Software Process & Project Management

Chapter Three :

Software Measurement

Outline

Software Measurement

Software Measurement:

Software Measurement: A measurement is a manifestation of the size, quantity, amount, or dimension of a particular attribute of a product or process. Software measurement is a titrate impute of a characteristic of a software product or the software process.

The software measurement process is defined and governed by ISO Standard.

Software Measurement Principles

The software measurement process can be characterized by five activities-

1. **Formulation**: The derivation of software measures and metrics appropriate for the representation of the software that is being considered.
2. **Collection**: The mechanism used to accumulate data required to derive the formulated metrics.
3. **Analysis**: The computation of metrics and the application of mathematical tools.
4. **Interpretation**: The evaluation of metrics results in insight into the quality of the representation.
5. **Feedback**: Recommendation derived from the interpretation of product metrics transmitted to the software team.

Need for Software Measurement

Software is measured to:

- Create the quality of the current product or process.
- Anticipate future qualities of the product or process.
- Enhance the quality of a product or process.
- Regulate the state of the project concerning budget and schedule.
- Enable data-driven decision-making in project planning and control.
- Identify bottlenecks and areas for improvement to drive process improvement activities.
- Ensure that industry standards and regulations are followed.
- Give software products and processes a quantitative basis for evaluation.
- Enable the ongoing improvement of software development practices.

The framework for software measurement is based on three principles −

1. Classifying the entities to be examined
2. Determining relevant measurement goals
3. Identifying the level of maturity that the organization has reached

1. Classifying the Entities to be Examined

In software engineering, mainly three classes of entities exist. They are −

1. Processes
2. Products
3. Resources

All of these entities have internal as well as external entities.

Internal attributes are those that can be measured purely in terms of the process, product, or resources itself. For example Size, complexity, and dependency among modules.

External attributes are those that can be measured only with respect to its relation with the environment. For example: The total number of failures experienced by a user, the length of time it takes to search the database and retrieve information.

Cont’d

The different attributes that can be measured for each of the entities are as follows −

Processes: Processes are collections of software-related activities. 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
- The different external attributes of a process are cost, controllability, effectiveness, quality and stability.

Products: Products are commitments made by management to deliver items or documents during the software life cycle. They include internal attributes like size, effort, cost, and functionality. External attributes include usability, integrity, efficiency, testability, reusability, portability, and interoperability.

Resources: These are entities required by a process activity. It can be any input for software production. It includes personnel, materials, tools, and methods.
The different internal attributes for the resources are age, price, size, speed, memory size, temperature, etc. The different external attributes are productivity, experience, quality, usability, reliability, comfort etc.

2. Determining Relevant Measurement Goals

A particular measurement will be useful only if it helps to understand the 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.

The Goal–Question–Metric (GQM) paradigm

The GQM approach provides a framework involving the following three steps −

1. Listing the major goals of the development or maintenance project
2. Deriving the questions from each goal that must be answered to determine if the goals are being met
3. Decide what must be measured to be able to answer the questions adequately

To use the GQM paradigm, first, we express the overall goals of the organization. Then, we generate the questions such that the answers are known so that we can determine whether the goals are being met. Later, analyze each question in terms of what measurement we need to answer each question.

Typical goals are expressed in terms of productivity, quality, risk, customer satisfaction, etc.

3. Identifying the Level of Maturity

- Process maturity suggests measuring only what is visible. Thus, the combination of process maturity with GQM will provide the 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 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.

3.1 Software Process and Product Measurement

Software Process Measurement: Focuses on measuring the characteristics of the software development process, such as effort, time, and defect rates. Examples include:

- Development time
- Cost
- Number of defects found during testing
- Team velocity

Software Product Measurement: involves assessing characteristics like size, complexity, and performance. Examples include:.

- Lines of code
- Function points
- Defect density
- Performance metrics

By Scope/Scale:

Personal Software Process (PSP): Focuses on individual software engineers improving their work practices

Team Software Process

Extends PSP principles to a team setting, emphasizing teamwork, communication, and process improvements

Organizational Software Process: Defines standardized processes at the organizational level to ensure consistency and quality across projects.

Why Measure Process and Product?

Process Measurement Benefits:

- Identify bottlenecks and inefficiencies.
- Improve process predictability and control.
- Track progress and manage risks

Product Measurement Benefits:

- Assess product quality and reliability.
- Identify areas for improvement in design and code.
- Make informed decisions about product release.

3.2 Quality of Measurement Results

"Quality of Measurement Results" refers to the degree of accuracy, precision, and reliability of a measurement, essentially how well a measurement reflects the true value of the quantity being measured, taking into account factors like consistency, error, and uncertainty associated with the measurement process.

Key aspects of quality measurement results:

- **Accuracy**: How close a measurement is to the true value of the quantity being measured.
- **Precision**: How consistent or repeatable a measurement is, even if it's not necessarily accurate.
- **Reliability**: The consistency of measurement results over repeated trials under similar conditions.
- **Validity**: Whether the measurement is actually capturing the intended concept or phenomenon.

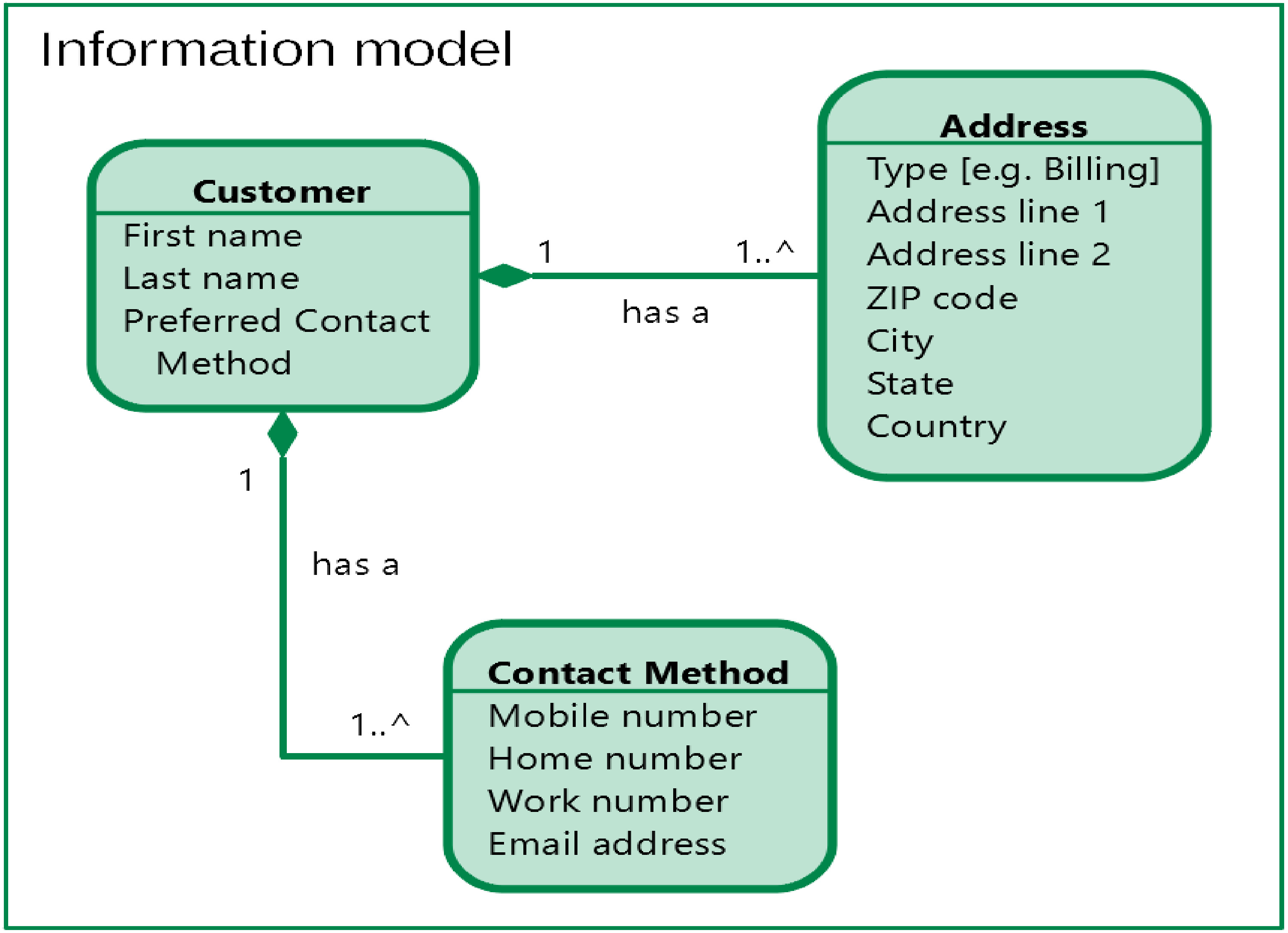
Factors affecting the quality of measurement results:

- **Instrument quality:** The quality and calibration of the measuring device used.
- **Measurement method:** The chosen technique or procedure for taking the measurement.
- **Environmental conditions:** External factors like temperature, humidity, and pressure that can impact measurements.
- **Operator skill:** The ability of the person performing the measurement.
- **Data analysis method:** How the raw data is processed and interpreted.

Assessing the quality of measurement results:

- **Calculating error:** Determining the difference between the measured value and the true value.
- **Uncertainty analysis:** Estimating the range within which the true value is likely to fall.
- **Statistical analysis:** Using statistical methods to assess variability and identify potential outliers.

3.3 Software Information Model



Software Information Model.. Cont’d

A software information model is a representation of concepts and relationships in a software system. It helps define the semantics of data for a specific domain.

Purpose :

- Helps manage the exchange of information between systems and devices
- Provides standardized syntax, semantics, and hierarchical structures for data
- Ensures interoperability and facilitates communication

How it's used

- Guides developers, designers, and stakeholders through the structure, behavior, and functionality of a system
- Helps specify data semantics for a chosen domain of discourse

Cont’d

How to create an information model

1. Determine the scope and level of detail of the model
2. Collect information needs
3. Identify object types
4. Design the information model
5. Validate the model
6. Publish and manage the model

Information model representation

- Information models are often represented in Unified Modeling Language (class) diagrams
- Informal information models can be written in plain English language

Related concepts Software modeling, Information system model, and Systems model.

3.4 Software Process Measurement Techniques

Methods used to quantify and analyze various aspects of the software development process. Provide data that helps teams understand, control, and improve their workflows.

- **Direct Measurement:** Directly measuring attributes, such as effort spent on a task or the number of defects found.
- **Indirect Measurement:** Deriving measurements from other measured values, such as calculating defect density from the number of defects and the size of the software.
- **Estimation:** Using expert judgment or historical data to estimate measurements, particularly useful in early stages of the project.
- **Automated Data Collection:** Using tools to automatically collect measurement data, such as version control systems or bug tracking systems.

Measurement Categories in Software Processes

Software process measurements can be classified into three broad categories:

A. Process Metrics
Evaluate the effectiveness and efficiency of software development processes.
Examples: Cycle time, Defect removal efficiency (DRE) , Review efficiency

B. Product Metrics
Assess the characteristics of the software product.
Examples: Code complexity (Cyclomatic Complexity), Code size (Lines of Code - LOC) , Maintainability Index

C. Project Metrics
Help in project planning and tracking.
Examples: Cost estimation, Schedule adherence, Effort estimation

Software Process Measurement Frameworks

Several frameworks guide organizations in applying software measurement techniques systematically:

A. Capability Maturity Model Integration (CMMI)

Provides maturity levels for process improvement.

Defines specific process areas for measurement.

B. ISO/IEC 15939 - Software Measurement Process

International standard for software measurement processes.

Defines measurement planning, data collection, and evaluation steps.

C. Six Sigma

Focuses on reducing defects and process variations using DMAIC (Define, Measure, Analyze, Improve, Control) methodology.

Challenges in Software Process Measurement

- Difficulty in selecting relevant metrics.
- Inconsistent or inaccurate data collection.
- Resistance from teams due to additional overhead.
- Complexity in interpreting results effectively.

Software process measurement is essential for continuous improvement and ensuring software quality. By selecting appropriate techniques and frameworks, organizations can enhance their development processes, reduce risks, and achieve better project outcomes.