Software Quality and Testing

Definitions

- error / mistake
  - represents mistakes made by people
- fault / defect / bug
  - result of an error
- failure
  - occurs when fault executes
- incident
  - behavior of fault
  - symptoms associated with failure
- root causes
  - analyzing defects to identify their cause
- verification
  - demonstration of consistency, completeness and correctness of the software (Am I building the product right?)
- validation
  - process of evaluating software at the end of the development to ensure compliance with respect to the customer needs and requirements (Am I building the right product?)
- quality
  - totality of the characteristics of an entity that bear on its ability to satisfy state or implied needs
  - testing does not improve quality (testing is determining the quality)
- test data
  - an element of the input domain
- test data set
  - finite set of test data

Quality factors

- testability, maintainability, modularity, reliability, efficiency, usability, reusability, legal requirements/standards

Software testing

- process of executing program to certify its quality
- process of executing program to find faults/failures
- process of executing program to verify that it satisfies the specified functional and non-functional requirements
- examination of behavior
- systematic exploration of a component or a system to find and report defects
- set of techniques
- should be repeatable
- not proving that there are not any faults

Testing types

- static: not executing the code
- dynamic: run the code with some test data

Testing objectives

- verify requirements
- validate stakeholder needs
- build confidence
- find defects
- reduce the level of risk
- find the tests that will uncover the faults in the program

Test techniques

- static
    - review
        - find and remove errors and ambiguities in document before they are used in the development process
    - inspection
    - static analysis
        - enables code to be analyzed for structural defects or systematic programming weaknesses
    - walkthrough
- dynamic
    - structural
        - data flow
            - symbolic execution
            - etc.
        - control flow
            - statement
            - decision
            - condition
            - MC/DC
            - LCSAJ
            - path
            - etc.
    - behavioral
        - functional
            - equivalence partitioning
            - boundary value analysis
            - decision table
            - random
            - state transition
            - etc.
        - non-functional

Bugs

- classification

    - by nature
    - by priority
    - by severity
    - by detection difficulty
        - first order bug: single data parameter triggers failure
        - high order bug: more parameters have specific values

Test selection

- the test set for the testing session
- select a smaller set of tests for functional-based or structural-based testing
- the **test selection criterion** is a guideline for the test selection
- **test adequacy criteria** is set of rules, that determines there are enough selected tests that have been executed
    - the test suite was good enough or not

Test coverage

- measures the amount of testing performed by a set of tests according to some test selection criteria
- the basic coverage is a ratio of the exercised items over all items
- 100% coverage does not mean 100% tested software

Basic coverage types

- requirements coverage
    - the software has been tested against all the requirements in normal usage
    - the software has been tested against all the requirements in abnormal usage
- structural coverage
    - each element of the software been exercised: code statements, decisions, conditions
- architectural coverage
    - actual control and data links been utilized during testing: data paths, control links

Test notions

Test basis

The source of the information or the document that is needed to write test cases and analyze tests

- should be well-designed and structured
- provide traceability between test basis and the various testing work products

Test object

Describes the target of the testing

- SUT - System Under Test
- MUT - Method Under Test
- OUT - Object Under Test
- CUT - Class Under Test
- IUT - Implementation Under Test

Test scenario

A special test object describing an end-to-end requirement to be tested

Test condition

A statement to the test object and can be stated to any part of the test object

- could be verified by one or more test cases

Testware

Artifacts produced during the test process required to plan, design and execute tests, such as documentation, scripts, inputs, expected results, files, databases, environment

Test case

A set of input values, execution preconditions, expected results and execution post conditions

- should be accurate, economic, effective, exemplary, evolvable, executable, repeatable, reusable, traceable
- approaches
    - testing-by-contract: create test cases only for those situations where the preconditions are met
    - defensive test case design: design test cases to accept any input

Test script

Sequence of actions for the execution of a test

Test suite

List of all the test cases that must be executed as a part of a test cycle or regression phase

Test levels

Unit test

Goal: isolate each part of the program and show that the individual parts are correct

- the smallest testable part of an application
- constructed in isolation
- ensures the code meets the specification
- performed by developer
- testing of functionality (mainly) and non-functional characteristics (in some cases)
- mostly structure based methods
    - statement testing
    - decision testing
    - condition testing
    - path testing
    - loop testing
- test basis
    - detailed design, code, data model, component specification
- test objects
    - components, code and data structures, classes, database modules

Integration test

The purpose is to expose defects in the interfaces and in the interactions between integrated components or systems

- types
  - component integration test
  - system integration test
- strategies
  - big-bang
    - integrate everything together in one step
    - no driver, no stub is needed
  - top-down
    - built in stages
    - start with the components what call other components
    - stubs or mocks are used for integration
    - depth-first or breadth-first
  - bottom-up
    - built in stages
    - start with the components that are called by others
    - drivers are used for integration
  - inside-out
    - from inside to out
    - with stubs and drivers
  - outside-in
    - from outs to inside
    - with stubs and drivers
  - branch-wise
    - integrate only branches of application
    - by feature, by functional, by logical connection
- test basis
  - design docs, sequence diagrams, communication protocol specifications, use cases, architecture, external interface definitions
- test objects
  - subsystems, databases, infrastructure, interfaces, APIs, microservices

Stub

A passive component. It replaces the called component

- inputs to the code
- return fixed value, throw exception, calculate a return value based on params

Mock

- outputs from the code
- making assertions on behavior

Driver

Specially designed user interface which enabled test data to be inputted and passed to a sub-system

System test

Focuses on the behavior of the whole system or product in a live environment

- Mostly non-functional tests

- installability, interoperability, maintainability, stress handling, load handling, portability, recovery, reliability, usability
- spikem, soak, volume, configuration, compatibility, environment
- test basis
    - system requirements specification, risk inventory, use cases, epics, state diagrams
- test object
    - application, OS, SUT, system configuration, configuration data

Acceptance test

Provide the end users with confidence that the system will function according to their expectations

- common forms
    - user acceptance testing
    - operational acceptance testing
    - contractual acceptance testing
    - alpha/beta acceptance testing
- test basis
    - business processes, user or business requirements, legal contracts, standards, use cases, system requirements, user documentation, risk analysis record
- test object
    - SUT, system configuration, configuration data, forms, reports, operational or maintenance processes

Maintenance test

Takes place on a system which is in operation in the live environment is called Maintenance testing

- when the system changes
    - additional features are added
    - the system was migrated to a new platform
    - new faults were found and fixed

Impact analysis

- needed
    - specification
    - documented test cases
    - traceability
    - domain knowledge

Test process

Test planning

- defining the scope and objectives
- identify risks
- determine test approaches
- implementing the test policy and test strategy
- determine the required resources
- scheduling test analysis and design tasks
- scheduling test implementation, execution and evaluation
- determine exit criteria

Test policy

Describe the philosophy of organization towards testing

- generally developed by top level IT
- contains definition of testing, processes and procedures
- contains rules for quality products, metrics and improvement approaches

Test strategy

Outline that describes the testing portion of the software development cycle

- info about key issues of testing process for project manager, testers and developers
- describe how the product risks of the stakeholder are mitigated in the test levels

Test approach

Implementation of the test strategy for a specific project

- includes decision made based on the project goals and risk assessment
- analytical approaches
    - risk-based testing
- model-based approaches
    - stochastic testing
- standard-compliant approaches
    - specified by industry-specific standards
- process-compliant approaches
    - agile developments

Entry criteria

Set of generic and specific conditions for permitting a process to go forward with a defined task

Exit criteria

Set of generic and specific conditions, agreed upon with the stakeholders, for permitting a process to be officially completed

- prevent a task from being considered completed where there are still outstanding parts of the task which have not been finished
- determines completeness
- agreed as early as possible

Test control

Test management task that deals with developing and applying a set of corrective actions to get a test project on track when monitoring shows a deviation

- monitor, measure and analyze results
- compare expected and actual progress, coverage and exit criteria
- make corrections

Test analysis

- reviewing the test basis
- evaluating testability of the test basis and test objects
- identify and prioritize test conditions based on analysis
- capture bi-directional traceability

Test design

- predicting how the SUT should behave in given circumstances
- design and prioritize test scenarios and test cases
- design sets of tests
- identify necessary test data
- design test environment and identify required infrastructure and tools

Test design techniques

Specification-based (black box) techniques

Generating test cases based on the specification of the software

- equivalence partitioning (EP)
    - reduce number of test cases to a manageable level
    - partition the input domain to equivalence classes
    - these classes can overlap with each other and we can choose subsection as test data
    - types
        - weak normal EP: we use intersect data
        - strong normal EP: we don't use intersect data
        - weak robust EP: like normal, just we use invalid data too
        - strong robust EP: like normal and weak robust
- boundary value analysis (BVA)
    - focuses on the boundaries (programmer mistakes) - ranges
    - ON, OFF, IN, OUT
- decision table
    - record complex business rules
    - can guide to creating test cases
    - logical conditions, combinations are causing actions
    - 1 test/column
- state transition
    - when actions are triggered by changes of the input conditions or change of state
    - the basis is the state transition diagram
    - transition causes by events
    - all possible events and states are recorded in a state table
    - finite state machines
- use case
    - depends on use cases (between actors and the system)
    - exercise real user processes or business scenarios
    - finds defects in the process flow
    - helping uncover the integration defects
    - basic flow and alternate flows lead to success, exception flow read to exception

Structure-based (white box) techniques

Generating test cases based on the structure of the program

- statement
    - all executable statements in the program should be executed at least once
    - 100% coverage does not mean we have tested everything
- decision
    - all programming decisions for both true and false exist of conditions should be executed
    - 100% coverage does not mean we have tested everything, but it guarantees 100% statement coverage too
- branch
    - all branches in the program should be executed at least once

- condition
- path
    - all execution paths in the program should be executed at least once
    - exponential in the number of conditional branches
    - expensive cost
    - not all path is executable
    - each cycle must be executed k (constant) times

Control Flow Graph

- nodes are basic blocks
- a basic block is a sequence of statements
- edges represent the control flow

Experience-based techniques

Useful when not having enough time to execute a full structured test set

- it demands tester's experience to reveal the most failure
- error guessing
    - the tester is trying to anticipate what defects might be present and design tests to special expose them
- exploratory testing
    - design test cases by the information what was gained in the previous test case
- checklist-based
    - remainder list of what to be tested
    - a guide list for testing
- attack testing
    - focuses on invoking specific type failure

Test oracle

A function that determines easily whether or not the results of executing a program under test conform with the specifications

Test implementation

- developing and implementing the test cases
- creating test data
- prepare expected results
- developing test procedures
- creating test suites
- verify test environment

Test execution

- record identities and versions of SUT, test tools and testware
- executing test procedures
- logging outcome and actual results
- reporting fails as incidents and analyzing them
- repeating test activities

Regression testing

Testing of a previously tested program following modification to ensure that defects have not been introduced or uncovered in unchanged areas of the software as the result of the changes made

Test completion

- evaluating exit criteria (definition of done)
- checking test results against the criteria
- determining whether more tests are needed
- reporting
    - archiving useful testware
    - writing test summary
    - communicating effectively the findings
    - providing analyzed information and metrics
    - assessment of defects remaining

Test closure

- checking deliverables have been delivered
- finalizing and archiving the testware, the test environment and the test infrastructure for later reuse
- analyzing lessons (retrospective)

Test documentation

- planning
    - test plan
- monitoring and control
    - progress reports, risk inventory, resource and effort reports
- test analysis
    - test condition documents
- test design
    - test cases, test coverage items, test data requirements, test environment documents
- test implementation
    - test procedures, test scripts, test suites, test data, test environment
- test execution
    - test logs, test results, test statuses
- text cycle completion
    - change requests, product backlog items, test summary
- should be traceable documents

Test monitoring and control

Test management task that deals with the activities related to periodically checking the status of a test project

- tracking well-defined metrics
- providing feedback
- estimating future courses

Quality

- quality depends on the asked stakeholder

Risk

A factor that could result in future negative consequences, usually expressed as likelihood and impact

- managing and reducing risk
- carry out risk analysis
    - prioritize tests
    - apportion time
    - understand the risk
- use risk to determine what to test first/most or not test

Risk management

- risk identification
    - which risks might affect the project and document their characteristics -> raw list of risks
- risk analysis
    - assess the impact and likelihood of occurrence
    - prioritize
- risk response
    - avoidance: change the project plan to eliminate the risk
    - transference: shift the consequences of the risk to another party
    - mitigation: reduce the probability or consequence of risk to an acceptable threshold
    - acceptance: not changing the project
- identify risks as early as possible
- test risks: poor test basis, aggressive delivery, lack of test expertise, bad estimation, poor test coverage

Cost of testing

- test cost and correction cost
    - the two have optimum (there is a limit, when not efficient testing more)

Configuration management

Process of identifying and defining the items in the system, controlling the change of these items through their lifecycle, recording and reporting the status of items and change requests, and verifying the completeness and correctness of items

- software changes, create a set of versions in maintained and managed form

Change management

- tracking change requests from customers and developers
- determine costs and impact of changes
- deciding about the implementation

Version management

- tracking versions of system components
- should be accessible and not interfere each other

Build management

- process of assembling program components, data and libraries into executable system

Release management

- prepare software for external release for customer use