TYBSc Computer Science

Software Testing – Unit 3 Notes

Quality Concepts

All engineered and manufactured parts exhibit variation. The variation betweensamples may not be obvious without the aid of precise equipment to measure thegeometry, electrical characteristics, or other attributes of the parts. However, withsufficiently sensitive instruments, we will likely come to the conclusion that no twosamples of any item are exactly alike.

Variation control is the heart of quality control. A manufacturer wants to minimizethe variation among the products that are produced, even when doing something relativelysimple like duplicating diskettes.From one project to another, we want to minimizethe difference between the predicted resources needed to complete a projectand the actual resources used, including staff, equipment, and calendar time. In general,we would like to make sure our testing program covers a known percentage ofthe software, from one release to another. Not only do we want to minimize thenumber of defects that are released to the field, we'd like to ensure that the variancein the number of bugs is also minimized from one release to another. (Our customerswill likely be upset if the third release of a product has ten times as many defects asthe previous release.) We would like to minimize the differences in speed and accuracyof our hotline support responses to customer problems. The list goes on and on.

When we examine an item basedon its measurable characteristics, two kinds of quality may be encountered: qualityof design and quality of conformance.
Quality of design refers to the characteristics that designers specify for an item. Thegrade of materials, tolerances, and performance specifications all contribute to thequality of design. As higher-grade materials are used, tighter tolerances and greater
levels of performance are specified, the design quality of a product increases, if theproduct is manufactured according to specifications.
Quality of conformance is the degree to which the design specifications are followedduring manufacturing. Again, the greater the degree of conformance, the higheris the level of quality of conformance.

In software development, quality of design encompasses requirements, specifications,and the design of the system. Quality of conformance is an issue focusedprimarily on implementation. If the implementation follows the design and the resultingsystem meets its requirements and performance goals, conformance quality ishigh

Quality Control

Variation control may be equated to quality control. But how do we achieve qualitycontrol? *Quality control* involves the series of inspections, reviews, and tests usedthroughout the software process to ensure each work product meets the requirementsplaced upon it. Quality control includes a feedback loop to the process thatcreated the work product. The combination of measurement and feedback allows usto tune the process when the work products created fail to meet their specifications.This approach views quality control as part of the manufacturing process.Quality control activities may be fully automated, entirely manual, or a combination
of automated tools and human interaction. A key concept of quality control isthat all work products have defined, measurable specifications to which we may comparethe output of each process. The feedback loop is essential to minimize thedefects produced.

Quality Assurance

Quality assurance consists of the auditing and reporting functions of management.The goal of quality assurance is to provide management with the data necessary tobe informed about product quality, thereby gaining insight and confidence that product
quality is meeting its goals. Of course, if the data provided through quality assuranceidentify problems, it is management's responsibility to address the problemsand apply the necessary resources to resolve quality issues.

Cost of Quality

The *cost of quality* includes all costs incurred in the pursuit of quality or in performingquality-related activities. Cost of quality studies are conducted to provide a base-line for the current cost of quality, identify opportunities for reducing the cost of quality,
and provide a normalized basis of comparison. The basis of normalization isalmost always dollars. Once we have normalized quality costs on a dollar basis, wehave the necessary data to evaluate where the opportunities lie to improve ourprocesses. Furthermore, we can evaluate the effect of changes in dollar-based terms.

Quality costs may be divided into costs associated with prevention, appraisal, andfailure. *Prevention costs* include
- quality planning
- formal technical reviews
- test equipment
- training

Appraisal costs include activities to gain insight into product condition the "first timethrough" each process. Examples of appraisal costs include
• in-process and interprocess inspection
• equipment calibration and maintenance
• testing

Failure costs are those that would disappear if no defects appeared before shipping aproduct to customers. Failure costs may be subdivided into internal failure costs andexternal failure costs.

Internal failure costs are incurred when we detect a defect inour product prior to shipment. Internal failure costs include
• rework
• repair
• failure mode analysis

External failure costs are associated with defects found after the product has beenshipped to the customer. Examples of external failure costs are
• complaint resolution
• product return and replacement
• help line support
• warranty work

THE QUALITY MOVEMENT

Today, senior managers at companies throughout the industrialized world recognizethat high product quality translates to cost savings and an improved bottom line.However, this was not always the case.Quality assurance is an essential activity for any business that produces products tobe used by others. Prior to the twentieth century, quality assurance was the soleresponsibility of the craftsperson who built a product.

SQA Activities

Software quality assurance is composed of a variety of tasks associated with two differentconstituencies—the software engineers who do technical work and an SQAgroup that has responsibility for quality assurance planning, oversight, record keeping,analysis, and reporting.Software engineers address quality (and perform quality assurance and qualitycontrol activities) by applying solid technical methods and measures, conducting formaltechnical reviews, and performing well-planned software testing.

The charter of the SQA group is to assist the software team in achieving a highqualityend product. The Software Engineering Institute [PAU93] recommends a

setof SQA activities that address quality assurance planning, oversight, record keeping,
analysis, and reporting. These activities are performed (or facilitated) by an independentSQA group that:

Prepares an SQA plan for a project.
The plan is developed during project planningand is reviewed by all interested parties. Quality assurance activities performedby the software engineering team and the SQA group are governed by the plan. The
plan identifies
• evaluations to be performed
• audits and reviews to be performed
• standards that are applicable to the project
• procedures for error reporting and tracking
• documents to be produced by the SQA group
• amount of feedback provided to the software project team

Participates in the development of the project's software process description.
The software team selects a process for the work to be performed. The SQAgroup reviews the process description for compliance with organizational policy,internal software standards, externally imposed standards (e.g., ISO-9001), and otherparts of the software project plan.

Reviews software engineering activities to verify compliance with the defined software process.
The SQA group identifies, documents, and tracks deviations fromthe process and verifies that corrections have been made.

Audits designated software work products to verify compliance with those defined as part of the software process.
The SQA group reviews selected workproducts; identifies, documents, and tracks deviations; verifies that corrections have
been made; and periodically reports the results of its work to the project manager.

Ensures that deviations in software work and work products are documentedand handled according to a documented procedure.
Deviations may be encounteredin the project plan, process description, applicable standards, or technical workproducts.

Records any noncompliance and reports to senior management.
Noncomplianceitems are tracked until they are resolved.In addition to these activities, the SQA group coordinates the control and managementof change (Chapter 9) and helps to collect and analyze software metrics.

SQA plan :

Software quality assurance encompasses a broad range of concerns and activitiesthat focus on the management of software quality. These can be summarized in the
following manner [Hor03]:

Standards. The IEEE, ISO, and other standards organizations have produceda broad array of software engineering standards and related documents.Standards may be adopted voluntarily by a software engineeringorganization or imposed by the customer or other stakeholders. The job ofSQA is to ensure that standards that have been adopted are followed andthat all work products conform to them.

Reviews and audits. Technical reviews are a quality control activityperformed by software engineers for software engineers. Their intent is to uncover errors. Audits are a type of review performed bySQA personnel with the intent of ensuring that quality guidelines are beingfollowed for software engineering work. For example, an audit of thereview process might be conducted to ensure that reviews are beingperformed in a manner that will lead to the highest likelihood ofuncovering errors.

Testing. Software testing is a quality control functionthat has one primary goal—to find errors. The job of SQA is to ensurethat testing is properly planned and efficiently conducted so that it has thehighest likelihood of achieving its primary goal.

Error/defect collection and analysis. The only way to improve is tomeasure how you're doing. SQA collects and analyzes error and defect datato better understand how errors are introduced and what software engineeringactivities are best suited to eliminating them.

Change management. Change is one of the most disruptive aspects ofany software project. If it is not properly managed, change can lead to confusion,and confusion almost always leads to poor quality. SQA ensures thatadequate change management practices have been instituted.

Education. Every software organization wants to improve its softwareengineering practices. A key contributor to improvement is education of softwareengineers, their managers, and other stakeholders. The SQA organizationtakes the lead in software process improvement and is akey proponent and sponsor of educational programs.

Vendor management. Three categories of software are acquired fromexternal software vendors—*shrink-wrapped packages* (e.g., Microsoft *Office*),a *tailored shell* [Hor03] that provides a basic skeletal structure that is customtailored to the needs of a purchaser, and *contracted software* that is customdesigned and constructed from specifications provided by the customerorganization. The job of the SQA organization is to ensure that high-qualitysoftware results by suggesting specific quality practices that the vendorshould follow (when possible), and incorporating quality mandates as part ofany contract with an external vendor.

Security management. With the increase in cyber crime and new governmentregulations regarding privacy, every software organization should institutepolicies that protect data at all levels, establish firewall protection forWebApps, and ensure that software has not been tampered with internally.SQA ensures that appropriate process and technology are used to achieve
software security.

Safety. Because software is almost always a pivotal component of humanratedsystems (e.g., automotive or aircraft applications), the impact of hiddendefects can be catastrophic. SQA may be responsible for assessing the impactof software failure and for initiating those steps required to reduce risk.

Risk management. Although the analysis and mitigation of risk is the concern of software engineers, the SQA organization ensures thatrisk management activities are properly conducted and that risk-relatedcontingency plans have been established.

SOFTWARE REVIEWS

Software reviews are a "filter" for the software engineering process. That is, reviewsare applied at various points during software development and serve to uncover errorsand defects that can then be removed. Software reviews "purify" the software engineering
activities that we have called *analysis, design,* and *coding.*

A review—any review—is a way of using the diversity of a group of people to:

1. Point out needed improvements in the product of a single person or team;
2. Confirm those parts of a product in which improvement is either not desired or notneeded;
3. Achieve technical work of more uniform, or at least more predictable, quality than canbe achieved without reviews, in order to make technical work more manageable.

Many different types of reviews can be conducted as part of software engineering.Each has its place. An informal meeting around the coffee machine is a form ofreview, if technical problems are discussed. A formal presentation of software designto an audience of customers, management, and technical staff is also a form ofreview. In this book, however, we focus on the *formal technical review,* sometimescalled a *walkthrough* or an *inspection.* A formal technical review is the most effectivefilter from a quality assurance standpoint. Conducted by software engineers (andothers) for software engineers, the FTR is an effective means for improving softwarequality.

Review Guidelines

Guidelines for the conduct of formal technical reviews must be established in advance,distributed to all reviewers, agreed upon, and then followed. A review that is uncontrolledcan often be worse that no review at all. The following represents a minimumset of guidelines for formal technical reviews:

1. *Review the product, not the producer.* An FTR involves people and egos. Conductedproperly, the FTR should leave all participants with a warm feeling ofaccomplishment. Conducted improperly, the FTR can take on the aura of aninquisition. Errors should be pointed out gently; the tone of the meetingshould be loose and constructive; the intent should not be to embarrass or
belittle. The review leader should conduct the review meeting to ensure thatthe proper tone and attitude are maintained and should immediately halt areview that has gotten out of control.
2. *Set an agenda and maintain it.* One of the key maladies of meetings of alltypes is *drift.* An FTR must be kept on track and on schedule. The reviewleader is chartered with the responsibility for maintaining the meeting scheduleand should not be afraid to nudge people when drift sets in.
3. *Limit debate and rebuttal.* When an issue is raised by a reviewer, there maynot be universal agreement on its impact. Rather than spending time debatingthe question, the issue should be recorded for further discussion off-line.
4. *Enunciate problem areas, but don't attempt to solve every problem noted.* Areview is not a problem-solving session. The solution of a problem can oftenbe accomplished by the producer alone or with the help of only one otherindividual. Problem solving should be postponed until after the review meeting.
5. *Take written notes.* It is sometimes a good idea for the recorder to make noteson a wall board, so that wording and priorities can be assessed by otherreviewers as information is recorded.
6. *Limit the number of participants and insist upon advance preparation.* Twoheads are better than one, but 14 are not necessarily better than 4. Keep thenumber of people involved to the necessary minimum. However, all reviewteam members must prepare in advance. Written comments should besolicited by the review leader (providing an indication that the reviewer hasreviewed the material).
7. *Develop a checklist for each product that is likely to be reviewed.* A checklisthelps the review leader to structure the FTR meeting and helps each reviewerto focus on important issues. Checklists should be developed for analysis,design, code, and even test documents.
8. *Allocate resources and schedule time for FTRs.* For reviews to be effective, theyshould be scheduled as a task during the software engineering process. Inaddition, time should be scheduled for the inevitable modifications that willoccur as the result of an FTR.
9. *Conduct meaningful training for all reviewers.* To be effective all review participantsshould receive some formal training. The training should stress bothprocess-related issues and the human psychological side of reviews. Freedmanand Weinberg [FRE90] estimate a one-month learning curve for every 20people who are to participate effectively in reviews.
10. *Review your early reviews.* Debriefing can be beneficial in uncovering problemswith the review process itself. The very first product to be reviewedshould be the review guidelines themselves.

STATISTICAL SOFTWARE QUALITY ASSURANCE

Statistical quality assurance reflects a growing trend throughout industry to becomemore quantitative about quality. For software, statistical quality assurance impliesthe following steps:

1. Information about software defects is collected and categorized.

2. An attempt is made to trace each defect to its underlying cause (e.g., non-conformance to specifications, design error, violation of standards, poorcommunication with the customer).

3. Using the Pareto principle (80 percent of the defects can be traced to 20 percentof all possible causes), isolate the 20 percent (the "vital few").

4. Once the vital few causes have been identified, move to correct the problemsthat have caused the defects.This relatively simple concept represents an important step towards the creation of

an adaptive software engineering process in which changes are made to improvethose elements of the process that introduce error.

To illustrate this, assume that a software engineering organization collects informationon defects for a period of one year. Some of the defects are uncovered as softwareis being developed. Others are encountered after the software has been releasedto its end-users. Although hundreds of different errors are uncovered, all can betracked to one (or more) of the following causes:

- incomplete or erroneous specifications (IES)
- misinterpretation of customer communication (MCC)
- intentional deviation from specifications (IDS)
- violation of programming standards (VPS)
- error in data representation (EDR)
- inconsistent component interface (ICI)
- error in design logic (EDL)
- incomplete or erroneous testing (IET)
- inaccurate or incomplete documentation (IID)
- error in programming language translation of design (PLT)
- ambiguous or inconsistent human/computer interface (HCI)
- miscellaneous (MIS)

SOFTWARE RELIABILITY

There is no doubt that the reliability of a computer program is an important elementof its overall quality. If a program repeatedly and frequently fails to perform, it matterslittle whether other software quality factors are acceptable.Software reliability, unlike many other quality factors, can be measured directed andestimated using historical and developmental data. *Software reliability* is defined in statisticalterms as "the probability of failure-free operation of a computer program in aspecified environment for a specified time" [MUS87]. To illustrate, program X is estimatedto have a reliability of 0.96 over eight elapsed processing hours. In other

words, if programX were to be executed 100 times and require eight hours of elapsed processingtime (execution time), it is likely to operate correctly (without failure) 96 times out of 100.

If we consider a computer-based system, a simple measure of reliability is *meantime-between-failure*(MTBF), where
MTBF = MTTF + MTTR
The acronyms MTTF and MTTR are mean-time-to-failure and mean-time-to-repair,Respectively
In addition to a reliability measure, we must develop a measure of availability.*Software availability* is the probability that a program is operating according to requirementsat a given point in time and is defined as
Availability = [MTTF/(MTTF + MTTR)] _ 100%
The MTBF reliability measure is equally sensitive to MTTF and MTTR. The availability measure is somewhat more sensitive to MTTR, an indirect measure of the maintainability
of software.

Six Sigma for Software Engineering
Six Sigma is the most widely used strategy for statistical quality assurance in industrytoday. Originally popularized by Motorola in the 1980s, the Six Sigma strategy“is a rigorous and disciplined methodology that uses data and statistical analysis tomeasure and improve a company’s operational performance by identifying and eliminatingdefects’ in manufacturing and service-related processes” [ISI08]. The termSix Sigma is derived from six standard deviations—3.4 instances (defects) per million
occurrences—implying an extremely high quality standard. The Six Sigma methodologydefines three core steps:
• *Define* customer requirements and deliverables and project goals via welldefinedmethods of customer communication.
• *Measure* the existing process and its output to determine current qualityperformance (collect defect metrics).
• *Analyze*defect metrics and determine the vital few causes.
If an existing software process is in place, but improvement is required, Six Sigmasuggests two additional steps:
• *Improve* the process by eliminating the root causes of defects.
• *Control* the process to ensure that future work does not reintroduce thecauses of defects.
These core and additional steps are sometimes referred to as the DMAIC (define,measure, analyze, improve, and control) method.
If an organization is developing a software process (rather than improving anexisting process), the core steps are augmented as follows:
• *Design* the process to (1) avoid the root causes of defects and (2) to meetcustomer requirements.

• *Verify* that the process model will, in fact, avoid defects and meet customerrequirements.

This variation is sometimes called the DMADV (define, measure, analyze, design,and verify) method.

THE ISO 9000 QUALITY STANDARDS

A *quality assurance system* may be defined as the organizational structure, responsibilities,procedures, processes, and resources for implementing quality management[ANS87]. Quality assurance systems are created to help organizations ensure theirproducts and services satisfy customer expectations by meeting their specifications.These systems cover a wide variety of activities encompassing a product's entire lifecycle includingplanning, controlling, measuring, testing and reporting, and improving
quality levels throughout the development and manufacturing process. ISO 9000describes quality assurance elements in generic terms that can be applied to anybusiness regardless of the products or services offered.

To become registered to one of the quality assurance system models contained in ISO9000, a company's quality system and operations are scrutinized by third-party auditorsfor compliance to the standard and for effective operation. Upon successful registration,a company is issued a certificate from a registration body represented by the auditors.Semiannual surveillance audits ensure continued compliance to the standard.The requirements delineated by ISO 9001:2000 address topics such as managementresponsibility, quality system, contract review, design control, document anddata control, product identification and traceability, process control, inspection andtesting, corrective and preventive action, control of quality records, internal qualityaudits, training, servicing, and statistical techniques.

In order for a software organizationto become registered to ISO 9001:2000, it must establish policies and proceduresto address each of the requirements just noted (and others) and then be ableto demonstrate that these policies and procedures are being followed.

Pareto Diagram :

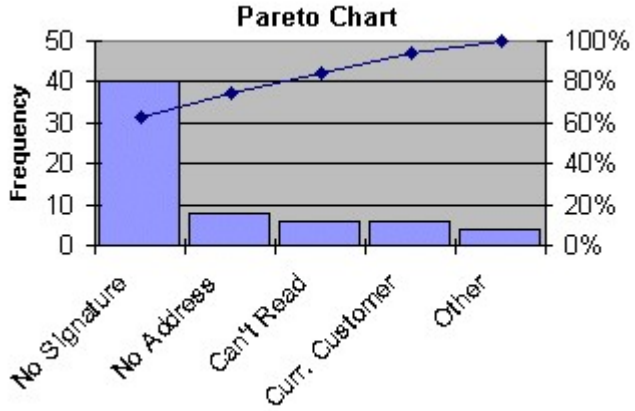
The Pareto chart is one of the seven basic tools of quality control, which include the histogram, Pareto chart, check sheet, control chart, cause-and-effect diagram, flowchart, and scatter diagram.

Typically on the left vertical axis is frequency of occurrence, but it can alternatively represent cost or other important unit of measure. The right vertical axis is the cumulative percentage of the total number of occurrences, total cost, or total of the particular unit of measure. The purpose is to highlight the most important among a (typically large) set of factors. In quality control, the Pareto chart often represents

the most common sources of defects, the highest occurring type of defect, or the most frequent reasons for customer complaints, etc.

Their use gives rise to the 80-20 Rule that is 80 percent of the problems stem from 20 percent of the causes.

Sample Pareto Chart Depiction



How to Construct a Pareto Chart

A pareto chart can be constructed by segmenting the range of the data into groups (also called segments, bins or categories). For example, if your business was investigating the delay associated with processing credit card applications, you could group the data into the following categories:

- No signature
- Residential address not valid
- Non-legible handwriting
- Already a customer
- Other

The left-side vertical axis of the pareto chart is labeled Frequency (the number of counts for each category), the right-side vertical axis of the pareto chart is the cumulative percentage, and the horizontal axis of the pareto chart is labeled with the group names of your response variables.

You then determine the number of data points that reside within each group and construct the pareto chart, but unlike the bar chart, the pareto chart is ordered in descending frequency magnitude. The groups are defined by the user.

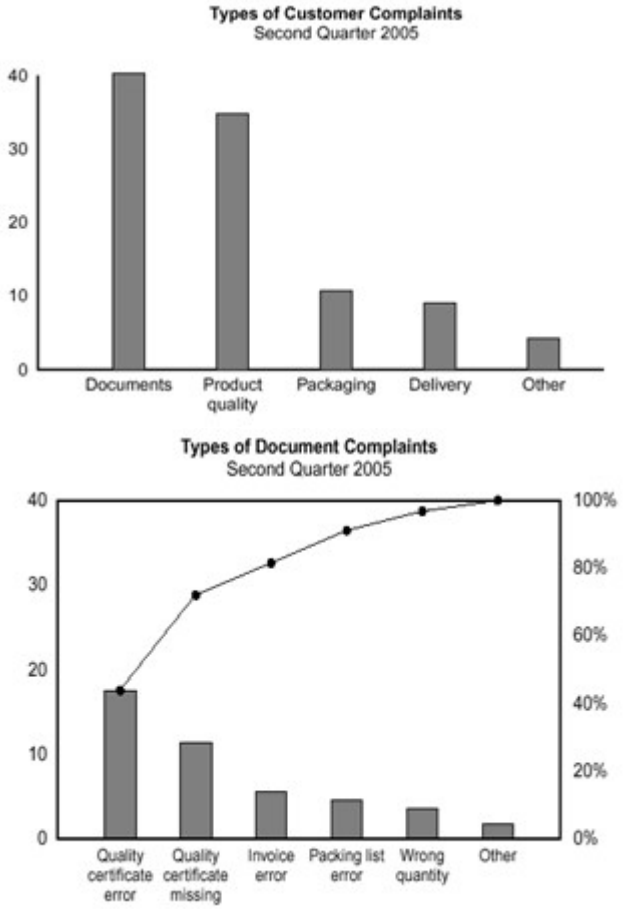
When to Use a Pareto Chart

- When analyzing data about the frequency of problems or causes in a process.
- When there are many problems or causes and you want to focus on the most significant.
- When analyzing broad causes by looking at their specific components.
- When communicating with others about your data.

Pareto Chart Examples

Example #1 shows how many customer complaints were received in each of five categories.

Example #2 takes the largest category, “documents,” from Example #1, breaks it down into six categories of document-related complaints, and shows cumulative values.



Scatter Diagram

The scatter diagram is known by many names, such as scatter plot, scatter graph, and correlation chart. This diagram is drawn with two variables, usually the first variable is independent and the second variable is dependent on the first variable.

A scatter diagram is a graph that shows the relationship between two variables. Scatter diagrams can demonstrate a relationship between any element of a process, environment, or activity on one axis and a quality defect on the other axis.

The scatter diagram is used to find the correlation between these two variables. This diagram helps you determine how closely the two variables are related. After determining the correlation between the variables, you can then predict the behavior of the dependent variable based on the measure of the independent variable. This chart is very useful when one variable is easy to measure and the other is not.

Example

You are analyzing the pattern of accidents on a highway. You select the two variables: motor speed and number of accidents, and draw the diagram.

Once the diagram is completed, you notice that as the speed of vehicle increases, the number of accidents also goes up. This shows that there is a relationship between the speed of vehicles and accidents happening on the highway.

Type of Scatter Diagram

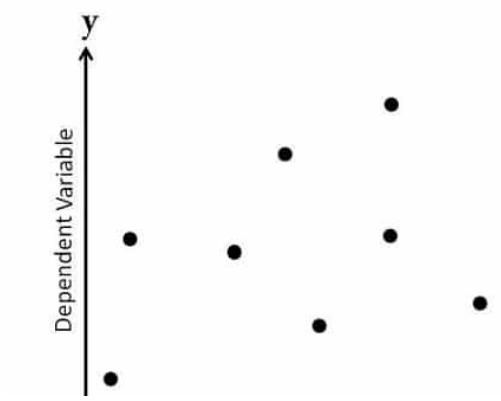
The scatter diagram can be categorized into several types; however, I will discuss the two types that will cover most scatter diagrams used in project management. The first type is based on the type of correlation, and the second type is based on the slope of trend.

According to the type of correlation, scatter diagrams can be divided into following categories:

- Scatter Diagram with No Correlation
- Scatter Diagram with Moderate Correlation
- Scatter Diagram with Strong Correlation

Scatter Diagram with No Correlation

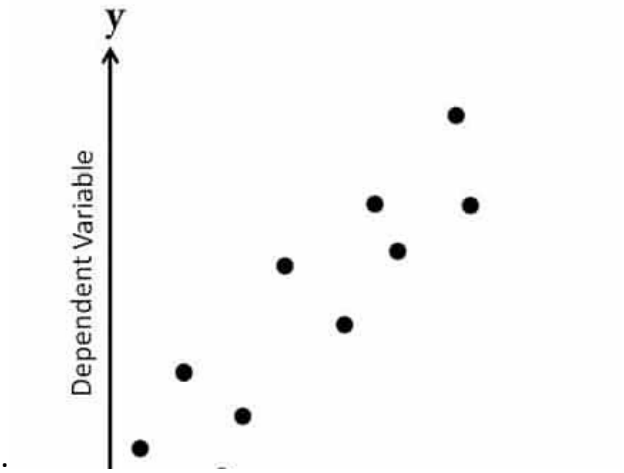
This type of diagram is also known as “Scatter Diagram with Zero Degree of Correlation”.



In this type of scatter diagram, data points are spread so randomly that you cannot draw any line through them.

In this case you can say that there is no relation between these two variables.Scatter Diagram with Moderate Correlation

This type of diagram is also known as “Scatter Diagram with Low Degree of Correlation”.

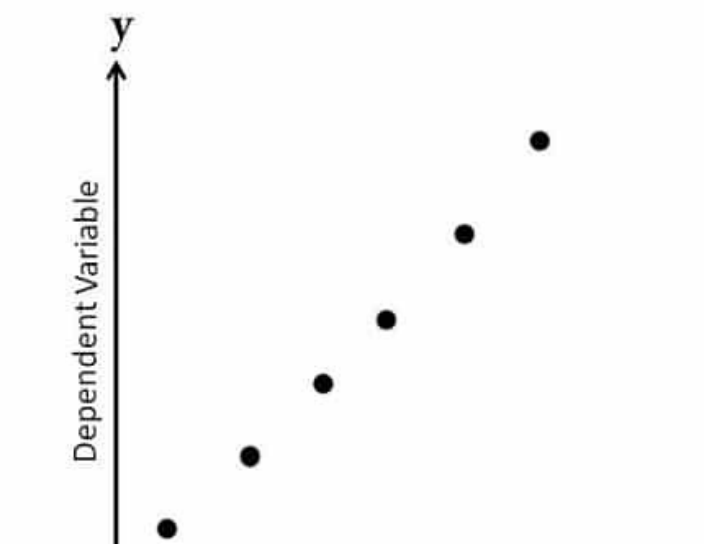


Here, the data points are little closer together and you can feel that some kind of relation exists between these two variables.

Scatter Diagram with Strong Correlation

This type of diagram is also known as “Scatter Diagram with High Degree of Correlation”.

In this diagram, data points are grouped very close to each other such that you can draw a line by following their pattern.



Cause-Effect diagram :

Ishikawa diagrams (also called **fishbone diagrams**, **herringbone diagrams**, **cause-and-effect diagrams**, or **Fishikawa**) are causal diagrams created by Kaoru Ishikawathat show the causes of a specific event.[1]

Common uses of the Ishikawa diagram are product design and quality defect prevention to identify potential factors causing an overall effect. Each cause or reason for imperfection is a source of variation. Causes are usually grouped into major categories to identify and classify these sources of variation.

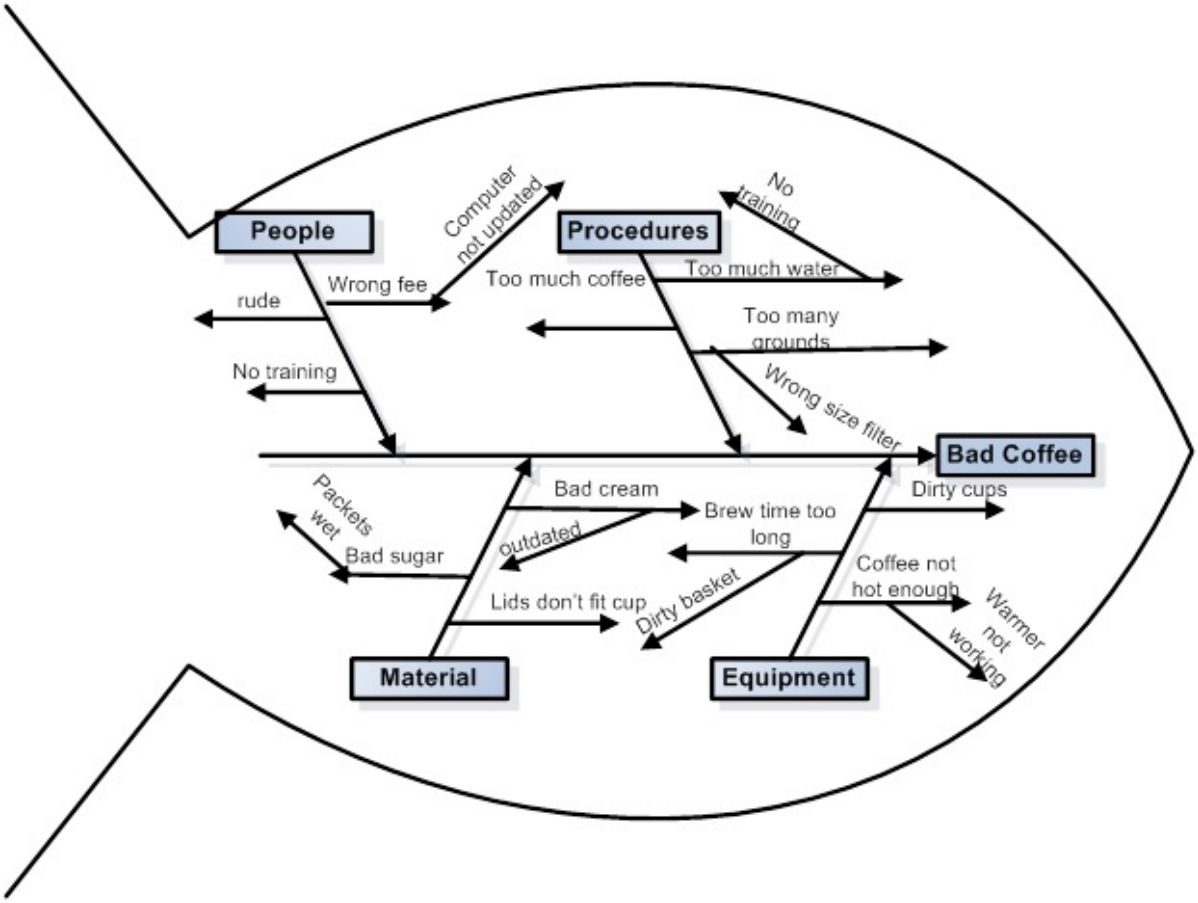
The *defect* is shown as the fish's head, facing to the right, with the *causes* extending to the left as fishbones; the ribs branch off the backbone for major without causes, with sub-branches for root-causes, to as many levels as required.

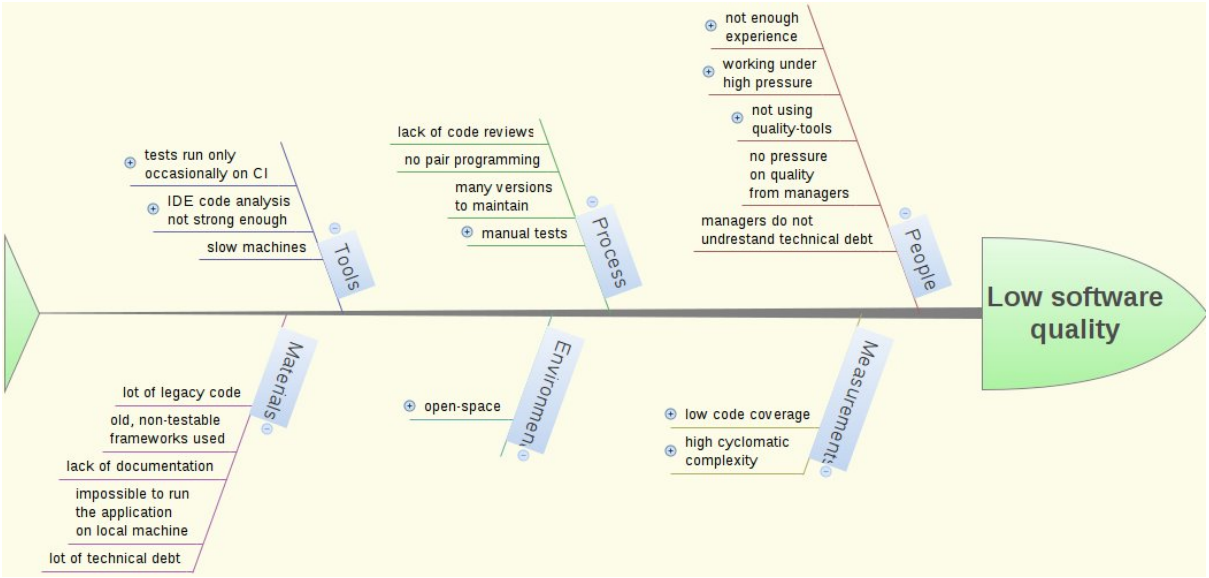
These are the best and most common practices when creating cause and effect diagrams.

- **Identify the problem.** Define the process or issue to be examined.
- **Brainstorm.** Discuss all possible causes and group them into categories.
- **Draw the backbone.** Once the topic is identified, draw a straight, horizontal line (this is called the spine or backbone) on the page, and on the right side, draw a rectangle at the end. Write a brief description of the problem in the rectangle.
- **Add causes and effects.** Causes are added with lines branching off from the main backbone at an angle. Write the description of the cause at the end of the branch. These are usually one of the main categories discussed above. Details

related to the cause or effect may be added as sub-categories branching off further from the main branch. Continue to add branches and a cause or effect until all factors have been documented. The end result should resemble a fish skeleton.

- **Analyze.** Once the diagram has been completed, analyze the information as it has been organized in order to come to a solution and create action items.




Low software quality
Tools
tests run only occasionally on CI
IDE code analysis not strong enough
slow machines
Process
lack of code reviews
no pair programming
many versions to maintain
manual tests
People
not enough experience
working under high pressure
not using quality-tools
no pressure on quality from managers
managers do not undrestand technical debt
Materials
lot of legacy code
old, non-testable frameworks used
lack of documentation
impossible to run the application on local machine
lot of technical debt
Environmen
open-space
Measurements
low code coverage
high cyclomatic complexity