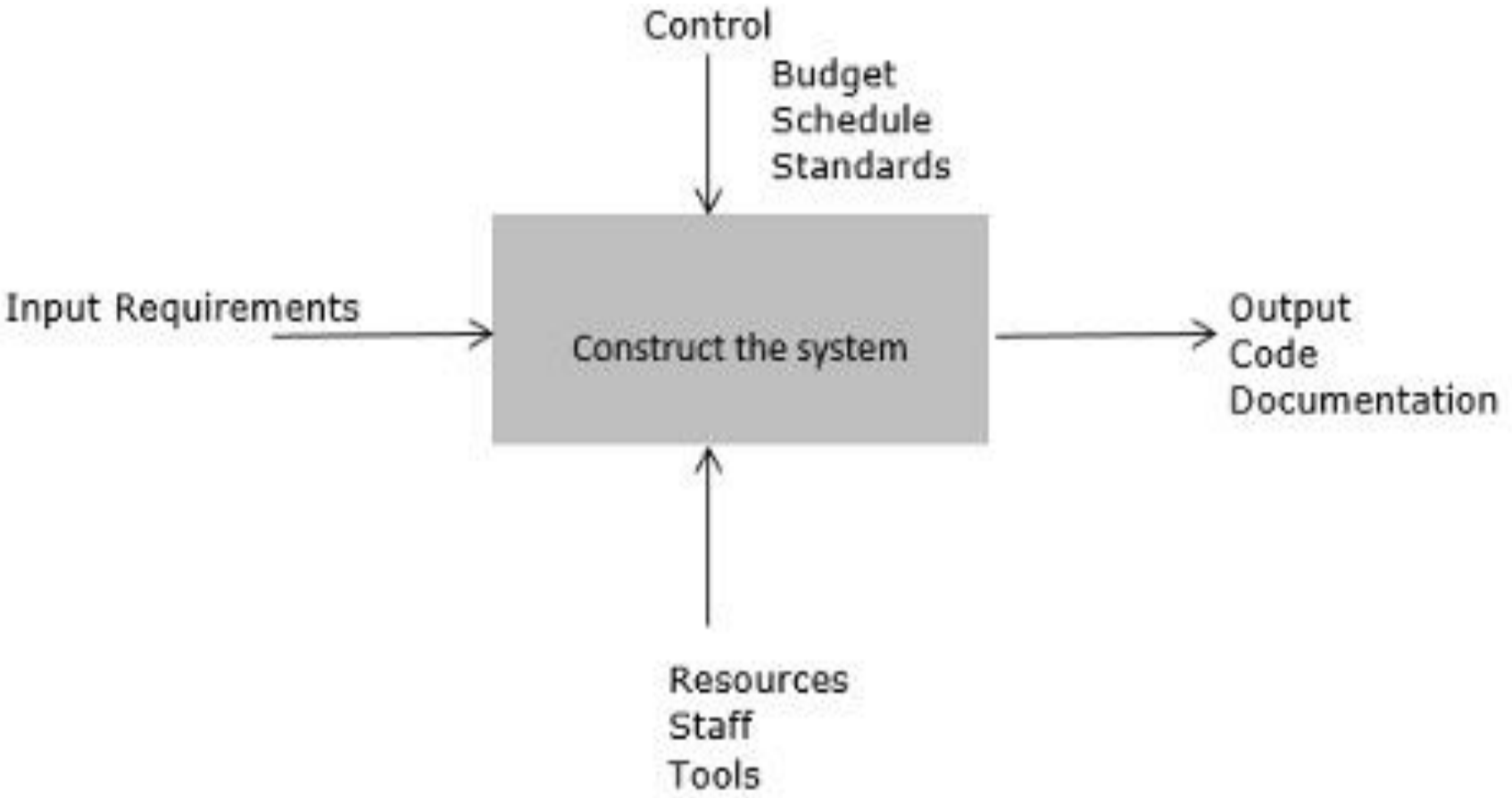
Goal	Questions	Metrics
Plan	How much does the inspection process cost?	Average effort per KLOC Percentage of reinspections
	How much calendar time does the inspection process take?	Average effort per KLOC Total KLOC inspected
Monitor and control	What is the quality of the inspected software?	Average faults detected per KLOC Average inspection rate Average preparation rate
	To what degree did the staff conform to the procedures?	Average inspection rate Average preparation rate Average lines of code inspected Percentage of reinspections
	What is the status of the inspection process?	Total KLOC inspected
Improve	How effective is the inspection process?	Defect removal efficiency Average faults detected per KLOC Average inspection rate Average preparation rate Average lines of code inspected
	What is the productivity of the inspection process?	Average effort per fault detected Average inspection rate Average preparation rate Average lines of code inspected

e.g. driving metrics from G&Q

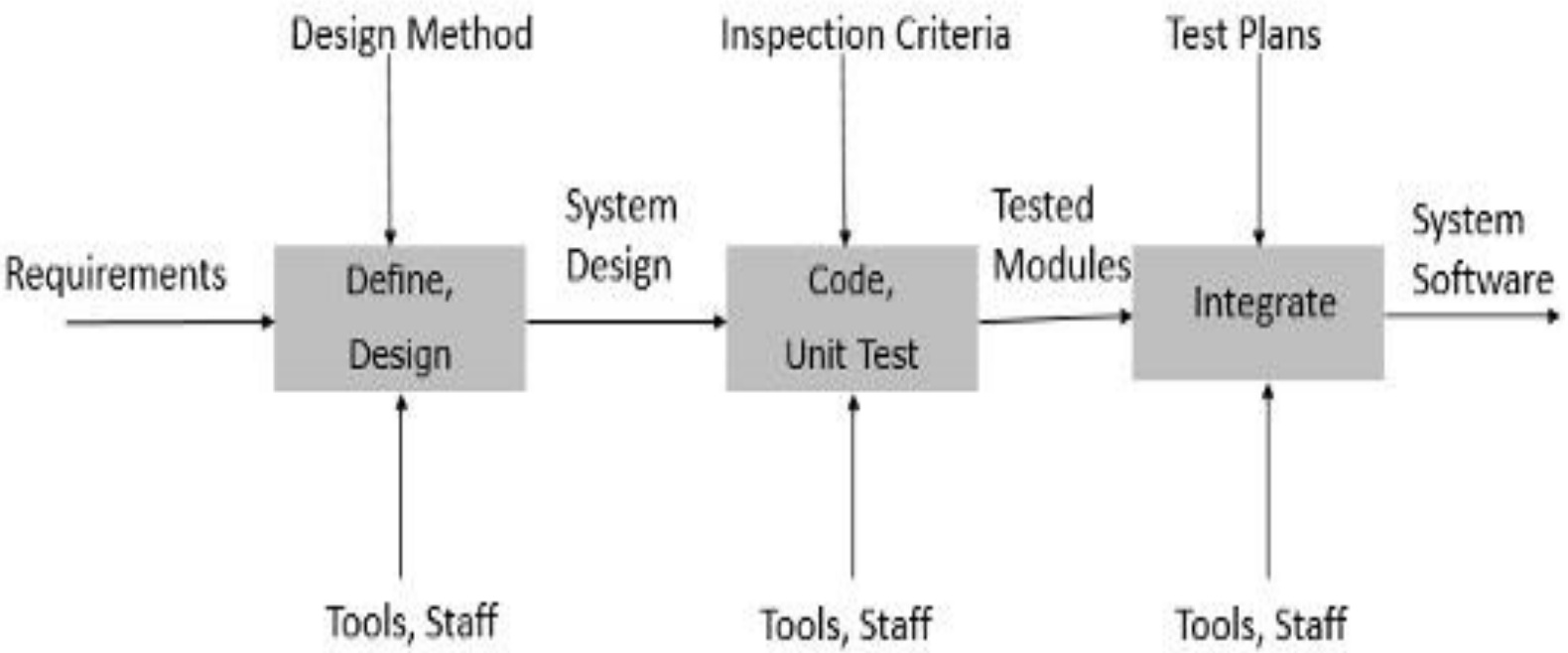


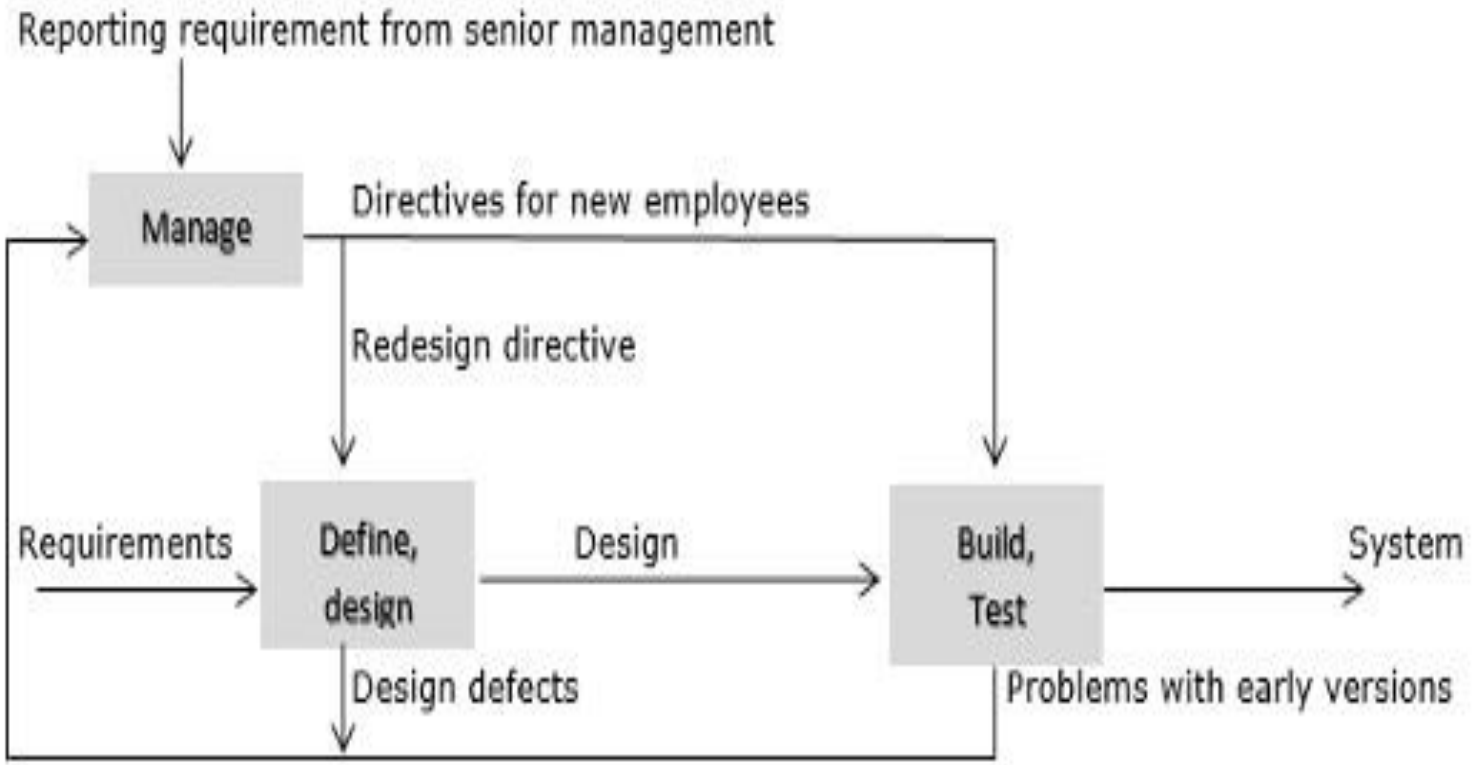
Measurement and Process Improvement

- Normally measurement is useful for:
  - Understanding the process and products
  - Establishing a baseline
  - Accessing and predicting the outcome
- According to the maturity level of the process given by SEI, the type of measurement and the measurement program will be different. It has suggested that there are five levels of process maturity. Namely: ad hoc, repeatable, defined, managed and optimized.
- The SEI distinguishes one level from another in terms of key process activity going on at each level.


Control
Budget
Schedule
Standards
Input Requirements
Construct the system
Output
Code
Documentation
Resources
Staff
Tools

-]


Design Method
Inspection Criteria
Test Plans
Requirements
Define,
Design
System
Design
Code,
Unit Test
Tested
Modules
Integrate
System
Software
Tools, Staff
Tools, Staff
Tools, Staff

- 
Reporting requirement from senior management
Manage
Directives for new employees
Redesign directive
Requirements
Define, design
Design
Build, Test
System
Design defects
Problems with early versions

- **Level 5: Optimizing**
- At this level, the measures from activities are used to improve the process by removing and adding process activities and changing the process structure dynamically in response to measurement feedback. Thus, the process change can affect the organization and the project as well as the process. The process will act as sensors and monitors, and we can change the process significantly in response to warning signs.

To sum-up

- At a given maturity level, we can collect the measurements for that level and all levels below it.

I. Ad hoc: Initial, Baseline

II. Repeatable: Process dependent on individual.

III. Defined: Process defined and institutionalized.

IV. Managed: Measured process.

V. Optimizing: Improvement feedback to the process.

- **Identifying the Level of Maturity:**
- Process maturity suggests to measure only what is visible. Thus, the combination of process maturity with GQM will provide most useful measures.
  - At **level 1**, the project is likely to have ill-defined requirements. At this level, the measurement of requirement characteristics is difficult.
  - At **level 2**, the requirements are well-defined and the additional information such as the type of each requirement and the number of changes to each type can be collected.
  - At **level 3**, intermediate activities are defined with entry and exit criteria for each activity

- The goal and question analysis will be the same, but the metric will vary with maturity.
- The more mature the process, the richer will be the measurements.
- The GQM paradigm, in concert with the process maturity, has been used as the basis for several tools that assist managers in designing measurement programs.
- GQM helps to understand the need for measuring the attribute, and process maturity suggests whether we are capable of measuring it in a meaningful way.
- Together they provide a context for measurement.

Software Measurement Validation

- Even when you know which entity and attribute you want to assess, there are many measures from which to choose.
- Sometimes, managers are confused by measurement which is not surprising.
- One of the roots of this confusion is the lack of software measurement validation.

- The validation approach depends on distinguishing measurement from prediction
  - **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.

Software Measurement Validation (con't)

- **Validating software Measures:** is the process of ensuring that the measure is a proper numerical characterization of the claimed attribute by showing that the representation condition is satisfied.
- The formal requirement for a validating measure involves demonstrating that it characterizes the stated attribute in the sense of measurement theory.

- **Validating a prediction system** in a given environment is the process of establishing the accuracy of the prediction system by empirical means; that is, by comparing model performance with known data in the given environment.
- It involves experimentation and hypothesis testing.
- To validate the prediction system formally you must first decide how stochastic it is, and then compare performance of the prediction system with known data points.

Performing Software Measurement Validation

- Software engineering community has always been aware of the need for validation.
- Thus, a measure must be viewed in the context in which it will be used.
- Validation must take into account the measurement's purpose; measure *X* may be valid for some uses but not for others.