CD 3 - Test Management

1) Test Plan
2) Scope Management
3) Test Approach / Strategy
4) Criteria
5) Staffing and Training
6) Identify resource requirements
7) Identify test deliverables
8) Size and effort estimation
9) Scheduling
10) Communication management
11) Risk management

Test Plan :

Anchor for the execution, tracking and reporting of the entire testing system

What? - resources, timelines, risks

How?

Scope Management :

Release of products

Feature priority

Deciding which features to test and not to test

Ex: Features which are new and critical, whose failures are catastrophic, complex and defect prone

Test strategy / approach

What testing to test the functionality?

What testing to test the non-functionality?

What configuration / scenario?

Criteria:

- Entry
- Exit
- Suspension
- Resumption

Staffing and Training:

- Clear Accountability
- Listing responsibility
- Complement / Supplement

Identify resource requirements:

- Machine configuration (RAM, Processer Disk)
- Compilers and test data generators
- Software licenses
- Office space, HR, etc.

Identify test deliverables:

- Test plan
- Testcase design specification
- Testcases
- Test logs
- Test summary reports

Estimation:

Size estimation (no. of testcases, no of configurations, etc.)
ex: LOC, FP

Effort estimation (productivity, reuse, robustness)
Ex: Reviews, Audits, No. of testcases developed per day, run per day

Schedule estimation (Gantt chart - taskname, duration, start time, finish time) first task - 2 days

Communication management:

Everyone is kept in sync with right level of detail

Risk management:

Risk identification

through checklist

Organization history, metrics

Informal networking across industry

Risk Quantification

Through probability and impact

Risk Mitigation Planning

Alternate strategies to compact risks

Risk Response

Response to risk

Test Management:

Choice of Standards

Test infrastructure management

Test people management

Choice of Standards:

External standards:

Standards that a product should comply with externally visible and stipulated with external consortia

Ex: All the standard test and acceptance test

Internal standards:

standards to being in consistence and predictability

→ Naming and storage convention for test artifacts

Ex: pmolnnnn.sh p - project mol - module

pmolnnnn.out nnnn - content in module

→ Document standards

Ex: Header line and inline commands

→ Test coding standards

Ex: Meaningful names for variables and reusability

→ Test reporting standards

Ex: Consistent timely reports of the progress of test to stakeholders

Test infrastructure management:

Testcase database:

Contains all relevant information about testcases

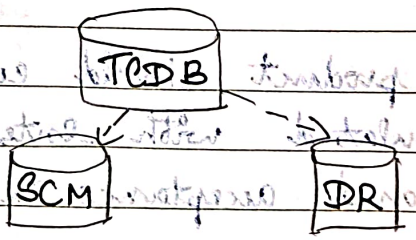
Defect repository:

Contains all relevant information about defects of the product

Configuration management:

Repository and tool (SCM)

It keeps track on change control and version control



Test people management:

Interaction b/w developer, tester and sales person

Test process:

Base lining a test plan
Testcase specification
Update tracebility matrix
Identify the test for automation
Developing and baselining testcases
Executing testcases
Collection and analyzing matrices
Test summary report
Recommending product release criteria

Test plan:

Anchor of the entire testing project
Test plan is baselined into configuration management repository.

Testcase specification:

It includes purpose of the test
Items being tested
Environment (HW, SW, people)
Input data for testcases
Steps to execute the test
Expected results
Comparison of actual and expected results
Relationship b/w this test and another test

Tracebility matrix:

Tool to validate that every requirement is tested
Created during the requirement gathering face itself by filling up unique id for each requirement
Ensure two way mapping b/w requirements and testcases

Criteria for Automation:

Repeated nature of the test
Effort involved in automation
Manual intervention required for the test
Cost of automation tool

Developing testcases:

If automated testcase, then write test scripts in the automation language

Manual:

If manual testcase, then write detailed step by step instructions for executing the test and validates the result

The testcase should capture it documentation for changes made to the testcase, since the original development

All the artifacts of testcases like test scripts, inputs, expected outputs, etc, ... are stored in TEDB and SCM

Executing testcases:

Software Test Automation:

Developing software to test software

Advantages of automation:

Saves time

Focus on creative tasks, not mundane tasks

Reliability

Helps in immediate testing

Protects against attrition of test engineers

Better utilization of global resources

Certain types of testing needs automation only. Example, stress, load testing, etc...

End-to-end, not just test execution alone

Skills needed for automation:

1st generation - Record and playback

2nd generation - Data driven

3rd generation - Action driven

Types of testing amenable for ~~tes~~ automation:

Stress testing

Reliability testing

Scalability testing

Performance testing

Regression testing

Functional testing that requires specialized skills

Automate:

Areas less prone to change
That pertain to standards
Test and obtain management commitment
Test that give good ROI - Return of Investment

Design and architecture for automation:

Test infrastructure
Defect Repository

Architecture: - components
External modules
Scenario and configuration file modules
Testcases and Test framework modules
Tools and Results modules
Ex: IP packet simulator, User login simulator, etc.
Report generator and Report & Metrics modules

Generic Requirements for Test Tool / framework:

No hardcoding in the test suite
Testcase / suite expandability
Reuse of code for different types of testing & testcases
Automatic setup and cleanup
Independent testcases
Testcase dependency
Insulating testcases during execution
Coding standards and directory structure
Selective execution of testcases
Random execution of testcases
Parallel execution of testcases
Looping the testcases
Grouping of test scenarios
Testcase execution based on previous results
Remote execution of testcases
Automatic archival of test data
Reporting scheme
Independent of languages
Portability to different platforms

Selection of Test Tool:

Criteria for selecting test tools:

1) Meeting requirements
2) Technology expectation
3) Training / skills
4) Management aspects

Meeting requirements:

Tools rarely meet all requirements of given product or organization

Forward and backward compatibility issues

Test the test first

Unable to provide the required amount of troubleshooting or error messages to help in analysis

Technology expectation:

Extensibility and customization

Test tools are not 100% cross platform

Instrumented code (libraries linked with source code) impacts performance testing

Training / skills:

Learn new languages or skills

Skill requirements for automation

Management aspects:

Test tool increases the system requirement and requires H/w and S/w to be updated

Measurements and Metrics:

Metrics:

Derive information from raw data with a view to help in decision making

To find out relationship b/w data points, cause and effect, correlation b/w data points, to find out how the data can be used for future planning and continuous improvements

Metrics are thus derived from measurements

Right parameters must be measured

Parameters may pertain to product or to process

Right analysis must be done on the data measured in order to draw crt conclusions

The results of the analysis must be presented in a appropriate form to all the stakeholders to enable them to make decisions

Effort:

Actual time spent on a particular activity / phase

Elapsed days:

Diff b/w start of the activity and completion of the activity

Elapsed days becomes the schedule for the project

Ex: Ordering through web

Effort = 5 mins

Elapsed days = 3 days

A metric can use 1 or more measurement

A Metric can use 1 or more measurement

Ex of measurement:
- Effort spent on testing
- No. of defects
- No. of testcases

Measurement should be at the right level of granularity

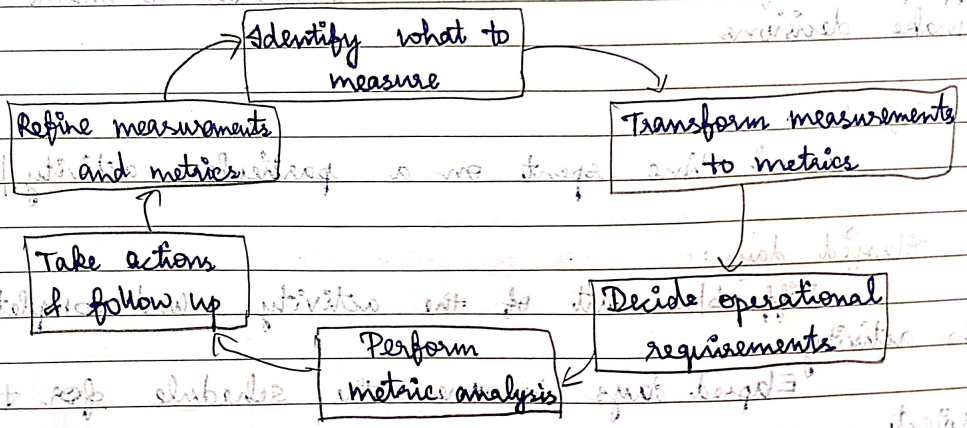
An approach involved in getting granular detail is called data drilling

Granular detail is used by diff people at diff levels

Metrics program:

Continuous and on-going process

Steps in a metrics program:



Types of metrics:

Product metrics

Project metrics

- → Effort variance
- → Schedule variance
- → Effort distribution

Productivity metrics

- → Defect per 100hrs of testing
- → Testcases executed per 100hrs of testing
- → Testcases developed per 100hrs
- → Defect per 100 testcases
- → Defect per 100 failed testcases
- → Testphase effectiveness
- → Closed defects distribution

Progress metrics

Testing defect metrics

- * Defect find rate
- * Defect fix rate
- * Outstanding defects rate
- * Priority defects rate
- * Defects trend
- * Defect classification trend
- * Weighted defects trend
- * Defect cause distribution

Development defect metrics

- * Component wise defect distribution
- * Defect density and defect removal rate
- * Age analysis of outstanding defects
- * Introduced and reopened defects rate

Project notes:

Development defect metrics:

Metrics that help in improving the development activities by mapping the defects to diff components of the product.

Component wise defect distribution:

Defects are assigned to the appropiate developer to fix them

Knowing the components producing more defects helps in defect fix plan and designing in deciding what to release

Defect density and defect removal rate:

Defect density maps the defects in the product with volume of code produced for the product

Defects per kLOC = Total defects in the product / Total executable lines of code in kLOC

AMD - Added, Modified, Deleted code

Defects per kLOC are of the current release is compared with previous releases

Defect per kLOC = Total defects in the product / Total executable AMD lines of code in kLOC

Defect removal rate = (Defects found by verification activities + Defects found in unit testing / Defects found by test teams) x 100

Defects per kLOC should decrease ↓
Defect removal rate increase ↑

Age analysis of outstanding defects:

Age means those defects that have been waiting to be fixed for a long time. Represents the complexity of the defect

Priority should be given to those outstanding defects that are waiting for a long time to be fixed

Introduced and reopened defects trend:

When adding new code or modifying the code, to provide a defect fix, something that was working earlier may stop working - Introduced defect

A fix may not have fixed the problem completely or some modification may have reproduced a defect that was fixed earlier - Reopened defect

Productivity metrics:

Combines measurements and parameters with effort spent on the product.

Used to estimate the release date, quality, cost of release, no. of defects that can be found, etc...

Defects per 100 hrs of testing =

$$= \left(\frac{\text{Total defects found in the product for a period}}{\text{Total hrs spent to get those defects}} \right) \times 100$$

Normalizes no. of defects found in the product with respect to the effort spent

Testcases executed per 100 hrs of testing =

$$\left(\frac{\text{Total testcases executed for a period}}{\text{Total hrs spent in test execution}} \right) \times 100$$

Used to track productivity and judge the product quality

Testcases developed per 100 hrs of testing =

$$\left(\frac{\text{Total testcases developed for a period}}{\text{Total hrs spent in testcase development}} \right) \times 100$$

Defects per 100 testcases =

$$= \left(\frac{\text{Total defects found for a period}}{\text{Total testcases executed for the same period}} \right) \times 100$$

Defects per 100 failed testcases =

$$= \left(\frac{\text{Total defects found for a period}}{\text{Total testcases failed due to those defects}} \right) \times 100$$

Used to find out the granularity of the testcases

Software Measurement:

A measurement is a manifestation of the size, quantity, amount or dimension of a particular attribute of a product or process

Software measurement Principles:

Formulation
Collection
Analysis
Interpretation
Feedback

Classification:

Direct measurement
Indirect measurement

Quality:

Quality defines to any measurable characteristics such as correctness, maintainability, portability, testability, usability, reliability, efficiency, integrity, reusability and interoperability

Quality is something that you immediately recognize but can't define explicitly

It may be conformation to original specification of the product or it may be features and functions of the product or how much is willing to pay for the product, etc...

Kinds of Quality:

Quality of design - specification of an item

Quality of conformance - degree to which the design specifications are followed during manufacturing

Garvin's Quality Dimensions:

Performance Quality

Feature Quality

Reliability

conformance

Durability

Serviceability

Aesthetics

Perception

Mccall's Quality factors:

- Correctness
- Reliability
- Efficiency
- Integrity
- Usability
- Reusability
- Maintainability
- Flexibility
- Testability
- Portability
- Interoperability

Cost of quality:

- Prevention cost
- Appraisal cost - Technical review, testing
- Failure cost - Internal | External
  - Internal → prior to shipping
  - External → after shipping the product

Software Quality Assurance (SQA):

Conformance to explicitly stated functional and performance requirements, explicitly documented development standards and implicit characteristics that are expected of all professionally developed software

Elements of SQA:

SQA is a software engineering umbrella activity applied at each step of the software process

Standards
Reviews and audits
Testing
Error / Defect collection and analysis
Change management
Education
Vendor management
Security management
Safety
Risk management

Quality control (QC):

Software engineering actions that help to ensure that each work product meet its quality goals

Quality Assurance (QA):

Consists of set of auditing and reporting functions that assess the effectiveness and completeness of QC actions

QA system is defined as the organizational structure, responsibilities, procedures, processes and resources for implementing quality management

SQA Tasks:

Prepares SQA plan for a project: Ex: Audits, Reviews, ...

Participates in the development of project software process description

Reviews software engineering activities to verify compliance with the defined software process

Audits designated software work products to verify compliance with those defined as part of software process

Ensures that deviations in software and work products are documented

Records any non-compliance and reports to senior management

6 Sigma:

Statistical quality assurance strategy that uses data and statistical analysis to measure and improve a company's operational performance by identifying and eliminating defects in manufacturing and service related processes

DMAIC

- Define
- Measure
- Analyze
- Improve
- Control

Test Maturity Model (TMM):

It is a 5 level model that provides a framework to measure the maturity of the testing processes

Based on CMM - Capability Maturity Model

Helps in assessment and enhancement of the testing process

TMM - attempts to achieve zero defects

Levels:

1. Initialization
2. Definition
3. Integration
4. Measurement and Management
5. Optimization

Steps of measurement process :

Formulation - measures and metrics

Collection - data to derive the metrics

Analysis - computation of metrics

Interpretation - evaluation of metrics

Feedback - recommendations / improvements

Measurement - Principles :

Metric should have desirable mathematical properties

Ex: 0 - Absence 1 - Maximum 0.5 - Halfway

When a metric represents a software characteristic that increases ↑ when +ve traits occur or decreases ↓ when undesirable traits occur, value of the metric should increase or decrease in some manner

Each metric should be validated empirically in a wide variety of content before being published / used to make decisions

Whenever possible the data collection and analysis should be automated

Valid statistical techniques should be applied to establish relationships b/w internal product attributes and external quality characteristics

Interpretative guidelines and recommendations should be established for each metric

Requirements related metrics:

These metrics examine the requirements model with the intent of predicting the size of the resultant system

FP metrics - Function Point:

used as a means of measuring the functionality delivered by a system

used to: * estimate cost/effort required to design code and test the software

* predict no. of errors that will be encountered during testing

* forecast no. of components / source lines in the system

Information domain values:

No. of external inputs EI

No. of external outputs EO

External enquiries no. of (done by user) EQ

No. of internal logical files ILF

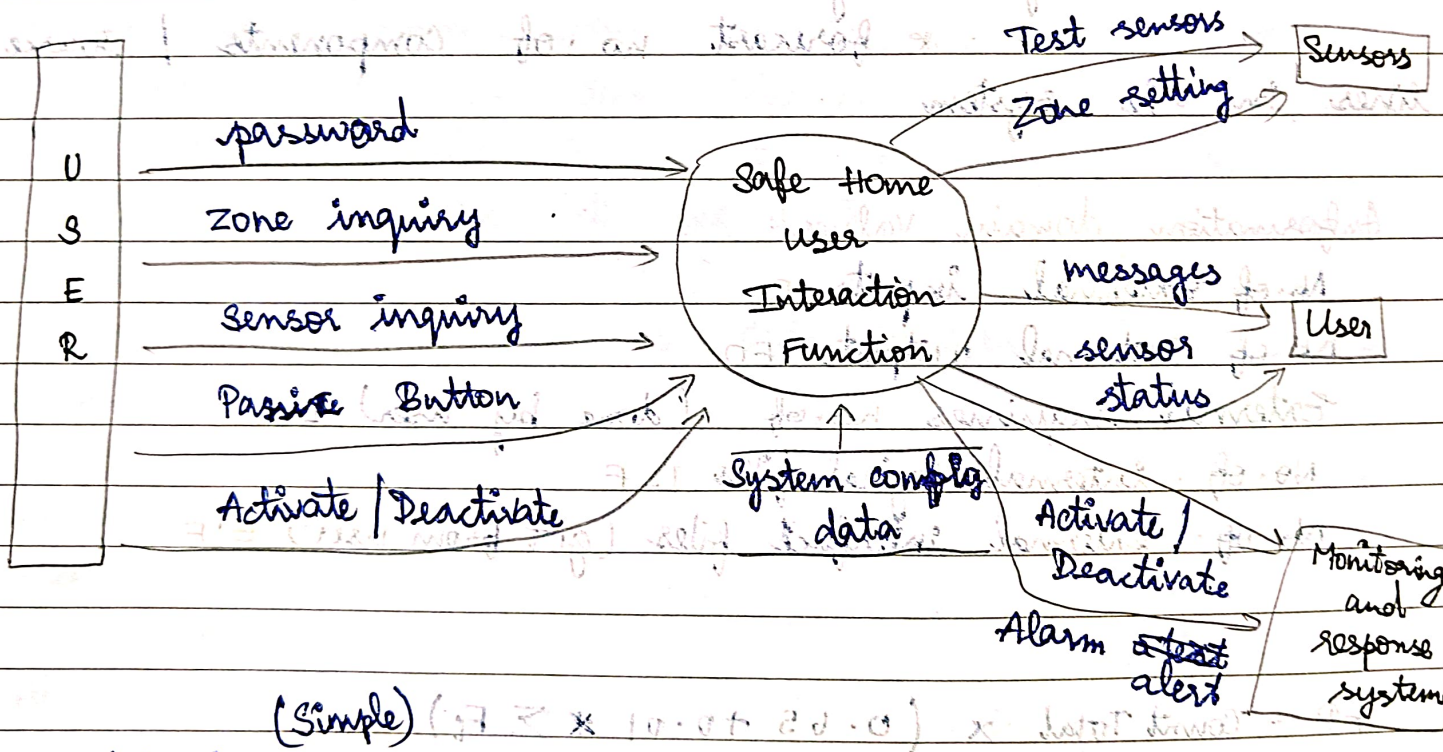
No. of external interface files (got from user) EIF

$$FP = \text{Count Total} \times (0.65 + 0.01 * \sum F_i)$$

F_i $i = 1$ to 14 ⇒ value adjustment factors

Info Domain value	count	Weighting Factors Simple	Avg	Complex	simple
EI	3	3	4	6	9
EO	2	4	5	7	8
EQ	2	3	4	6	6
ILF	1	7	10	15	7
EIF	4	5	7	10	20
					50

Count Total : 50



(Simple)

Count Total = $3 \times 3 + 2 \times 4 + 2 \times 3 + 1 \times 7 + 4 \times 5 = 50$

$\Sigma F_i = 46$

$FP = 50 \times (0.65 + (0.01 * 46)) = 50 \times 1.11 = 55.5 = 56$

FP - used to estimate the overall implemented size of safe home user interaction fn

EI:

Password
Panic button
Activate | Deactivate

EO:

messages
Sensor status

EQ:

Zone inquiry
Sensor inquiry

ILF:

System config data

EIF:

Test sensors
Zone setting
Activate | Deactivate
Alarm ~~a~~test alert