Software Engineering

Dr. Mritunjay Shall Peelam
Assistant Professor-SG, UPES

Course Introduction and Objectives

Course Code	Course name	L	T	P	C
CSEG2064	Software Engineering	3	0	0	3
Total Units to be Covered: 5		Total Contact Hours: 45			
Prerequisite(s):		Syllabus version: 1.0			

Course Objectives

1. To explore software development methodologies (waterfall, agile, DevOps) and their integration of testing, quality assurance, reliability, and risk management.
2. To comprehend software requirements engineering and develop skills in creating well-structured Software Requirements Specifications (SRS).
3. To acquire understanding of planning a software project, its cost estimation models and to understand the software quality models.

Course Introduction and Objectives

Course Outcomes

CO 1. Understand the fundamental concepts and importance of Software Engineering in modern software development.

CO 2. Learn various software development methodologies, including Agile, Waterfall, and iterative approaches.

CO 3. Explore software design principles and architectural patterns for creating robust and maintainable software systems.

CO 4. Apply project management principles to effectively plan, monitor, and control software projects.

Course Introduction and Objectives

Syllabus

Unit I: Introduction to Software Engineering **7 Lecture Hours**

Definition of Software Engineering, S/W characteristics, applications, Software development life cycle ; Life Cycle Models – Waterfall (classical and iterative), Spiral, Prototyping & RAD Models, Software processes, Process Models – overview Agile Model and Various Agile methodologies - Scrum, XP, Lean, and Kanban. Scope of each model and their comparison in real-world case studies.

Course Introduction and Objectives

Unit II: Requirements Modelling and Design **9 Lecture Hours**

System and software requirements; Requirements Engineering-Crucial steps; types of requirements, Functional and non-functional requirements; Domain requirements; User requirements; Elicitation and analysis of requirements; Requirements documentation – Nature of Software, Software requirements specification, Use case diagrams with guidelines, DFD, SRS Structure, SRS Case study, Design concepts and principles - Abstraction - Refinement - Modularity Cohesion coupling, Architectural design, Detailed Design Transaction Transformation, Refactoring of designs, Object-oriented Design User-Interface Design.

Course Introduction and Objectives

Unit III: Software Reliability **9 Lecture Hours**

Introduction to Software Reliability; Hardware reliability vs. Software reliability; Reliability metrics; Failure and Faults – Prevention, Removal, Tolerance, Forecast; Dependability Concept – Failure Behavior, Characteristics, Maintenance Policy; Reliability and Availability Modeling; Reliability Evaluation Testing methods, Limits, Starvation, Coverage, Filtering; Microscopic Model of Software Risk; Classes of software reliability Models; Statistical reliability models; Reliability growth models; Defining and interpreting reliability metrics; Fault Detection and Prevention; Techniques for detecting and mitigating software faults; Static analysis tools and techniques; Dynamic analysis methods; Software Fault Tolerance; Software Maintenance and Reliability; Reliability Assessment and Evaluation; Methods for assessing and quantifying software reliability; Case Studies and Real-world Applications.

Course Introduction and Objectives

Unit IV: Software Testing, metrics and Quality Assurance **10 Lecture Hours**

Testing types and techniques such as black box, white box, and gray box testing, functional and structural testing; Test-driven development, code coverage, and quality metrics; Testing process, design of Test cases, testing techniques - boundary value analysis - equivalence class testing - decision table testing, cause-effect graphing, path testing, data flow testing, and mutation testing. Unit, integration, system, alpha, and beta testing, debugging techniques; verification and validation techniques, levels of testing, regression testing, quality management activities, product and process quality standards (ISO9000, CMM), metrics understanding (process, product, project metrics), size metrics (LOC, Function Count, Albrecht FPA), product metrics, metrics for software maintenance, cost estimation techniques (static, single variable, multivariable models), cost-benefit evaluation techniques, Testing tools and standards such as Jira and Selenium, test automation frameworks and tools (Selenium, Appium, JUnit), performance testing and load testing, and defect management and root cause analysis.

Course Introduction and Objectives

Unit V: Software Quality and Risk Management **10 Lecture Hours**

McCall quality factors, ISO and CMM Model, Tools and Techniques for Quality Control, Pareto Analysis, Statistical Sampling, Quality Control Charts and the seven Run Rule. Modern Quality Management, Risk Management – importance, types, process and phases, qualitative and quantitative risk analysis, Risk Analysis and Assessment, Risk Strategies, Risk Monitoring and Control, Risk Response and Evaluation. Software Reliability: Reliability Metrics, Reliability Growth Modeling. Use Case: Defect Tracking and Management. Test Automation Tools: Jira, Selenium:, Appium; JUnit.

Text Books and References

Textbooks	1. Roger S. Pressman, "Software Engineering: A practitioner's approach", 7th Edition, McGraw Hill, 2009. 2. Pankaj Jalote, "An integrated approach to Software Engineering", 3rd Edition, Springer/Narosa, 2005.
Reference books	1. James F. Peters, and Witold Pedrycz, "Software Engineering: an Engineering approach", John Wiley, 2007. 2. Waman S Jawadekar, "Software Engineering principles and practice", McGraw Hill, 2004.
Web Resources **Journals**	
MOOCs, online courses	

Modes of Evaluation

Modes of Evaluation: Quiz/Assignment/ presentation/ extempore/ Written Examination etc.

Examination Scheme

Components	IA	MID SEM	End Sem	Total
Weightage (%)	50	20	30	100

Unit-5

Unit V: Software Quality and Risk Management **10 Lecture Hours**

McCall quality factors, ISO and CMM Model, Tools and Techniques for Quality Control, Pareto Analysis, Statistical Sampling, Quality Control Charts and the seven Run Rule. Modern Quality Management, Risk Management – importance, types, process and phases, qualitative and quantitative risk analysis, Risk Analysis and Assessment, Risk Strategies, Risk Monitoring and Control, Risk Response and Evaluation. Software Reliability: Reliability Metrics, Reliability Growth Modeling. Use Case: Defect Tracking and Management. Test Automation Tools: Jira, Selenium:, Appium; JUnit.

Software Risk Management

- We Software developers are extremely optimists.
- We assume, everything will go exactly as planned.
- Other view
  - not possible to predict what is going to happen ?

Software surprises

- Never good news

Risk management is required to reduce this surprise factor

Dealing with concern before it becomes a crisis.

Quantify probability of failure & consequences of failure.

What is risk ?

Tomorrow's problems are today's risks.

"Risk is a problem that may cause some loss or threaten the success of the project, but which has not happened yet".

Risk management is the process of identifying addressing and eliminating these problems before they can damage the project.

Current problems &

Potential Problems

Typical Software Risk

Capers Jones has identified the top five risk factors that threaten projects in different applications.

1. Dependencies on outside agencies or factors.
    - Availability of trained, experienced persons
    - Inter group dependencies
    - Customer-Furnished items or information
    - Internal & external subcontractor relationships

2. Requirement issues

Uncertain requirements

Wrong product

or

Right product badly

Either situation results in unpleasant surprises and unhappy customers.

- Lack of clear product vision
- Lack of agreement on product requirements
- Unprioritized requirements
- New market with uncertain needs
- Rapidly changing requirements
- Inadequate Impact analysis of requirements changes

3. Management Issues

Project managers usually write the risk management plans, and most people do not wish to air their weaknesses in public.

- Inadequate planning
- Inadequate visibility into actual project status
- Unclear project ownership and decision making
- Staff personality conflicts
- Unrealistic expectation
- Poor communication

4. Lack of knowledge

- Inadequate training
- Poor understanding of methods, tools, and techniques
- Inadequate application domain experience
- New Technologies
- Ineffective, poorly documented or neglected processes

5. Other risk categories

- Unavailability of adequate testing facilities
- Turnover of essential personnel
- Unachievable performance requirements
- Technical approaches that may not work

Risk Management Activities

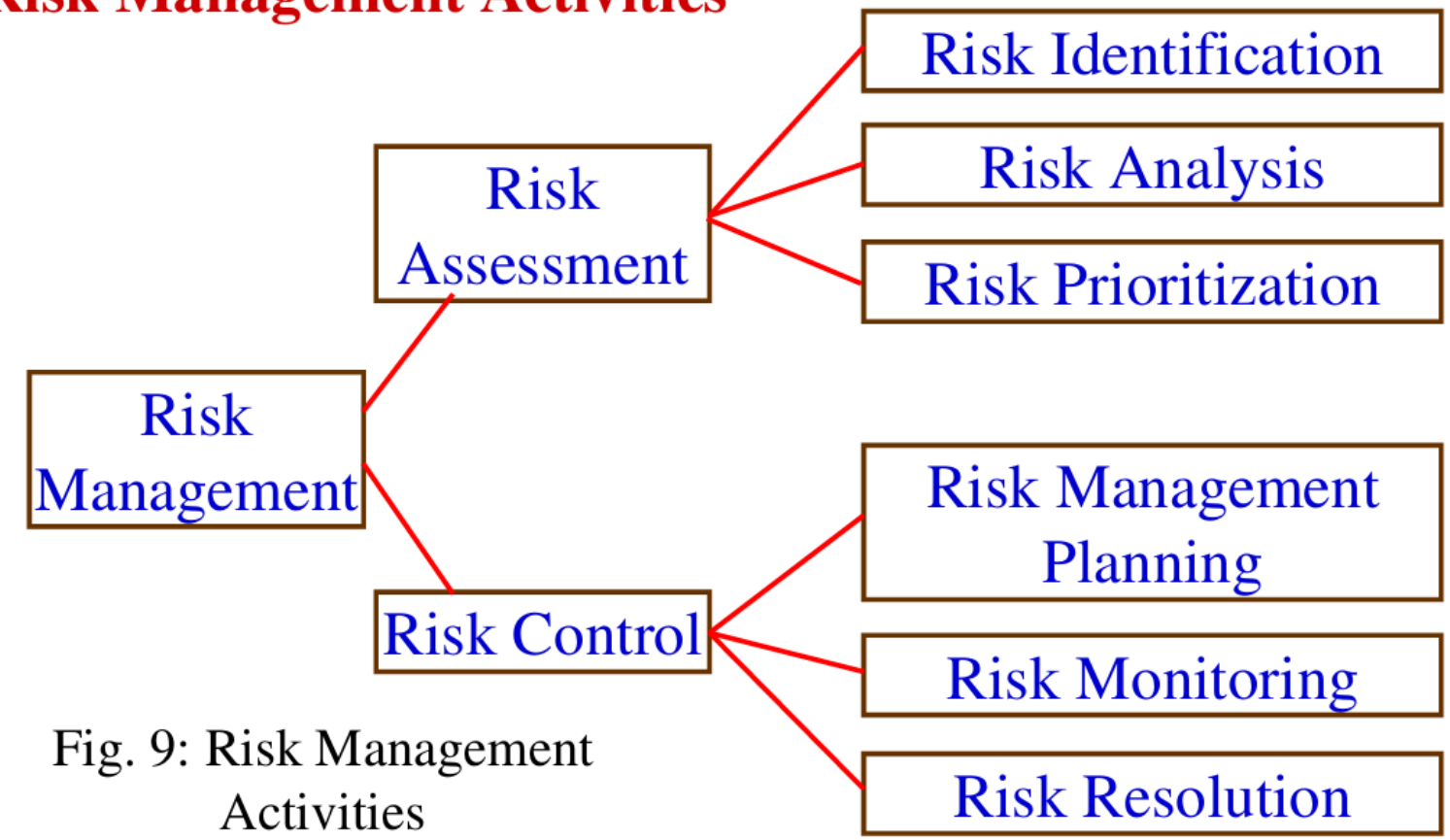


Fig. 9: Risk Management Activities

Risk Assessment

Identification of risks

Risk analysis involves examining how project outcomes might change with modification of risk input variables.

Risk prioritization focus for severe risks.

Risk exposure: It is the product of the probability of incurring a loss due to the risk and the potential magnitude of that loss.

Another way of handling risk is the risk avoidance. Do not do the risky things! We may avoid risks by not undertaking certain projects, or by relying on proven rather than cutting edge technologies.

Risk Control

Risk Management Planning produces a plan for dealing with each significant risks.

- Record decision in the plan.

Risk resolution is the execution of the plans of dealing with each risk.

McCall Software Quality Model

- McCall et. al. [MCCA77] model of software quality was introduced in 1977 and many quality factors were incorporated.
- The model distinguishes between two levels of quality attributes. Higher level quality attributes are known as quality factors.
- These are external attributes and can be measured directly. The second level of quality attributes is named as quality criteria.
- Quality criteria can be measured either subjectively or objectively. The software quality factors are organised in three product quality factors as shown in Fig. 7.9.

McCall Software Quality Model

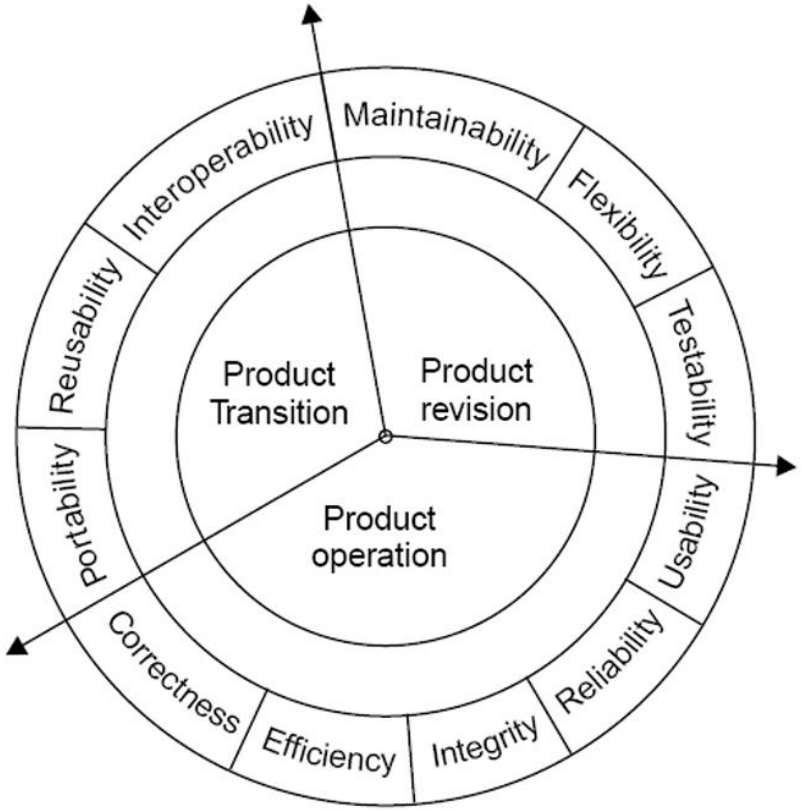


Fig 7.9: Software quality factors

McCall Software Quality Model

i. Product Operation

Factors which are related to the operation of a product are combined. The factors are:

- Correctness
- Efficiency
- Integrity
- Reliability
- Usability

These five factors are related to operational performance, convenience, ease of usage and its correctness. These factors play a very significant role in building customer's satisfaction.

McCall Software Quality Model

ii. Product Revision

The factors which are required for testing & maintenance are combined and are given below:

- Maintainability
- Flexibility
- Testability

These factors pertain to the testing & maintainability of software. They give us idea about ease of maintenance, flexibility and testing effort. Hence, they are combined under the umbrella of product revision.

McCall Software Quality Model

iii. Product Transition

We may have to transfer a product from one platform to an other platform or from one technology to another technology. The factors related to such a transfer are combined and given below:

- Portability
- Reusability
- Interoperability

McCall Software Quality Model

1	Reliability	The extent to which a software performs its intended functions without failure.
2	Correctness	The extent to which a software meets its specifications.
3	Consistency & precision	The extent to which a software is consistent and give results with precision.
4	Robustness	The extent to which a software tolerates the unexpected problems.
5	Simplicity	The extent to which a software is simple in its operations.
6	Traceability	The extent to which an error is traceable in order to fix it.
7	Usability	The extent of effort required to learn, operate and understand the functions of the software

McCall Software Quality Model

8	Accuracy	Meeting specifications with precision.
9	Clarity & Accuracy of documentation	The extent to which documents are clearly & accurately written.
10	Conformity of operational environment	The extent to which a software is in conformity of operational environment.
11	Completeness	The extent to which a software has specified functions.
12	Efficiency	The amount of computing resources and code required by software to perform a function.
13	Testability	The effort required to test a software to ensure that it performs its intended functions.
14	Maintainability	The effort required to locate and fix an error during maintenance phase.

McCall Software Quality Model

15	Modularity	It is the extent of ease to implement, test, debug and maintain the software.
16	Readability	The extent to which a software is readable in order to understand.
17	Adaptability	The extent to which a software is adaptable to new platforms & technologies.
18	Modifiability	The effort required to modify a software during maintenance phase.
19	Expandability	The extent to which a software is expandable without undesirable side effects.
20	Portability	The effort required to transfer a program from one platform to another platform.

McCall Software Quality Model

Most of the quality factors are explained in table 7.4. The remaining factors are given in table 7.5.

Sr.No.	Quality Factors	Purpose
1	Integrity	The extent to which access to software or data by the unauthorized persons can be controlled.
2	Flexibility	The effort required to modify an operational program.
3	Reusability	The extent to which a program can be reused in other applications.
4	Interoperability	The effort required to couple one system with another.

Table 7.5: Remaining quality factors (other are in table 7.4)

McCall Software Quality Model

Quality criteria

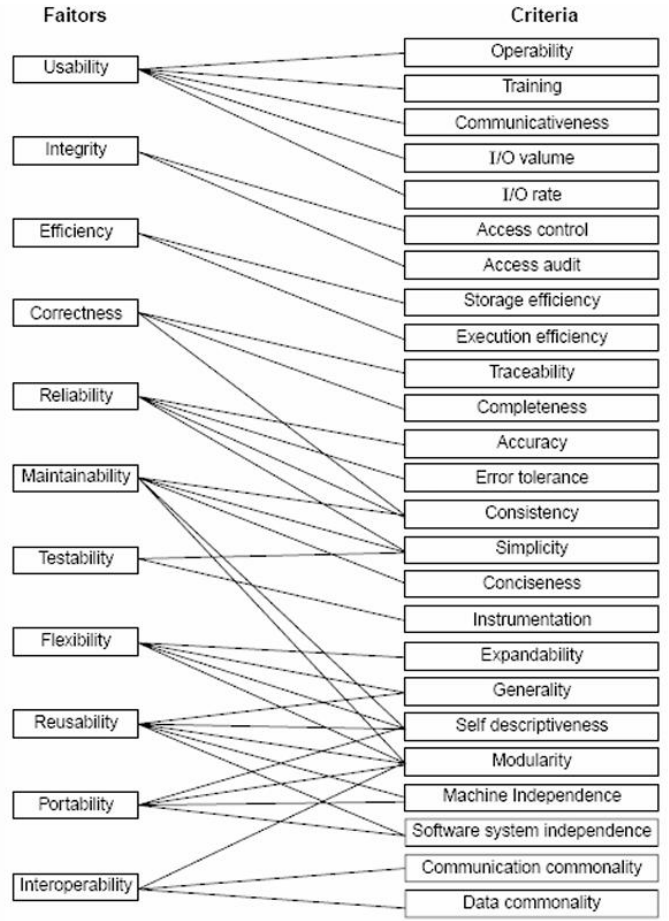
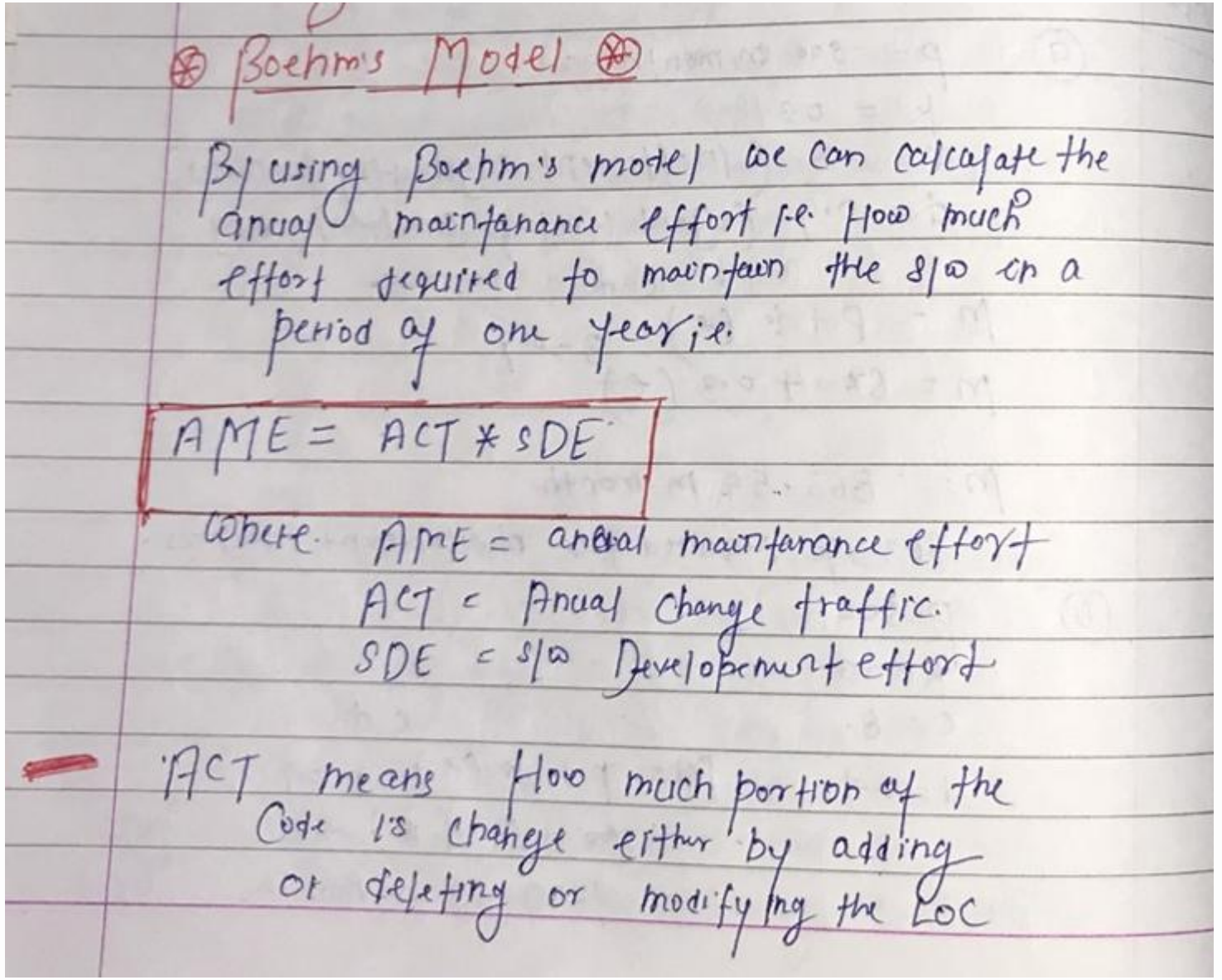
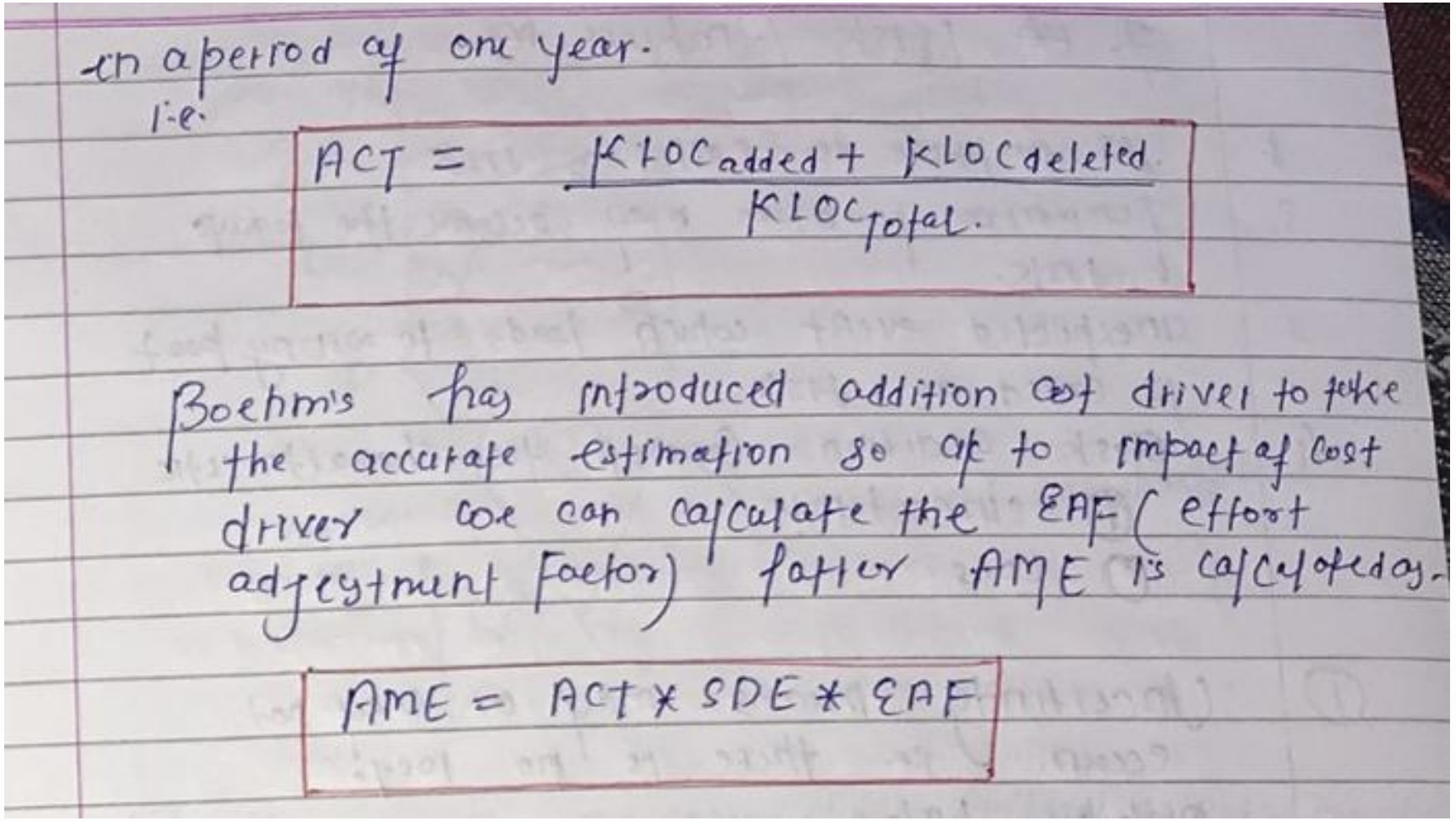


Fig 7.10: McCall's quality model


Boehm's Model
By using Boehm's model we can calcajate the anual maintanance effort i.e. How much effort required to maintain the s/w in a period of one year i.e.
AME = ACT * SDE
Where AME = anual maintanance effort
ACT = Anual change traffic.
SDE = s/w Development effort
ACT means How much portion of the Code is change either by adding or deleting or modifying the LOC


in a period of one year.
i.e.
ACT = KLOC added + KLOC deleted / KLOC total.
Boehm's has introduced addition cost driver to take the accurate estimation so as to impact of cost driver we can calculate the EAF (effort adjustment factor) after AME is calculated.
AME = ACT * SDE * EAF

Q. Anual C.T. of the product is 15% per year. The Development effort of the project is 600 m-month. The life time of the product is 10 year. What is Total effort required for the project?

Sol.

Given. ACT = 15% / year.
SDE = 600 m-mont.
Life time = 10 yr.
Total effort = ?

AME = ACT * SDE
= 15% X 600
= 90 m-month

10 yr life time = 10 X 90.
Maintance effort = 900 m-months.
Total project effort = Developent effoo + AME
= 900 + 600.
= 1500 m-month

- Capability Maturity Model

It is a strategy for improving the software process, irrespective of the actual life cycle model used.

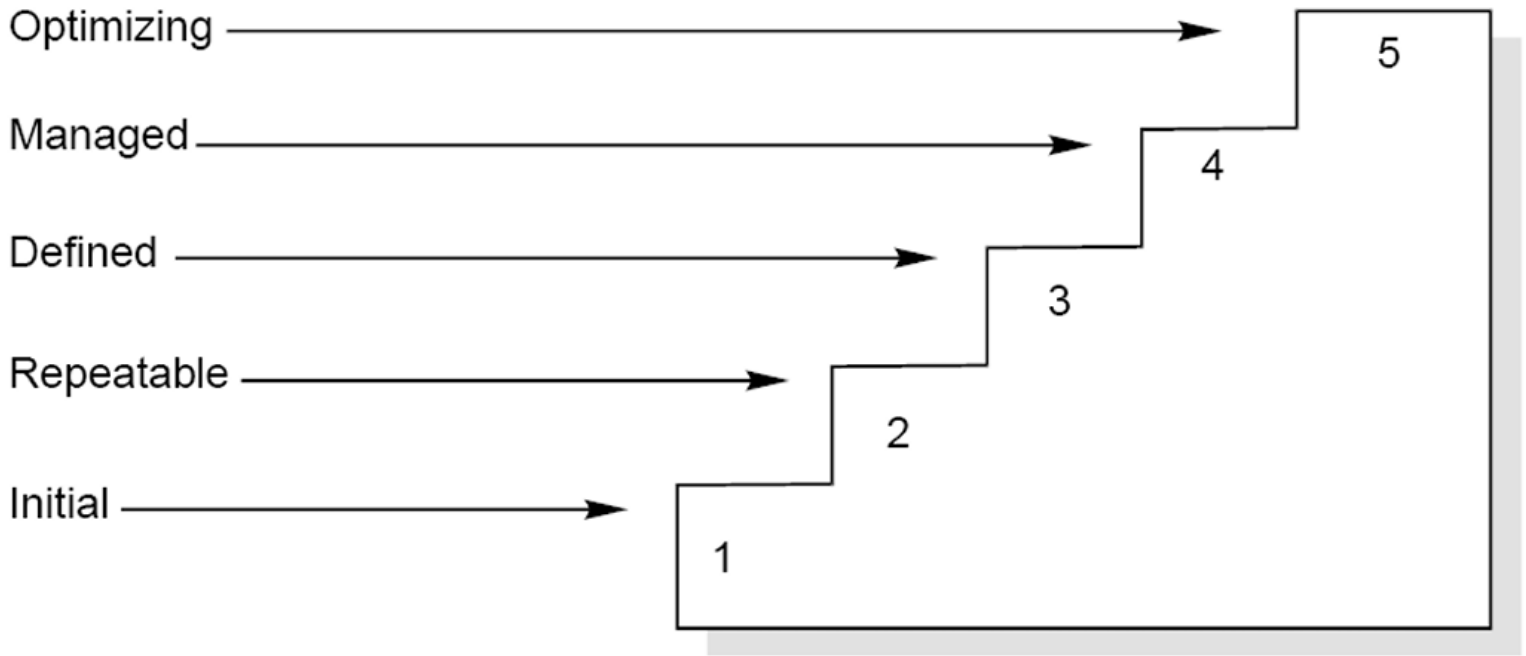


Fig.7.23: Maturity levels of CMM

Maturity Levels:

- ✓ Initial (Maturity Level 1)
- ✓ Repeatable (Maturity Level 2)
- ✓ Defined (Maturity Level 3)
- ✓ Managed (Maturity Level 4)
- ✓ Optimizing (Maturity Level 5)

Maturity Level	Characterization
Initial	Adhoc Process
Repeatable	Basic Project Management
Defined	Process Definition
Managed	Process Measurement
Optimizing	Process Control

Fig.7.24: The five levels of CMM

Key Process Areas

The key process areas at level 2 focus on the software project's concerns related to establishing basic project management controls, as summarized below:

Requirements Management (RM)	Establish a common relationship between the customer requirements and the developers in order to understand the requirements of the project.
Software Project Planning (PP)	Establish reasonable plans for performing the software engineering and for managing the software project.
Software Project Tracking and Oversight (PT)	Establish adequate visibility into actual progress so that management can take effective actions when the software project's performance deviates significantly from the software plans.
Software Subcontract Management (SM)	Select qualified software subcontractors and manage them effectively.
Software Quality Assurance (QA)	Provide management with appropriate visibility into the process being used by the software project and of the products being built.
Software Configuration Management (CM)	Establish and maintain the integrity of the products of the software project throughout the project's software life cycle.

The key process areas at level 3 address both project and organizational issues, as summarized below:

Organization Process Focus (PF)	Establish the organizational responsibility for software process activities that improve the organization's overall software process capability.
Organization Process Definition (PD)	Develop and maintain a usable set of software process assets that improve process performance across the projects and provide a basis for cumulative, long-term benefits to the organization.
Training Program (TP)	Develop the skills and knowledge of individuals so that they can perform their roles effectively and efficiently.
Integrated Software Management (IM)	Integrate the software engineering and management activities into a coherent, defined software process that is tailored from the organization's standard software process and related process assets.

(Contd.)...

Software Product Engineering (PE)	Consistently perform a well-defined engineering process that integrates all the software engineering activities to produce correct, consistent software products effectively and efficiently.
Inter group Coordination (IC)	Establish a means for the software engineering group to participate actively with the other engineering groups so the project is better able to satisfy the customer's needs effectively and efficiently.
Peer Reviews (PR)	Remove defects from the software work products early and efficiently. An important corollary effect is to develop a better understanding of the software work products and of the defects that can be prevented.

The key process areas at level 4 focus on establishing a quantitative understanding of both the software process and the software work products being built, as summarized below:

Quantitative Process Management (QP)	Control the process performance of the software project quantitatively.
Software Quality Management (QM)	Develop a quantitative understanding of the quality of the project's software products and achieve specific quality goals.

The key process areas at level 5 cover the issues that both the organization and the projects must address to implement continuous and measurable software process improvement, as summarized below:

Defect Prevention (DP)	Identify the causes of defects and prevent them from recurring.
Technology Change Management (TM)	Identify beneficial new technologies (i.e., tools, methods, and processes) and transfer them into the organization in an orderly manner.
Process Change Management (PC)	Continually improve the software processes used in the organization with the intent of improving software quality, increasing productivity, and decreasing the cycle time for product development.

Common Features

Commitment to Perform (CO)	Describes the actions the organizations must take to ensure that the process is established and will endure. It includes practices on policy and leadership.
Ability to Perform (AB)	Describes the preconditions that must exist in the project or organization to implement the software process competently. It includes practices on resources, organizational structure, training, and tools.
Activities Performed (AC)	Describes the role and procedures necessary to implement a key process area. It includes practices on plans, procedures, work performed, tracking, and corrective action.
Measurement and Analysis (ME)	Describes the need to measure the process and analyze the measurements. It includes examples of measurements.
Verifying Implementation (VE)	Describes the steps to ensure that the activities are performed in compliance with the process that has been established. It includes practices on management reviews and audits.

- ISO 9000

The SEI capability maturity model initiative is an attempt to improve software quality by improving the process by which software is developed.

ISO-9000 series of standards is a set of document dealing with quality systems that can be used for quality assurance purposes. ISO-9000 series is not just software standard. It is a series of five related standards that are applicable to a wide variety of industrial activities, including design/ development, production, installation, and servicing. Within the ISO 9000 Series, standard ISO 9001 for quality system is the standard that is most applicable to software development.

- Mapping ISO 9001 to the CMM

1. Management responsibility
2. Quality system
3. Contract review
4. Design control
5. Document control
6. Purchasing
7. Purchaser-supplied product

8. Product identification and traceability
9. Process control
10. Inspection and testing
11. Inspection, measuring and test equipment
12. Inspection and test status
13. Control of nonconforming product
14. Corrective action

15. Handling, storage, packaging and delivery
16. Quality records
17. Internal quality audits
18. Training
19. Servicing
20. Statistical techniques

- Contrasting ISO 9001 and the CMM

There is a strong correlation between ISO 9001 and the CMM, although some issues in ISO 9001 are not covered in the CMM, and some issues in the CMM are not addressed in ISO 9001.

The biggest difference, however, between these two documents is the emphasis of the CMM on continuous process improvement.

The biggest similarity is that for both the CMM and ISO 9001, the bottom line is **"Say what you do; do what you say"**.

The 7 Quality Control Tools

- Quality is very important for the long-term success and growth of any organization. Whether you are experienced or new, the 7 Quality Control tools help improve your skills.
- The famous quality expert Kaoru Ishikawa introduced these 7 tools during Japan's recovery after World War II.
- These tools made quality concepts simple and useful for everyone in an organization.
- These tool uses diagrams and charts to identify problems, understand them clearly, and find solutions.
- They provide a structured and data-based way to improve processes and make decisions.
- With these tools, teams can make better decisions based on understanding and facts, rather than guessing or relying only on intuition.

The 7 Quality Control Tools

1. Cause-and-Effect Diagram (Fishbone/Ishikawa Diagram)
2. Check Sheets (Tally Sheets)
3. Control Charts (Shewhart Charts)
4. Histograms
5. Pareto Charts
6. Scatter Diagrams
7. Stratification (Flowcharts/Run Charts)

Benefits of using the 7 quality control tools

Effective Problem-Solving: By providing a structured framework for identifying root causes, analyzing data, and visualizing relationships, the 7 QC tools equip teams with the necessary insights to address quality issues effectively.

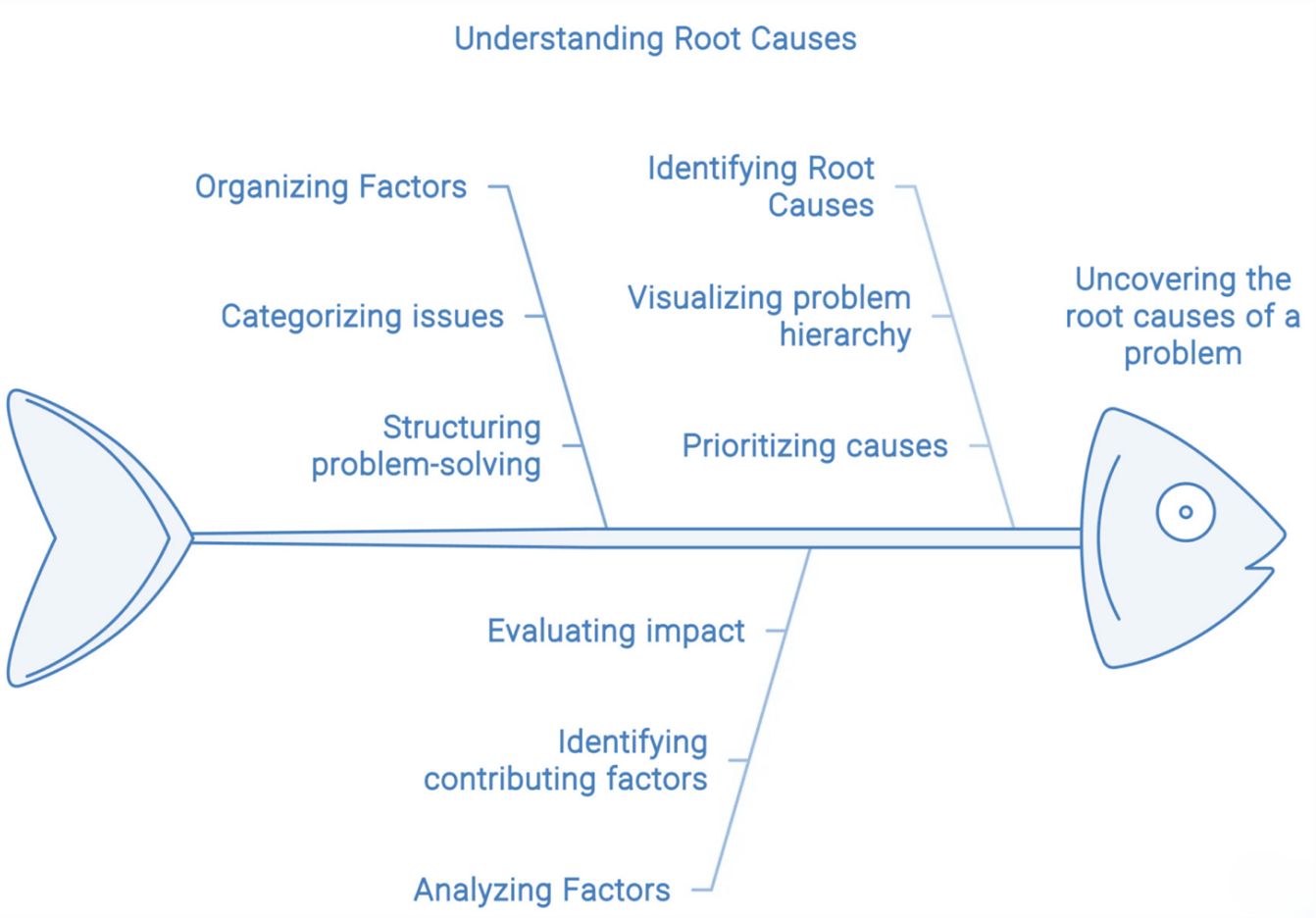
Process Improvement: Through data-driven analysis and monitoring, these tools enable organizations to identify areas for improvement, streamline processes, and eliminate inefficiencies, ultimately enhancing productivity and reducing waste.

Data-driven Decision Making: The 7 quality control tools empower teams to base their decisions on objective data and statistical analysis, minimizing the risk of biases or unfounded assumptions, and leading to more informed and effective decision-making processes.

Cause-and-Effect Diagram (Fishbone Diagram)

- The Cause-and-Effect Diagram, also known as the Fishbone Diagram or Ishikawa Diagram, is a powerful tool designed to facilitate root cause analysis and identify potential causes contributing to a specific problem or effect.
- Named after its creator, Kaoru Ishikawa, this diagram visually represents the relationship between an effect and its potential causes, resembling the skeletal structure of a fish.
- The primary purpose of the Cause-and-Effect Diagram is to systematically explore and organize the various factors that could potentially contribute to a particular issue or outcome.
- Teams trained in root cause analysis training often excel at using Fishbone Diagrams to systematically trace issues to their source."

Cause-and-Effect Diagram (Fishbone Diagram)



Cause-and-Effect Diagram (Fishbone Diagram)

Creating an effective Cause-and-Effect Diagram involves the following steps:

- **Define the problem or effect:** Clearly state the issue or outcome you wish to analyze, which will be represented as the “fish head” on the diagram.
- **Identify the main cause categories:** Determine the primary categories or broad areas that could potentially contribute to the problem, such as materials, methods, machinery, environment, or personnel. These categories will form the “bones” or main branches of the fishbone diagram.
- **Brainstorm potential causes:** For each main category, engage in a structured brainstorming session to identify specific potential causes or contributing factors. These sub-causes will be represented as smaller “bones” branching off from the main categories.
- **Analyze and prioritize causes:** Once all potential causes have been identified, analyze the diagram to determine which causes are most likely to be contributing to the problem. Prioritize these causes based on their perceived impact or likelihood of occurrence.
- **Develop and implement countermeasures:** Based on the prioritized causes, develop and implement targeted countermeasures or corrective actions to address the root causes and mitigate the problem effectively.

Check Sheets (Tally Sheets)

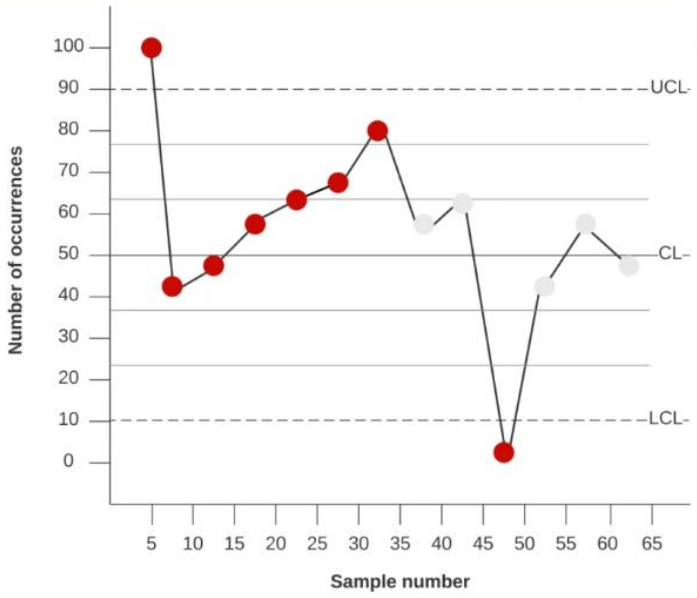
	Monday	Tuesday	Wednesday	Thursday	Friday	Total
Wrong item in the pick location	𝍸	𝍸	𝍸	𝍸	𝍸	17
Item was picked incorrectly	𝍸	𝍸	𝍸	𝍸	𝍸	19
Two customers received each other's items	𝍸	𝍸	𝍸	𝍸	𝍸	4
Product was damaged	𝍸	𝍸	𝍸	𝍸	𝍸	7
Total	13	8	8	12	6	47

Check Sheets (Tally Sheets)

- Check sheets, also known as tally sheets, are straightforward yet powerful tools designed to facilitate the systematic collection and organization of data related to quality issues, defects, or process performance.
- These sheets serve as a structured means of recording and tabulating data, enabling organizations to identify patterns, trends, and areas for improvement.
- The primary purpose of check sheets is to streamline the process of data collection and analysis, allowing teams to gather quantitative or qualitative information consistently and efficiently.

Control Chart (Shewhart Chart)

Control chart



Legend

- Out of control
- In control
- UCL: Upper control limit
- CL: Center line
- LCL: Lower control limit

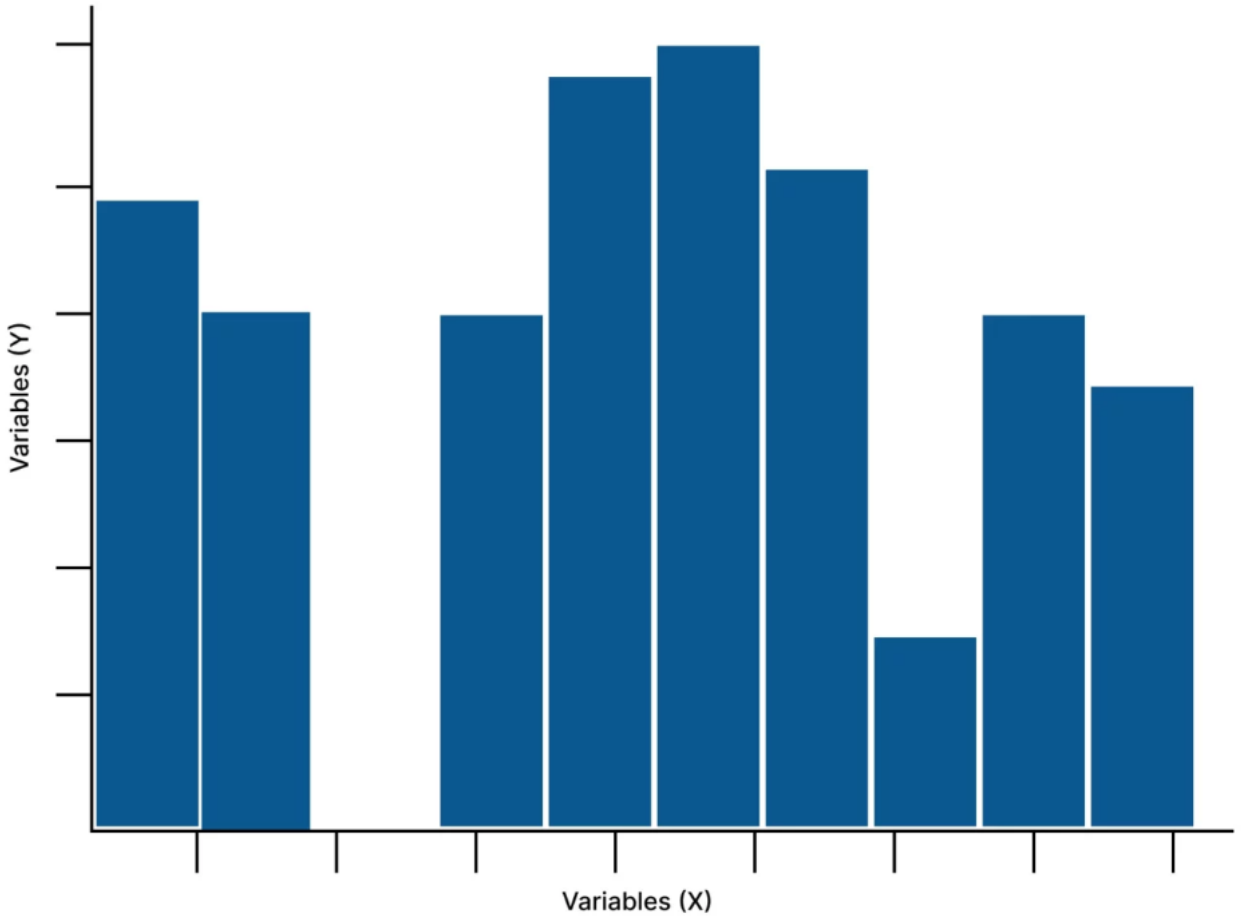
Action plan

Action	Owner	Start date	End date	Required resources	Status
Analyze chart	John	3/15	3/20	Control chart, process analyst	In progress
Investigate the cause for the out-of-control metric	Chris	3/15	3/22	Documentation of expected process, process manger	In progress
Determine if changes need to be made to the process	Peter	3/16	3/22	Process manager	Completed

Control Chart (Shewhart Chart)

- Control charts, also known as Shewhart charts, are powerful statistical tools used for monitoring and analyzing process performance over time.
- Named after Walter A. Shewhart, a pioneer in the field of statistical quality control, these charts are designed to help organizations determine whether a process is stable and predictable, or if it is subject to undesirable variations that require intervention.
- The primary purpose of control charts is to enable organizations to practice statistical process control (SPC), which involves monitoring and controlling a process to ensure that it operates within predetermined statistical limits.

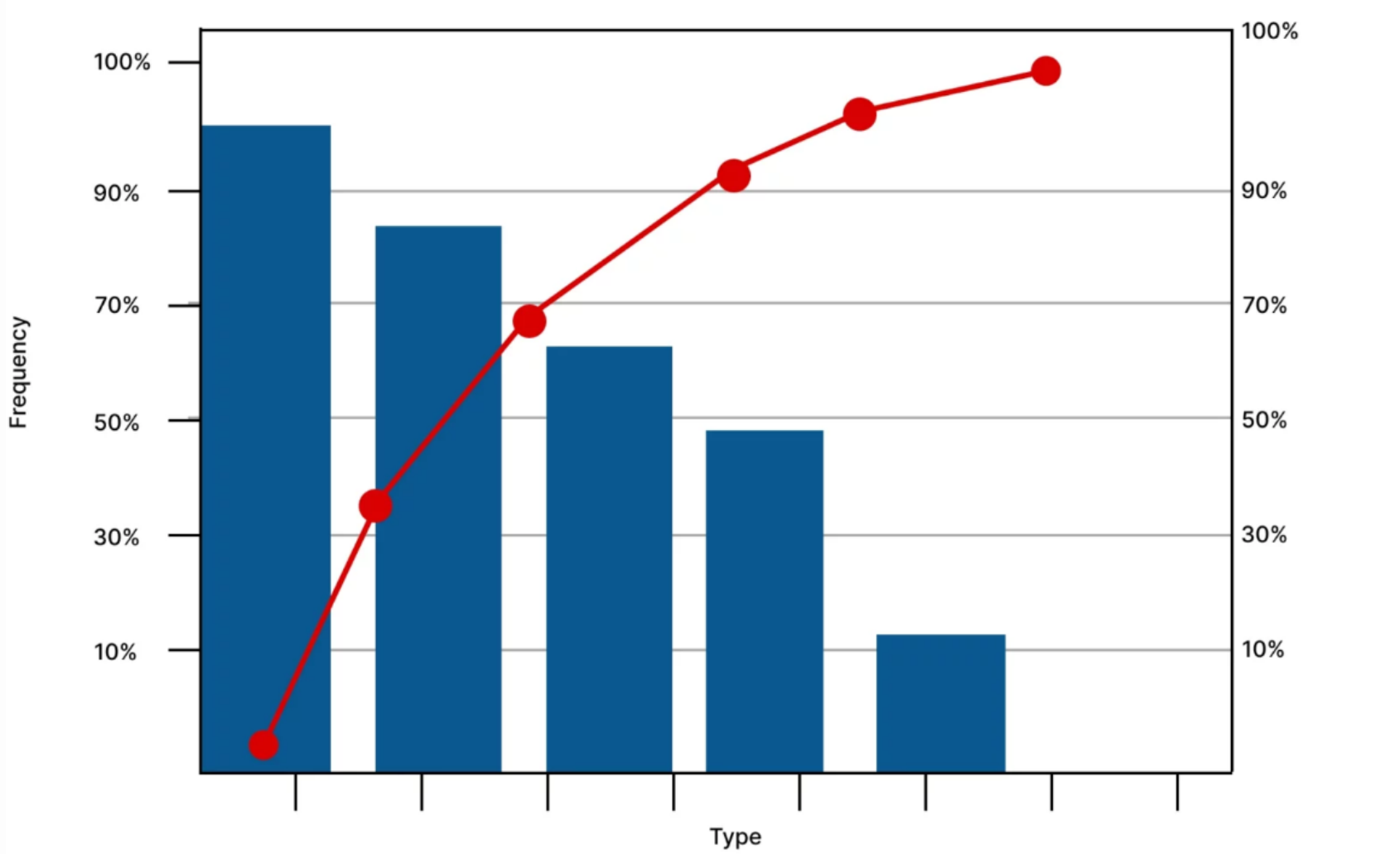
Histogram



Histogram

- A histogram is a powerful data visualization tool that graphically represents the frequency distribution of a set of data.
- It is a type of bar chart that displays the number of occurrences or observations within specific ranges or intervals, providing a clear visual representation of how data is distributed.

Pareto Chart

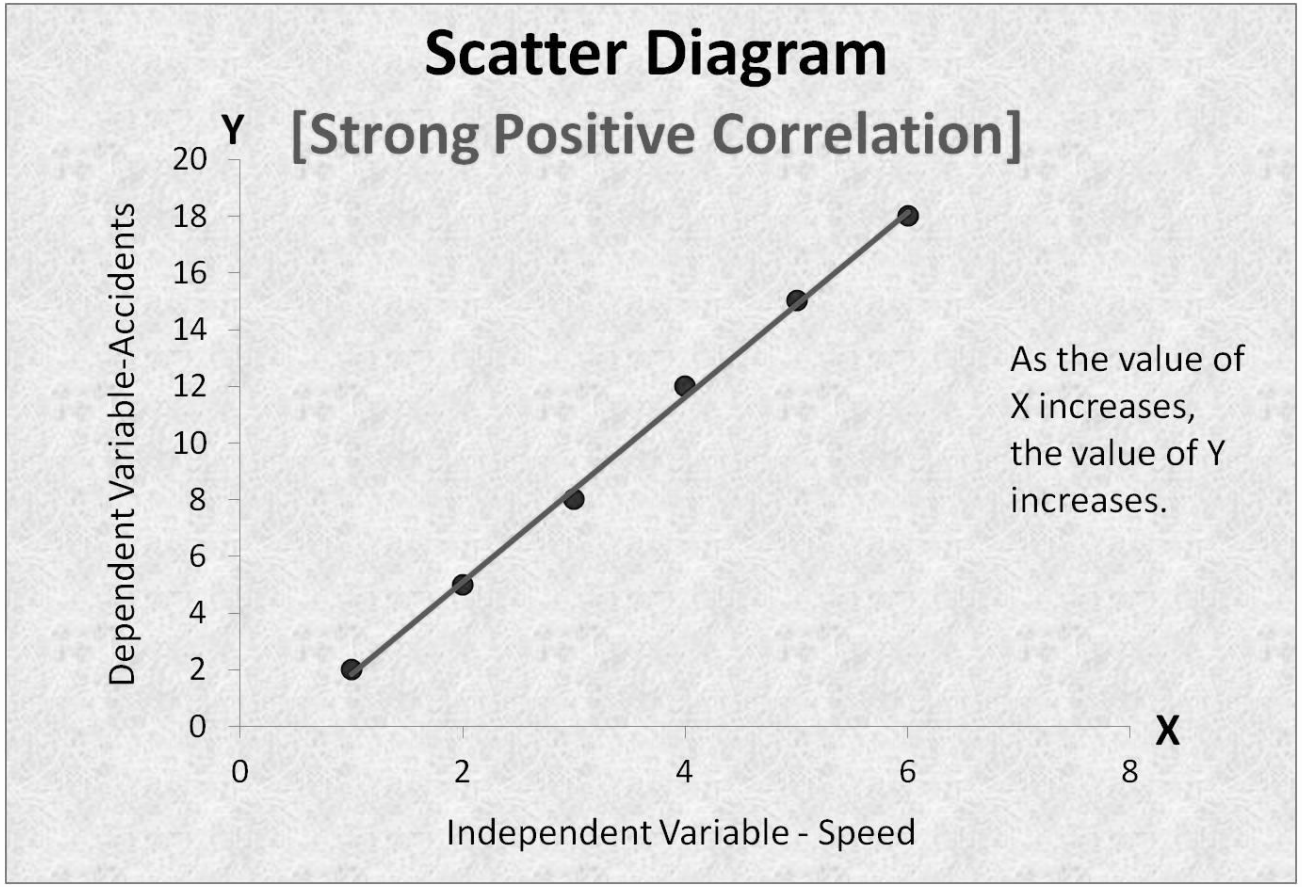


Pareto Chart

- The Pareto chart, named after the Italian economist Vilfredo Pareto, is a powerful tool that helps organizations prioritize issues or factors based on their relative importance or impact.

- It is based on the Pareto principle, also known as the 80/20 rule, which suggests that a majority of consequences (typically around 80%) are often influenced by a minority of causes (approximately 20%).

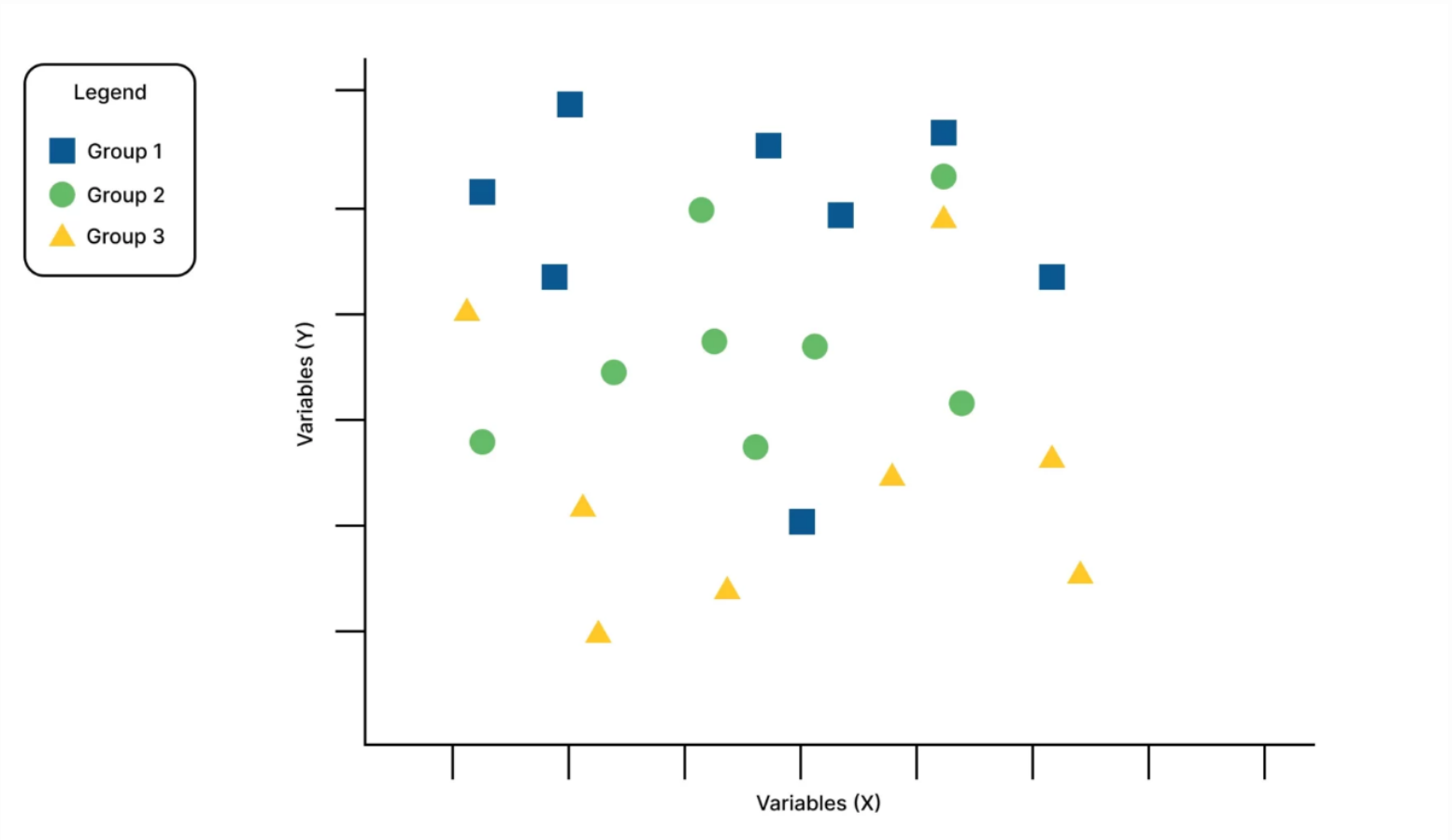
Scatter Diagram



Scatter Diagram

- A scatter diagram, also known as a scatter plot, is a graphical tool used to analyze and visualize the relationship between two variables.
- It plots pairs of numerical data, with one variable represented on the horizontal (x) axis and the other variable on the vertical (y) axis, forming a collection of data points.
- The primary purpose of a scatter diagram is to identify and understand the nature and strength of the relationship between two variables.

Stratification (Flowchart, Run Chart)



Stratification (Flowchart, Run Chart)

- Stratification, also known as a flowchart or run chart, is a quality control tool used to categorize and visually represent data or process steps in a structured manner.
- It involves dividing or grouping data into distinct categories or strata based on specific characteristics or factors, enabling organizations to identify patterns, trends, or potential areas for improvement within each stratum.
- The primary purpose of stratification is to enhance process understanding by revealing insights that may be obscured when data is analyzed as a whole.


thank
you