

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, and tracking of 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

e.g.: Features which are new and critical, whose failures are catastrophic, complex and defect-prone, and maintainable

Test strategy / approach (and scenarios, difficulties) maintenance traffic

what to test - the functionality? (which scenario to test)

what to test - the non-functionalities? (which scenario to test)

what configuration / scenario? (which test to run)

Criteria:

Entry
Exit

Suspension
Resumption

Staffing and Training:

Clear Accountability

Listing responsibility

Complement / Supplement

Identify resource requirements:

Machine configuration (RAM, Processor, Disk)

Compilers and test data generators

Software licenses

Office space, HR, etc. etc., including all soft factors

Identify test deliverables:

Test plan

Testcase design specification

Testcases

Test logs

Test summary reports

Estimation: size of similar (existing) how can we start with size?

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, duration per day

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

Communication management:

Everyone is kept in sync with right level of detail

Risk management: test soft. engineering aspects don't emphasis

Risk identification: telology - of development engs through checklist - need two main stages

Organization history, metrics, structure

Informal networking across industry network

Risk Quantification: stakeholders, project staff

through probability and impact assessment

Risk Mitigation Planning: mitigate risk

Alternate strategies to compact risks

Risk Response

Response to risk

Test Management: to make maintenance favorable like structure

Choice of Standards

Test infrastructure management

Test people management

Choice of Standards: provide one check point

External standards:

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

Ex: All the standard test and acceptance test

Standard test: standard test and acceptance test

Acceptance test: when user requirements will implemented

Internal standards:

Refers to standards for being in consistency 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

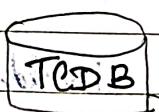
Configuration management:

Repository and tool (SCM)

It keeps track on change control and version

control

New changes



changes in test database

introduced derivative after



changes in defect database

: defect database

Test people management:

Interaction b/w developer, tester and sales person

Test process:

Base lining a test plan ^{test will be written later}

Testcase specification ^{written in informal terms}

Update traceability matrix ^{in requirement document}

Identify the test for automation ^{alternatives by test}

Developing and baselining testcases

Executing testcases

Collection and Analyzing matrices ^{of automation of requirements}

Test summary report

Recommending product release criteria

Test plan: ^{document defining test environment and methodology}
Anchor of the entire testing project ^{test cases}

Test plan is ^{is} baselined into configuration management repository ^{traceability matrix for structure of QA}

Testcase specification:

It includes purpose of the test

Item 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

Traceability matrix:

Tool to validate that every requirement is tested

Created during the requirement gathering phase 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 testeship, start a similar way

Effort involved in automation

Mannual intervention required for the test

list of automation tools for fast software development

agentes patologici han causado

testcases: *exists* *not exists*

If automated testcase, then write test scripts in JUnit

language

Wetzel's Wallaby - *Macropus rufogularis*

If manual testcase, then write detailed step by step

ons for executing the test and validates the results.

The testcase should capture documentation for the

the testcase, since the original development by test

All the artifacts of testcases like test scripts, logs, screenshots etc.

outputs, etc, ... are stored in TEDB and ECM

Digitized by srujanika@gmail.com

testcases: tank with fire medium, establish fire

the last sentence is the same as the first sentence

the *lateral* (flexion, etc., off) "transversal"

It is also suggested that the dog

10. 1. 1957

Planned by [unclear]

children determine how alert to remain?

first written and first left with girlfriend

Page 1 of 1

patient with long-standing mild to moderate pain after stroke.

most of the time were very interested at first but soon lost interest.

Stomach contents: Liver and gizzard.

that have been prepared and written up by our members can

Wertorientierung fast konstitutiv

gerichtet fast nur sozialen Anliegen

Wertorientierung der Wertesysteme

Wertorientierung

Wertorientierung kann nicht mit allen Werten übereinstimmen

Wertorientierung

Wertorientierung ist abhängig von sozialer Rolle

Wertorientierung kann die Wertesysteme unterscheiden

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 ~~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 ~~to~~ automation:

Stress testing

Reliability testing

Scalability testing

Performance testing

Regression testing

Functional testing that requires specialized skills

Automate:

Areas less prone to change are in ~~existing~~ areas that pertain to standards like ISO 9001, etc.

Test and obtain management commitment for test.

Test that give good ROI - Return of Investment.

Design and architecture for automation.

Test infrastructure with low maintenance cost.

Defect Repository

Architecture: - components

External modules

Scenarios and configuration file modules.

Testcases and Test framework modules.

Tools and Results modules for various drivers.

Ex: IP packet simulator, User login simulator, etc.

Report generator and Report & Metrics modules.

Protocol for interchange

Interfacing traffic at different layers

generic Requirements for Test Tool / framework:

No hardcoding in the test fixture among test cases

Testcase / suite expandability private at unitary level

Reuse of code for different types of testing & testcases

Automatic setup and cleanup - $O(1)$, $O(n)$ test time

Independent testcases

Testcase dependency

Insulating testcases during execution with different class objects

Coding standards and directory structure standard test

Selective execution of testcases instead of entire project

Random execution of testcases

Parallel execution of testcases

Looping the testcases

Grouping of test scenarios

Testcase execution based on previous results assert

Remote execution of testcases

Automatic archival of test data

Reporting schemes

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 HW and SW to be updated

in which to be held
at last suitable outfit
strawhopper outfit
suitable outfit
Hick's outfit
strawhopper outfit

the trunks and strawhoppers the team place about
the wagons to be ready to start
several strawhoppers loaded have been
told but all tell
strawhoppers to dinner because of driving at dinner
and so it has been said at dinner time so

the team is to be ready to start
strawhoppers have strawhoppers
about all needed were put in the outfit but
(one more after being loaded) when dinner time
comes the strawhoppers outfit strawhopper strawhopper
outfit strawhopper outfit

will be supper and will be supper
strawhoppers outfit strawhoppers outfit
strawhoppers outfit strawhoppers outfit
strawhoppers outfit strawhoppers outfit
strawhoppers outfit strawhoppers outfit

Measurements and Metrics: ~~What are they~~

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 updating 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 out 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:

Difference 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 approaches

Ex of measurement:

Effort spent on testing, measurement index

No. of defects

No. of testcases and milestones, two half of

event item kept at, string items and milestones, traffic time

Level of 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

Continuous and on-going process involves taking continuous data events of what is happening at project at every level

Metrics program:

Steps in metrics program:

1. Identify what to measure

Refine measurements

Transform measurements into metrics

Identify what metrics, refine to what

Take actions

Decide operational requirements

for initial follow up

Perform metric analysis

Get feedback

from last step

add P & X

such as math, diag

... Translating result into real world A.

Types of metrics:

Product metrics

Project metrics

- Effort Variance
- Schedule variance
- Effort 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

Productivity metrics

- Defect per 100hs of testing

- Testcases executed per 100hs of testing

- Testcases developed per 100hs

- Defect per 100 testcases

- Defect per 100 failed testcases

- Testphase effectiveness

- Closed defects 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 metrics:

C. cinctus (Gmelin)
C. cinctus (Fabricius)

Development defect metrics: $\frac{\text{Total defects}}{\text{Total executable code}} = \text{Defect density}$

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 appropriate 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 lines of 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

$$\text{Defect removal rate} = \frac{\text{Defects found by verification activities}}{\text{Defects found in unit testing}} \times 100$$

~~(or a measure of the effectiveness of the verification activity)~~

Defects per kLOC should decrease
 Defect removal rate should 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 =

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

Defects per 100 failed testcases =

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

Used to find out the granularity of the testcases

Used to find out the granularity of the testcases

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

= smallest followed by next, then larger, repeat

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

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~~ measurement Principles:

Formulation

Collection

Analysis

Interpretation

Feedback

Classification:

Direct measurement

Indirect measurement

Quality:

Quality refers 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 one 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.

Gravin'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 - avoid at source - prevent failure

Prevention cost - internal inspection, testing, rework

Appraisal cost - Technical review, testing

Failure cost - Internal / external

prior to shipping after shipping the product

internal failure

external failure

defects

rejects

scratches

smudges

Software Quality Assurance (SQA):

保证了 conformance to explicitly stated functional and performance requirements, explicitly documented for development, test standards and implicit characteristics that are expected of all professionally developed software

Elements of SQA:

Qualitative SQA is an software engineering umbrella activity applied at each step of the software process.

Standards

Reviews and audits

Testing

Error or defect collection and analysis

Change management

Education

Vendor management

Security management

Safety management

Risk management

Objectives meeting in contractual terms agreed

of stronger and more rigorous and strict

management policies

Quality Control (QC): (AOQ) & (AQL)
Software engineering actions that help to ensure

that each product meets its quality goals

Quality Assurance (QA): (AOQ) & (AQL)
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

QA Tasks:

Prepares QA plan for a project & 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: Identify business needs, define strategy, Define interventions, determine current baseline.

Measure

Process Flow = 200 minutes → Process A = 0.718

Analyze: Identify a structured method to identify root cause.

Improve: Implement changes to remove defect.

Control: Create a process to monitor and maintain the improvements.

Test Maturity Model (TMM): A framework for assessing the maturity of testing processes.

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

Based on technical capability, Maturity Model

Helps in assessment and enhancement of the testing process.

Attempts to achieve zero defects through continuous improvement.

Levels: Awareness, basic understanding, intermediate?

Initialization, Plan, Do, Check, Act, Block.

Definition

Integration

Measurement and Management

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 same manner

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

Job 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

Interpretive guidelines and recommendations should be established for each metric

estimating techniques based on count F

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:
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) Eo

No. of internal logical files ILF

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

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

$$08 = 2 \times 3 + 1 \times 1 + 2 \times 6 + 1 \times 6 + 8 \times 6 = 60 \text{ Function}$$

$$d1 = 37 \approx$$

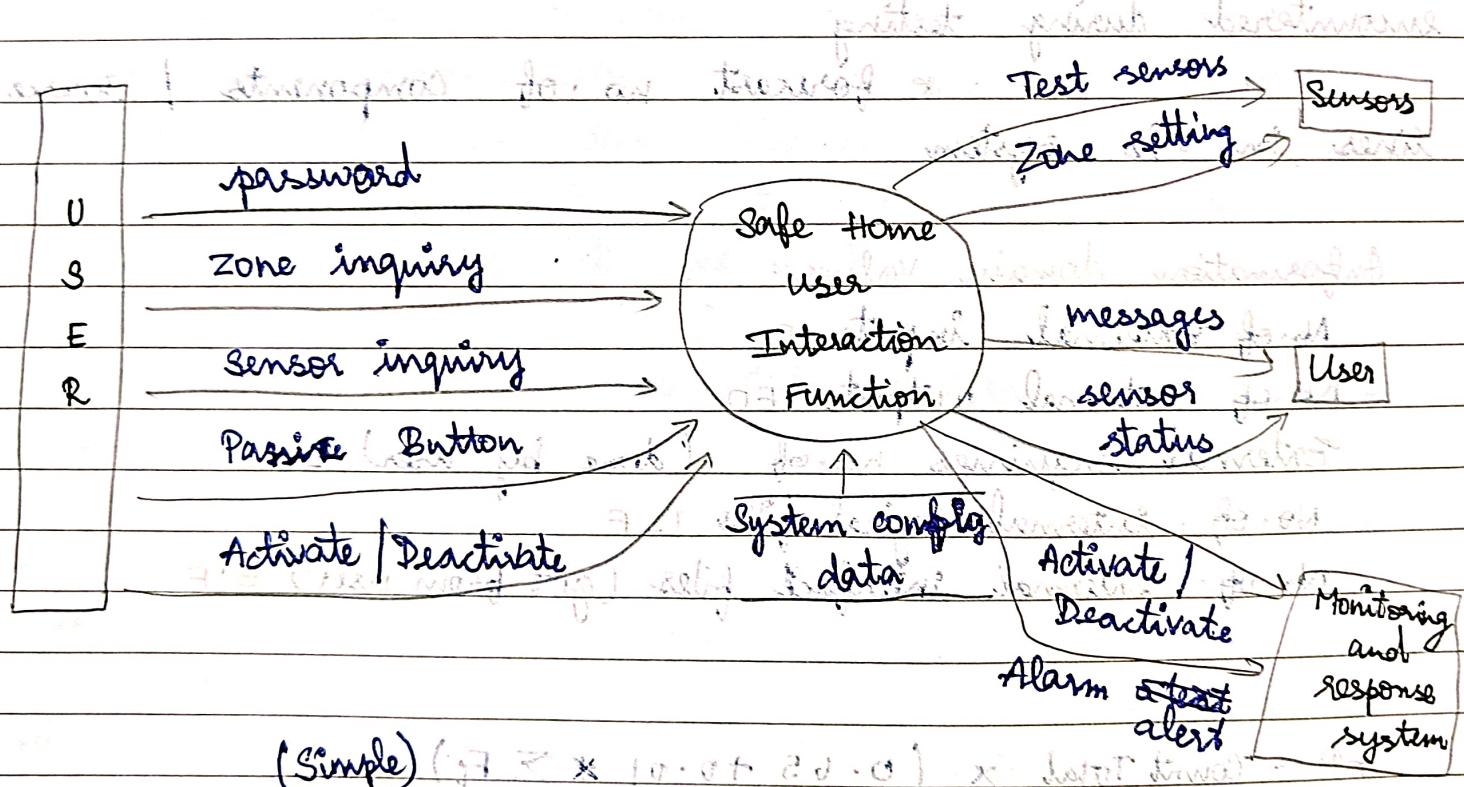
$$d2 = 2 \cdot 22 = 11 \cdot 1 \times 02 = (64 + 10 \cdot 0 + 20 \cdot 0) \times 02 = 97$$

will be multiplied allowing all elements of base = 97

of extracted next unit of 4

$F_i : i = 1 \text{ to } 14 \Rightarrow$ value adjustment factors

Info Domain	count	Weighting Factors	transmissions
Info value transmitter	3	Simple	avg 2.0
Info for zone	2	medium	avg 3.0
Info for sensor	2	medium	avg 3.0
Info for user	4	medium	avg 4.0
Info for system	2	complex	avg 6.0
Info for alarm	1	complex	avg 7.0
Info for test	1	simple	avg 8.0
Info for zone setting	1	simple	avg 9.0
Info for sensor status	1	simple	avg 10.0
Info for user interaction	1	simple	avg 11.0
Info for system config	1	simple	avg 12.0
Info for alarm alert	1	simple	avg 13.0
Info for monitoring and response	1	simple	avg 14.0
Count Total	50		



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

$$\sum F_i = 46$$

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

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

LF:

System config data

EF:

Test sensors

Zone setting

Activate | Deactivate

Alarm ~~off~~ alert