- # SCM repositories

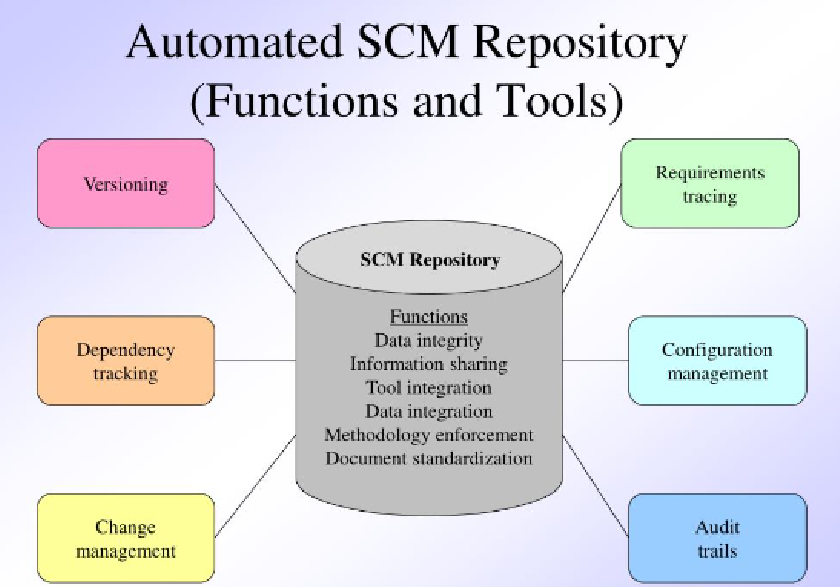
Software Configuration Management (SCM) is a process to systematically manage, organize and control changes in documents, codes and other entities during software development lifecycle.

The primary goal is to increase productivity with minimal mistakes.

SCM Tools: Puppet, ConfigHub, Saltstack, Ansible, Git, BitBucket, Docker & CHEF etc.

SCM Repository is a set of control processes and data structures that allow software teams to manage change in an effective manner.

It manage version control, change control and release control process



SCM Repository Functions

- Data Sharing: task can be performed using one or more resources
- Information Sharing: manage and control multi user access
- Tool Integration: establish data models using SE tools

- Data Integration: ensure consistent and accurate information
- Methodology Enforcement: define entity relationship model
- Document Standardization: define standard approach for creating SE documents

SCM Repository Tools

- Versioning: Save and retrieve all repository object based on there version number
- Dependency Tracking and Change Management: track and manage the changes in repository
- Requirement Tracing:
  - *Forward tracing*: trace design and construction of components and there deliverable result
  - *Backward tracing*: as per work product,which requirement is responsible for which feature
- Configuration management: track a series of configurations representing specific production releases.
- Audit Trails: establish information when, why and by whom changes are made in the repository.

Importance of SCM

It ensures changes to software systems are properly planned, tested and integrated into the final product.

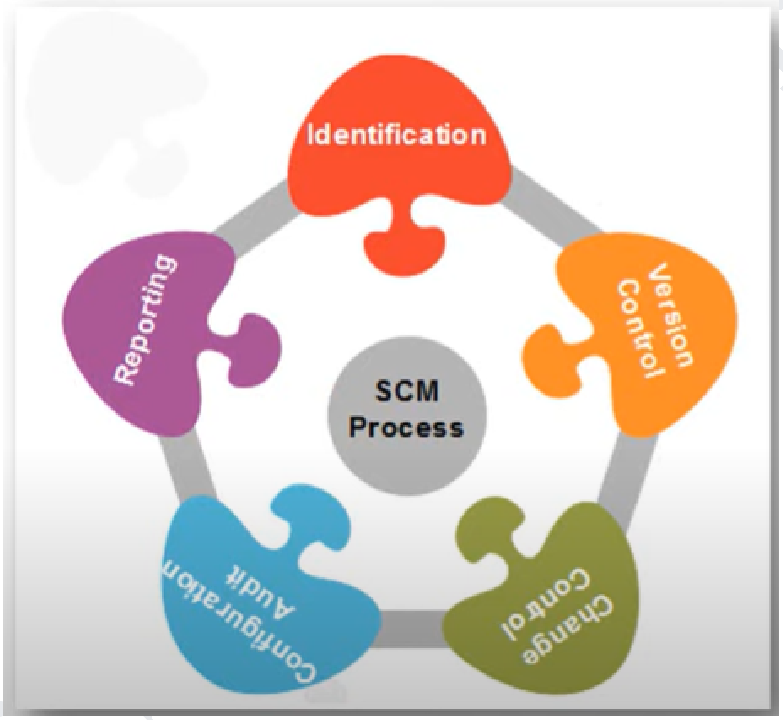
Helps teams to collaborate and coordinate their work, everyone working from the same version of the software system.

It manages and tracks different versions of the system and reverts to earlier versions if necessary.

It ensures that software systems can be easily replicated and distributed to other environments such as test, production and customer sites.

It improves the quality and reliability of software systems as well as increases efficiency and reduces the risk of errors.

- SCM process



<u>Planning and Identification</u>

This method determines the scope of the software system.

This is accomplished by having meetings and brainstorming session with your team

Identifying items like test cases, specification requirements, modules and schedule time.

Identifying each computer software configuration items in the process.

Group basic details of why, when and what changes will be made and who will be in charge of making them

Mention proper naming conventions

Example:

Instead of naming a file login.php it should be named as login_v1.1.php

Version Control Process or Baselines

The aim of this step is to control the alteration and modification done to the product.

It handles different versions of configuration objects that are generated during the software process.

Also focuses on developing ways to track the hierarchy of different versions of the software.

Developing standardized label schemes for all products, revisions and files so that everyone is on the same page.

Change Control Process:

This method is used to ensure that any changes that are made are consistent with the rest of the project.

Examples: to add or edit various configuration items, Change user permissions or changing requirements of clients.

Process:

- Software team send changes to the software configuration manager
- SCM checking examining the overall impact they will have on the project
- Making approved changes or explaining why change were denied
- If it is approved then implement all changes

Configuration Auditing Process:

This process is used to ensure that application will develop as per the project plan and test/verify the application as per scope.

The audit confirms the completeness, correctness and consistency of modified items in the SCM system and tracks action items from the audit to closure.

It mentioned what is new in each version and why the changes were necessary

It ensures that what is built is what is delivered.

- ## Software Quality Assurance Task and Plan, Metrics

Software Quality Assurance (SQA) is simply a way to assure quality in the software.

It is the set of activities which ensure processes, procedures as well as standards are suitable for the project and implemented correctly.

Software Quality Assurance is a process which works parallel to development of software.

It focuses on improving the process of development of software so that problems can be prevented before they become a major issue.

Software Quality Assurance is a kind of Umbrella activity that is applied throughout the software process.

Software Quality Assurance (SQA) encompasses

It is a SQA process

Specific quality assurance and quality control tasks (including technical reviews and a multitiered testing strategy).

It is effective software engineering practice (methods and tools).

SQA is a control of all software work products and the changes made to them

SQA is a procedure to ensure compliance with software development standards (when applicable). It is a measurement and reporting mechanism.

Elements Of Software Quality Assurance

Standards: The IEEE, ISO, and other standards organizations have produced a broad array of software engineering standards and related documents. The job of SQA is to ensure that standards that have been adopted are followed, and all work products conform to them.

Reviews and audits: Technical reviews are a quality control activity performed by software engineers for software engineers. Their intent is to uncover errors. Audits are a type of review performed by SQA personnel (people employed in an organization) with the intent of ensuring that quality guidelines are being followed for software engineering work.

Testing: Software testing is a quality control function that has one primary goal—to find errors. The job of SQA is to ensure that testing is properly planned and efficiently conducted for the primary goal of software.

Error/defect collection and analysis: SQA collects and analyzes error and defect data to better understand how errors are introduced and what software engineering activities are best suited to eliminating them.

Change management: SQA ensures that adequate change management practices have been instituted.

Education: Every software organization wants to improve its software engineering practices. A key contributor to improvement is education of software engineers, their managers, and other stakeholders. The SQA organization takes the lead in software process improvement which is a key proponent and sponsor of educational programs.

Security management: SQA ensures that appropriate process and technology are used to achieve software security.

Risk management: The SQA organization ensures that risk management activities are properly conducted and that risk-related contingency plans have been established.

Major Software Quality Assurance Activities

SQA Management Plan: Make a plan for how you will carry out the SQA throughout the project. Think about which set of software engineering activities are the best for a project. check the level of SQA team skills.

Set The Check Points: SQA team should set checkpoints. Evaluate the performance of the project on the basis of collected data on different checkpoints.

Measure Change Impact: The changes for making the correction of an error sometimes reintroduces more errors. Keep the measure of impact of change on the project. Reset the new change to check the compatibility.

Multi testing Strategy: Do not depend on a single testing approach. When you have a lot of testing approaches available, use them.

Manage Good Relations: In the working environment managing good relations with other teams involved in the project development is mandatory. Bad relationship between the SQA team and the programmers team will impact directly and badly on the project. Don't play politics.

Managing Reports and Records: Document and share QA activities (test cases, defects, client changes) for future reference.

Software Reliability

Software Reliability means Operational reliability. It is described as the ability of a system or component to perform its required functions under specific conditions for a specific period.

Software reliability is also defined as the probability that a software system fulfills its assigned task in a given environment for a predefined number of input cases, assuming that the hardware and the input are free of error.

Software Failure Mechanism

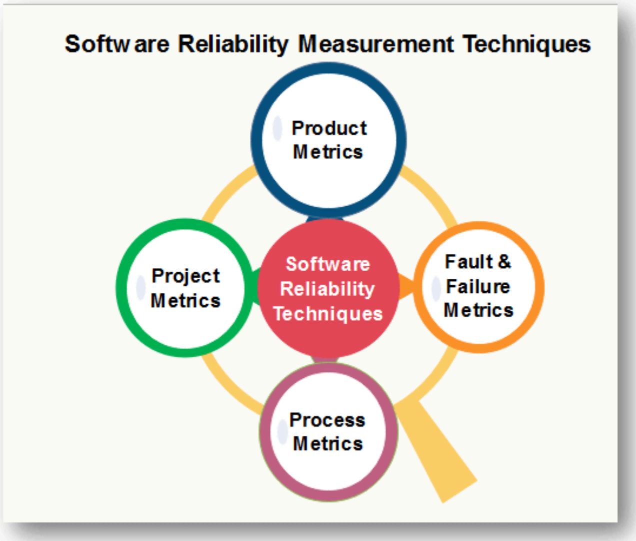
The software failure can be classified as:

- Transient failure: These failures only occur with specific inputs.
- Permanent failure: This failure appears on all inputs.
- Recoverable failure: System can recover without operator help.
- Unrecoverable failure: System can recover with operator help only.
- Non-corruption failure: Failure does not corrupt system state or data.

- Corrupting failure: It damages the system state or data.

Software failures may be due to bugs, ambiguities, oversights or misinterpretation of the specification that the software is supposed to satisfy, carelessness or incompetence in writing code, inadequate testing, incorrect or unexpected usage of the software or other unforeseen problems.

Software Reliability Measurement Techniques


Software Reliability Measurement Techniques
Product Metrics
Project Metrics
Software Reliability Techniques
Fault & Failure Metrics
Process Metrics

1. Product Metrics

LOC/KLOC

Function Point

Complexity

DRE(Defect Removal Efficiency) metric: different quality assurance and control activities applied throughout the development process.

2. Project Management Metrics

Number of software developers

Cost and schedule

Productivity

3. Process Metrics

The effort required in the process

Time to produce the product

Effectiveness of defect removal during development

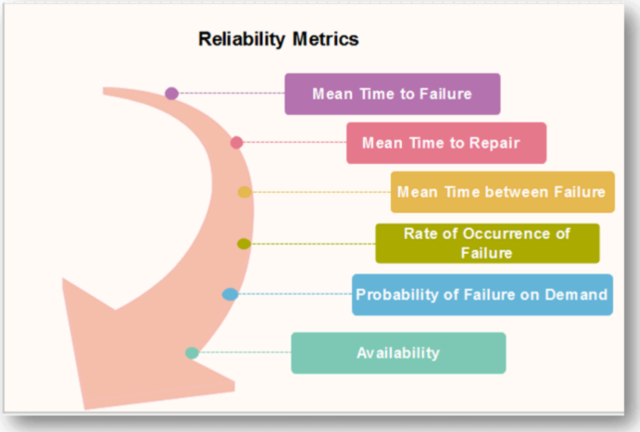
Number of defects found during testing

4. Fault and Failure Metrics

A fault is a defect in a program which appears when the programmer makes an error and causes failure when executed under particular conditions. These metrics are used to determine the failure-free execution software.

To achieve this objective, a number of faults found during testing and the failures or other problems which are reported by the user after delivery are collected, summarized, and analyzed. Failure metrics are based upon customer information regarding faults found after release of the software.

Reliability Metrics


Reliability Metrics
Mean Time to Failure
Mean Time to Repair
Mean Time between Failure
Rate of Occurrence of Failure
Probability of Failure on Demand
Availability

Mean Time to Failure (MTTF)

MTTF is described as the time interval between the two successive failures

Let the failures appear at the time instants t1,t2.....tn.

MTTF can be calculated as

$$\sum_{i=1}^{n} \frac{t_{i+1}-t_i}{(n-1)}$$

Mean Time to Repair (MTTR)

Once failure occurs, some-time is required to fix the error. MTTR measures the average time it takes to track the errors causing the failure and to fix them.

Mean Time Between Failure (MTBF)

We can merge MTTF & MTTR metrics to get the MTBF metric.

MTBF = MTTF + MTTR

Thus, an MTBF of 300 denoted that once the failure appears, the next failure is expected to appear only after 300 hours.

Rate of occurrence of failure (ROCOF)

It is the number of failures appearing in a unit time interval. The number of unexpected events over a specific time of operation. ROCOF is the frequency of occurrence with which an unexpected role is likely to appear.

Probability of Failure on Demand (POFOD)

POFOD is described as the probability that the system will fail when a service is requested. It is the number of system deficiencies given several systems inputs.

POFOD is the possibility that the system will fail when a service request is made.

A POFOD of 0.1 means that one out of ten service requests may fail.

Availability (AVAIL)

Availability is the probability that the system is applicable for use at a given time. It takes into account the repair time & the restart time for the system.

If a system is down an average of four hours out of 100 hours of operation, its AVAIL is 96%.

AVAIL=(MTTF/(MTTF+MTTR))*100%

● Formal Technical Review (FTR)

Formal Technical Review (FTR) is a software quality control activity performed by software engineers.

Objectives of formal technical review (FTR)

- Detect Identification: Identify defects in technical objects by finding and fixing mistakes, inconsistencies, and deviations.
- Quality Assurance: To ensure high-quality deliverables, and confirm compliance with project specifications and standards.
- Risk Mitigation: To stop risks from getting worse, proactively identify and manage possible threats.
- Knowledge Sharing: Encourage team members to work together and build a common knowledge base.
- Consistency and Compliance: Verify that all procedures, coding standards, and policies are followed.
- Learning and Training: Give team members the chance to improve their abilities through learning opportunities.

Steps in FTR

- The Review Meeting
- Review reporting and record-keeping
- A review summary report answers three questions
- Review Guidelines

The Review Meeting

Every review meeting should be conducted by considering the following constraints-

- Involvement of people: Between 3 and 5 people should be involved in the review.
- Advance preparation Advance preparation should occur but it should be very short that is at the most 2 hours of work for each person can be spent in this preparation
- Short duration The short duration of the review meeting should be less than two hours.
- Rather than attempting to review the entire design, walkthroughs are conducted for modules or for small groups of modules.
- The focus of the FTR is on work products (a software component to be reviewed). The review meeting is attended by the review leader, all reviewers and the producer.
- The review leader is responsible for evaluating the product for its deadlines. The copies of product material are then distributed to reviewers. -The producer organizes a “walkthrough” of the product, explaining the material, while the reviewers raise the issues based on their advance preparation.
- One of the reviewers becomes a recorder who records all the important issues raised during the review. When errors are discovered, the recorder notes each.
- At the end of the review, the attendees decide whether to accept the product or not, with or without modification.

Review reporting and record-keeping

During the FTR, the reviewer actively records all issues that have been raised.

At the end of the meeting all these issues raised are consolidated and a review list is prepared.

Finally, a formal technical review summary report is prepared.

A review summary report answers three questions

What was reviewed?

Who reviewed it?

What were the findings and conclusions?

Review Guidelines

Guidelines for the conducting of formal technical review must be established in advance. These guidelines must be distributed to all reviewers, agreed upon, and then followed.

For example,

Guideline for review may include following things

- Concentrate on work products only. That means reviewing the product, not the producers.
- Set an agenda for a review and maintain it.
- When certain issues are raised then debate or arguments should be limited. Reviews should not ultimately result in some hard feelings.
- Find out problem areas, but don't attempt to solve every problem noted.
- Take written notes (it is for record purpose)
- Limit the number of participants and insists upon advance preparation.
- Develop a checklist for each product that is likely to be reviewed.

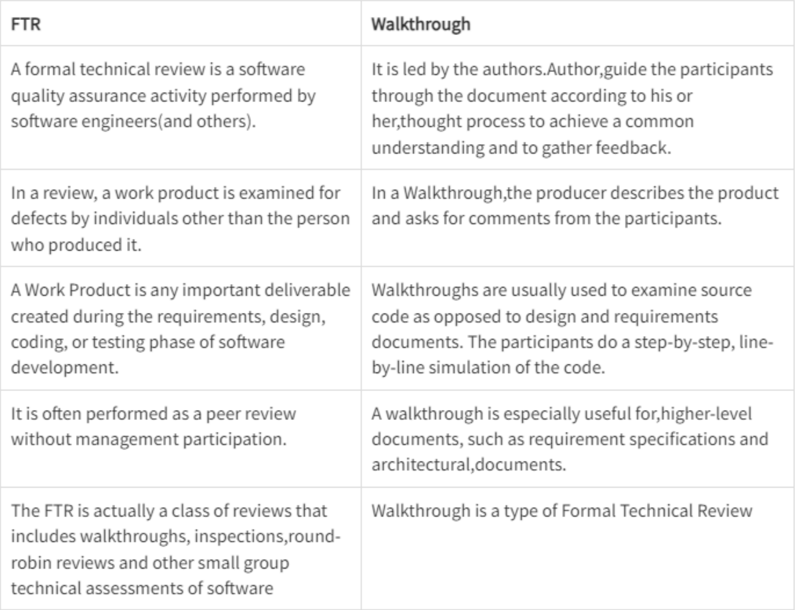
- Allocate resources and time schedules for FTRs in order to maintain a time schedule.
- Conduct meaningful training for all reviewers in order to make reviews effective.

Walkthrough

The walkthrough is a review meeting process but it is different from the Inspection, as it does not involve any formal process i.e. it is a non formal process. Basically, the walkthrough [review meeting process] is started by the Author of the code.

In the walkthrough, the code or document is read by the author, and others who are present in the meeting can note down the important points or can write notes on the defects and can give suggestions about them. The walkthrough is an informal way of testing, no formal authority has been involved in this testing.

As there is an informal way of testing involved so there is no need for a moderator while performing a walkthrough. We can call a walkthrough an open-ended discussion, it does not focus on the documentation. Defect tracking is one of the challenging tasks in the walkthrough.



FTR	Walkthrough
A formal technical review is a software quality assurance activity performed by software engineers(and others).	It is led by the authors.Author,guide the participants through the document according to his or her,thought process to achieve a common understanding and to gather feedback.
In a review, a work product is examined for defects by individuals other than the person who produced it.	In a Walkthrough,the producer describes the product and asks for comments from the participants.
A Work Product is any important deliverable created during the requirements, design, coding, or testing phase of software development.	Walkthroughs are usually used to examine source code as opposed to design and requirements documents. The participants do a step-by-step, line-by-line simulation of the code.
It is often performed as a peer review without management participation.	A walkthrough is especially useful for,higher-level documents, such as requirement specifications and architectural,documents.
The FTR is actually a class of reviews that includes walkthroughs, inspections,round-robin reviews and other small group technical assessments of software	Walkthrough is a type of Formal Technical Review