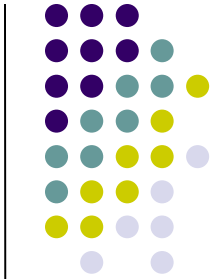
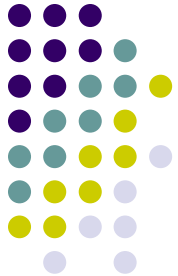
Project Quality management:



- International Standard of Organization ISO defines quality as:
  - “The totality of characteristics of an entity that bear on its ability to satisfy stated or implied needs”
  - **“Conformance to requirements** and **fitness for use**.
  - **Conformance to requirement means**: Project processes and products meet written specifications and **fitness for use**: product can be used as intended.

Main Processes in Project Quality Management



The main purpose of quality management is to ensure that the project will satisfy the needs for which it was undertaken.

- Quality management involves establishing effective relationships with the stakeholders to meet their stated and implied needs.
- MAIN Processes involved in Quality management are:
  - Quality Planning
  - Quality Assurance
  - Quality Control

Quality Planning:

- Includes which quality standards are relevant to the project and how to satisfy them.
- Incorporating quality standards into project design is a key part.
- E.g.: how long it would take to get reply from helpdesk or how long it should take to ship a replacement part of h/w under warranty? or response time of a system or consistent or accurate information is produced.

Quality Assurance:

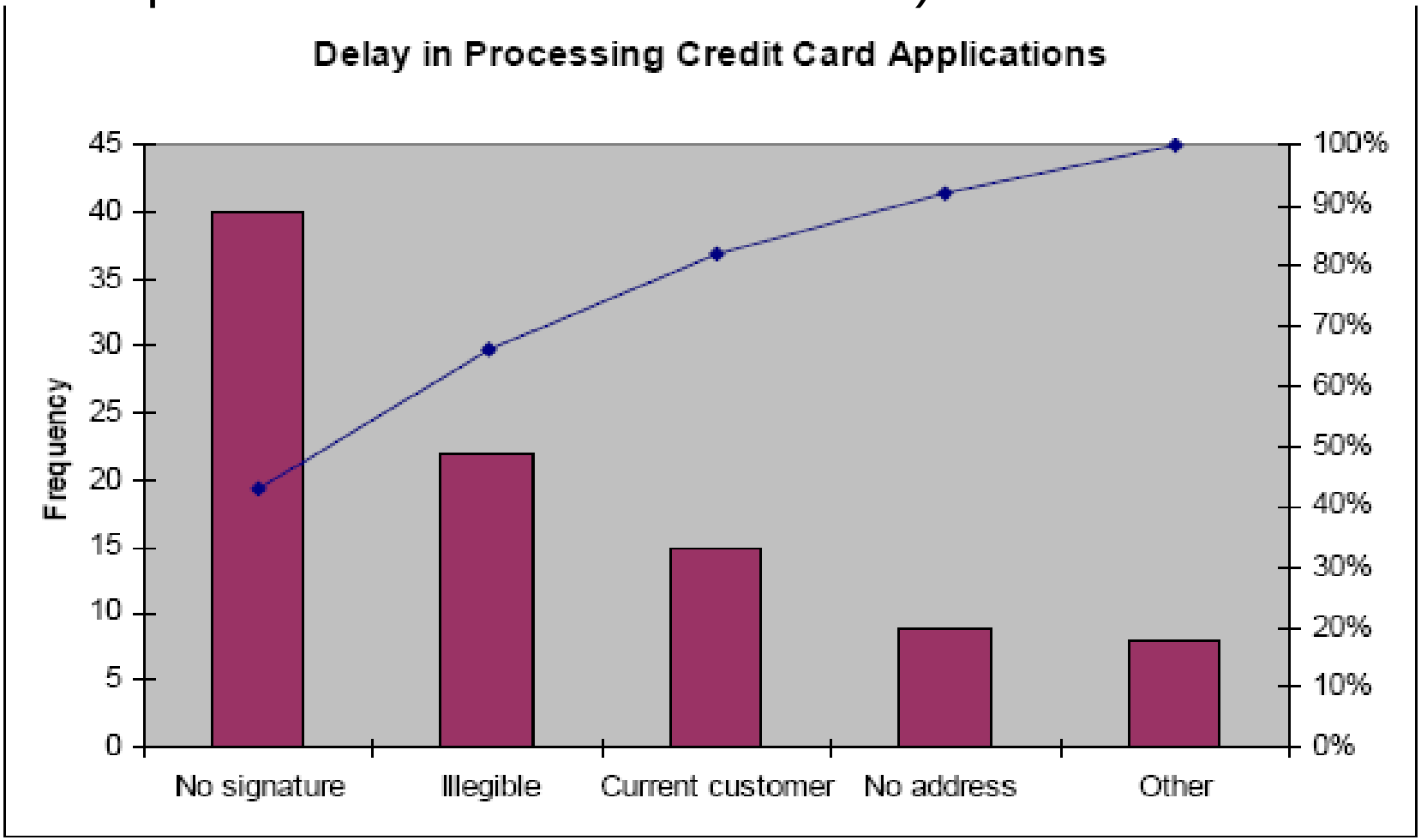
- Involves periodically evaluating project performance to ensure the project will satisfy the relevant quality standards
- It involves taking responsibility of quality during and at the end of the project

Quality Control:

- Involves monitoring specific project results to ensure that they comply with the relevant quality standards while identifying ways to improve over all quality.
- This method is associated with tools and techniques such as Pareto charts, quality control charts and statistical sampling.

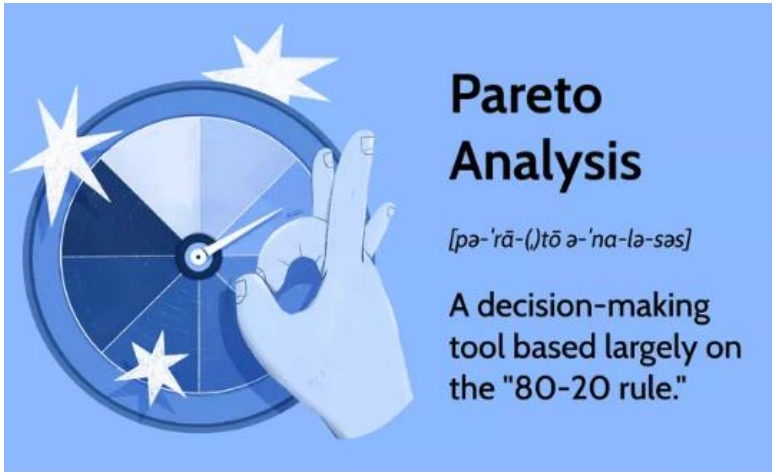
Quality control tool: Pareto Chart

Involves identifying the vital few contributors that account for most quality problems. It is sometimes referred as 80-20 rule (80 % of problems due to 20% of causes)



Quality control tool: Pareto Chart

A Pareto Chart is a graph that indicates the **frequency of defects**, as well as their **cumulative/growing impact**. Pareto Charts are useful to find the defects to prioritize in order to observe the greatest overall improvement.



The importance of software quality

- Increasing criticality of software
- The intangibility of software
- Project control concerns:
  - **errors** accumulate with each stage
  - **errors become more expensive to remove the later they are found**
  - it is **difficult** to control the **error removal process** (e.g. testing)

Quality specifications

Where there is a specific need for a quality, produce a quality specification

- Definition/description of the quality
- Scale: the unit of measurement
- Test: practical test of extent of quality
- Minimally acceptable: lowest acceptable value, if compensated for by higher quality level elsewhere
- Target range: desirable value
- Now: value that currently applies

ISO standards

ISO 9126 Software product quality

Attributes of software product quality

- **External qualities** i.e. apparent to the user of the deliverable
- **Internal qualities** i.e. apparent to the developers of the deliverables and the intermediate products

ISO 14598 Procedures to carry out the assessment of the product qualities defined in ISO 9126

Types of Quality Assessment

- During **software development**, to **assist developers** to build software with the required qualities
- During **software acquisition** to allow a customer to compare and **select the best quality product**
- **Independent evaluation** by assessors rating a software product for a particular community of users

Quality Assessment Goal > Goals

- **Effectiveness** – ability to achieve user goals with accuracy and completeness
- **Productivity** – avoids excessive use of resources in achieving user goals
- **Safety** – within reasonable levels of risk of harm to people, business, software, property, environment etc,
- **Satisfaction** – happy users!

'users' include those maintain software as well as those who operate it.

ISO 9126 software qualities

functionality	does it satisfy user needs?
reliability	can the software maintain its level of performance?
usability	how easy is it to use?
efficiency	relates to the physical resources used during execution
maintainability	relates to the effort needed to make changes to the software
portability	how easy can it be moved to a new environment?

Sub-characteristics of Functionality

- **Suitability**
- **Accuracy**
- **Interoperability**
  - ability of software to interact with other software components
- **Functionality compliance**
  - degree to which software adheres to application-related standards or legal requirements e.g audit
- **Security**
  - control of access to the system

Sub-characteristics of Reliability

- **Maturity**
  - frequency of failure due to faults - the more the software has been used, the more faults will have been removed
- **Fault-tolerance**
- **Recoverability**
  - note that this is distinguished from ‘security’ - see above
- **Reliability compliance**
  – complies with standards relating to reliability

Sub-characteristics of Usability

- **Understandability**
  - easy to understand?
- **Learnability**
  - easy to learn?
- **Operability**
  - easy to use?
- **Attractiveness** – this is a recent addition
- Usability compliance
  - compliance with relevant standards

Sub-characteristics of Efficiency

- **Time behaviour**
  - e.g. response time
- **Resource utilization**
  - e.g. memory usage
- **Efficiency compliance**
  - compliance with relevant standards

Sub-characteristics of Maintainability

- Changeability
  - how easy is software to change?
- “Testability”
- Maintainability conformance

Sub-characteristics of portability

- Adaptability
- “Installability”
- Co-existence
  - Capability of co-existing with other independent software products

Correction of errors

- Errors are more expensive to correct at later stages
  - need to rework more stages
  - later stages are more detailed and less able to absorb change
- Barry Boehm
  - **Error typically 10 times more expensive** to correct at coding stage than at requirements stage
  - 100 times more expensive at maintenance stage

ERROR CORRECTION > For each activity, *define*:

- **Entry requirements**
  - These have to be in place before an activity can be started
  - example: ‘a comprehensive **set of test data** and expected results be prepared and independently reviewed against the system requirement before program testing can commence’

ERROR CORRECTION > For each activity, *define*

- **Implementation requirements**
  - These define **how the process is to be conducted**
  - example ‘whenever an error is found and corrected, *all* test runs must be completed, **including those previously successfully passed**’

ERROR CORRECTION > For each activity, *define*

- **Exit requirements**
  - **an activity will not be completed until these requirements have been met**
  - example: ‘the testing phase is finished only when all tests have been run in succession with no outstanding errors’