Presentation on Testing

What is Testing?

Testing is an iterative process that is carried out in conjunction with implementation.

Take measures to check the quality, performance, or reliability of (something), esp. before putting it into widespread use or practice.

is an investigation conducted to provide stakeholders with information about the quality of the product or service under **test**.

is the most common way of checking that it meets its specification and does what the customer wants.

is a critical element of software quality assurance and represents the ultimate review of specification, design, and code generation.

Why Testing?

To improve quality

To verify and validate

For reliability estimation

Objectives of Testing

To demonstrate to the developer and the customer that the software meets its requirements.

To discover faults or defects in the software where the behavior of the software is incorrect, undesirable or does not conform to its specification.

Principle of Testing

All tests should be traceable to customer requirements.

Tests should be planned long before testing begins.

The Pareto principle implies that 80 percent of all errors uncovered during testing will likely be traceable to 20 percent of all program components. The problem, of course, is to isolate these suspect components and to thoroughly test them.

Testing should begin “in the small” and progress toward testing “in the large.”

Test Cases

A rich variety of test case design methods, which provide the developer with a systematic approach to testing.

A test case in software engineering is a set of conditions or variables under which a tester will determine whether an application or software system is working correctly.

These methods provide a mechanism that can help to ensure the completeness of tests and provide the highest likelihood for uncovering errors in software.

The mechanism for determining whether a software program or system has passed or failed such a test is known as a **test oracle**.

Test Plan

test plan is a document detailing a systematic approach to testing a system such as a machine or software.

The plan typically contains a detailed understanding of what the eventual workflow will be.

A test plan documents the strategy that will be used to verify and ensure that a product or system meets its design specifications and other requirements.

A test plan is usually prepared by or with significant input from Test Engineers.

Test Plan Continued…

A test plan will include the following.

- Introduction to the Test Plan document
- Assumptions when testing the application
- List of test cases included in Testing the application
- List of features to be tested
- What sort of Approach to use when testing the software
- List of Deliverables that need to be tested
- The resources allocated for testing the application
- Any Risks involved during the testing process
- A Schedule of tasks and milestones as testing is started

Types of Testing

Two types:

- *Manual Testing*: This type includes the testing of the Software manually i.e. without using any automated tool or any script. In this type the tester takes over the role of an end user and test the Software to identify any un-expected behavior or bug.
- *Automation Testing:* Automation testing which is also known as *Test Automation*, is when the tester writes scripts and uses another software to test the software. This process involves automation of a manual process. Automation Testing is used to re-run the test scenarios that were performed manually, quickly and repeatedly.

Testing Methods

Three types:

- *Black box testing:* The technique of testing without having any knowledge of the interior workings of the application is Black Box testing. The tester is oblivious to the system architecture and does not have access to the source code. Typically, when performing a black box test, a tester will interact with the system's user interface by providing inputs and examining outputs without knowing how and where the inputs are worked upon.
- *White box testing:* White box testing is the detailed investigation of internal logic and structure of the code. White box testing is also called glass testing or open box testing. In order to perform white box testing on an application, the tester needs to possess knowledge of the internal working of the code. The tester needs to have a look inside the source code and find out which unit/chunk of the code is behaving inappropriately.
- *Grey box testing:* Grey Box testing is a technique to test the application with limited knowledge of the internal workings of an application. In software testing, the term *the more you know the better* carries a lot of weight when testing an application.

Level of Testing

There are different levels during the process of Testing

Levels of testing include the different methodologies that can be used while conducting Software Testing.

Following are the main levels of Software Testing:

- Functional Testing.
- Non-Functional Testing.

Functional Testing

is based on the specifications of the software that is to be tested.

tested by providing input and then the results are examined that need to conform to the functionality it was intended for.

Functional Testing of the software is conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements.

Types:

- Unit Testing
- Integration Testing
- System Testing
- Acceptance Testing

Unit Testing

This type of testing is performed by the developers before the setup is handed over to the testing team to formally execute the test cases.

Unit testing is performed by the respective developers on the individual units of source code assigned areas.

The developers use test data that is separate from the test data of the quality assurance team.

The goal of unit testing is to isolate each part of the program and show that individual parts are correct in terms of requirements and functionality.

Integration Testing

The testing of combined parts of an application to determine if they function correctly together is Integration testing.

There are two methods of doing Integration

- *Testing Bottom-up Integration testing*: This testing begins with unit testing, followed by tests of progressively higher-level combinations of units called modules or builds.
- *Top Down Integration testing:* This testing, the highest-level modules are tested first and progressively lower-level modules are tested after that.

System Testing

This is the next level in the testing and tests the system as a whole.

Once all the components are integrated, the application as a whole is tested rigorously to see that it meets Quality Standards.

This type of testing is performed by a specialized testing team.

System Testing enables us to test, verify and validate both the business requirements as well as the Applications Architecture.

The application is tested in an environment which is very close to the production environment where the application will be deployed.

Acceptance Testing

This is arguably the most importance type of testing as it is conducted by the Quality Assurance Team who will gauge whether the application meets the intended specifications and satisfies the client's requirements.

The QA team will have a set of pre written scenarios and Test Cases that will be used to test the application.

Types:

- Alpha testing
- Beta testing

Alpha Testing

This test is the first stage of testing and will be performed amongst the teams (developer and QA teams).

Unit testing, integration testing and system testing when combined are known as alpha testing. During this phase, the following will be tested in the application:

- Spelling Mistakes
- Broken Links
- Cloudy Directions
- The Application will be tested on machines with the lowest specification to test loading times and any latency problems.

Beta Testing

This test is performed after Alpha testing has been successfully performed. In beta testing a sample of the intended audience tests the application.

Beta testing is also known as pre-release testing.

In this phase the audience will be testing the following:

- Users will install, run the application and send their feedback to the project team.
- Typographical errors, confusing application flow, and even crashes.
- Getting the feedback, the project team can fix the problems before releasing the software to the actual users.
- The more issues you fix that solve real user problems, the higher the quality of your application will be.
- Having a higher-quality application when you release to the general public will increase customer satisfaction.

Non functional Testing

is based upon the testing of the application from its non-functional attributes.

Non-functional testing of Software involves testing the Software from the requirements which are non functional in nature related but important a well such as performance, security, user interface etc.

commonly used non-functional testing types are

- Performance Testing
- Usability Testing
- Security Testing
- Portability Testing

Test Strategies

A strategy for software testing must accommodate low-level tests that are necessary to verify that a small source code segment has been correctly implemented as well as high-level tests that validate major system functions against customer requirements.

A strategy must provide guidance for the practitioner and a set of milestones for the manager.

Because the steps of the test strategy occur at a time when deadline pressure begins to rise, progress must be measurable and problems must surface as early as possible.

Strategic Approach to Testing

Testing begins at the component level and works outward toward the integration of the entire computer-based system.

Different testing techniques are appropriate at different points in time.

The developer of the software conducts testing and may be assisted by independent test groups for large projects.

The role of the independent tester is to remove the conflict of interest inherent when the builder is testing his or her own product.

Testing and debugging are different activities.

Debugging must be accommodated in any testing strategy.

Make a distinction between verification (are we building the product right?) and validation (are we building the right product?)

Verification and Validation

Verification: "Are we building the product right?"

Validation: "Are we building the right product?“

Verification refers to the set of activities that ensure that software correctly implements a specific function.

Validation refers to a different set of activities that ensure that the software that has been built is traceable to customer requirements.

V and V Process

Is a whole life-cycle process - V & V must be

applied at each stage in the software process.

Has two principal objectives

- The discovery of defects in a system;
- The assessment of whether or not the system is useful and useable in an operational situation.

V and V goals

Verification and validation should establish confidence that the software is fit for purpose.

This does NOT mean completely free of defects.

Rather, it must be good enough for its intended use and the type of use will determine the degree of confidence that is needed.

software quality assurance

The definition of V&V encompasses many of the activities that we have referred to as *software quality assurance* (SQA).

Verification and validation encompasses a wide array of SQA activities that include

- formal technical reviews,
- quality and configuration audits,
- performance monitoring,
- simulation,
- feasibility study,
- documentation review,
- database review,
- algorithm analysis,
- development testing,
- qualification testing,
- installation testing

References

- Software Engineering, By Sommerville, Pearson Education
- Software Engineering, By Pressman, MC Graw Hill
- http://www.tutorialspoint.com/software_testing/levels_of_testing.htm
- en.wikipedia.org/wiki/Test_plan
- http://www.tutorialspoint.com/software_testing/testing_methods.htm
- http://www.tutorialspoint.com/software_testing/testing_types.htm

Software Project Management Presentation on SEI, CMM and SQA, SQA Plan

SEI

- The Carnegie Mellon *Software Engineering Institute* (SEI) is a federally funded research and development center headquartered on the campus of Carnegie Mellon University in Pittsburgh, Pennsylvania, United States.
- works closely with industry and academia through research collaborations.
- principal areas: acquisition, process management, risk, security, software development, and system design.

Missions of SEI

- Research – advancing the science and practice
- Collaboration – bringing together and building on work found in industry, academia, and government
- Development and Demonstration – maturing promising technologies and practices and demonstrating their utility through trial application and prototypes
- Transition – propagating proven technologies and practices through publication, standards and other venues

CMM

- CMM stands for Capability Maturity Model.
- is a development model created after study of data collected from organizations that contracted with the U.S. Department of Defense.
- The model's aim is to improve existing software-development processes.

Structure of CMM

- The model involves five aspects:
- *Maturity Levels:* a 5-level process maturity continuum - where the uppermost (5th) level is a notional ideal state where processes would be systematically managed by a combination of process optimization and continuous process improvement.
- *Key Process Areas:* a Key Process Area identifies a cluster of related activities that, when performed together, achieve a set of goals considered important.
- *Goals:* the goals of a key process area summarize the states that must exist for that key process area to have been implemented in an effective and lasting way. The extent to which the goals have been accomplished is an indicator of how much capability the organization has established at that maturity level. The goals signify the scope, boundaries, and intent of each key process area.
- *Common Features:* common features include practices that implement and institutionalize a key process area. There are five types of common features: commitment to perform, ability to perform, activities performed, measurement and analysis, and verifying implementation.
- *Key Practices:* The key practices describe the elements of infrastructure and practice that contribute most effectively to the implementation and institutionalization of the area.

Levels of CMM

- **Level 1** - characterized by period and efforts required by individuals to successfully complete projects.
- **Level 2** – software project tracking, requirements management, realistic planning, and configuration management processes are in place; successful practices can be repeated.
- **Level 3** – standard software development and maintenance processes are integrated throughout an organization; a Software Engineering Process Group is in place to oversee software processes, and training programs are used to ensure understanding and compliance.
- **Level 4** – metrics are used to track productivity, processes, and products. Project performance is predictable, and quality is consistently high.
- **Level 5** – the focus is on continuous process improvement. The impact of new processes and technologies can be predicted and effectively implemented when required.

Software Quality

- According to Demin: "*The problem inherent in attempts to define the quality of a product, almost any product, were stated by the master Walter A. Shewhart. The difficulty in defining quality is to translate future needs of the user into measurable characteristics, so that a product can be designed and turned out to give satisfaction at a price that the user will pay. This is not easy, and as soon as one feels fairly successful in the endeavor, he finds that the needs of the consumer have changed, competitors have moved in, etc.*"
- reflects how well it complies with or conforms to a given design, based on functional requirements or specifications.
- refers to how it meets non-functional requirements that support the delivery of the functional requirements, such as robustness or maintainability, the degree to which the software was produced correctly.
- Software quality measurement quantifies to what extent a software or system rates along each of these five dimensions.

Software Quality Assurance

- the process of evaluating the quality of a product and enforcing adherence to software product standards and procedures.
- Software quality assurance (SQA) consists of a means of monitoring the software engineering processes and methods used to ensure quality.
- SQA encompasses the entire software development process, which includes processes such as requirements definition, software design, coding, source code control, code reviews, change management, configuration management, testing, release management, and product integration.
- SQA is organized into goals, commitments, abilities, activities, measurements, and verifications.

Software Quality Assurance Continued…

- Conformance to software requirements is the foundation from which software quality is measured.
- Specified standards are used to define the development criteria that are used to guide the manner in which software is engineered.
- Software must conform to implicit requirements (ease of use, maintainability, reliability, etc.) as well as its explicit requirements.

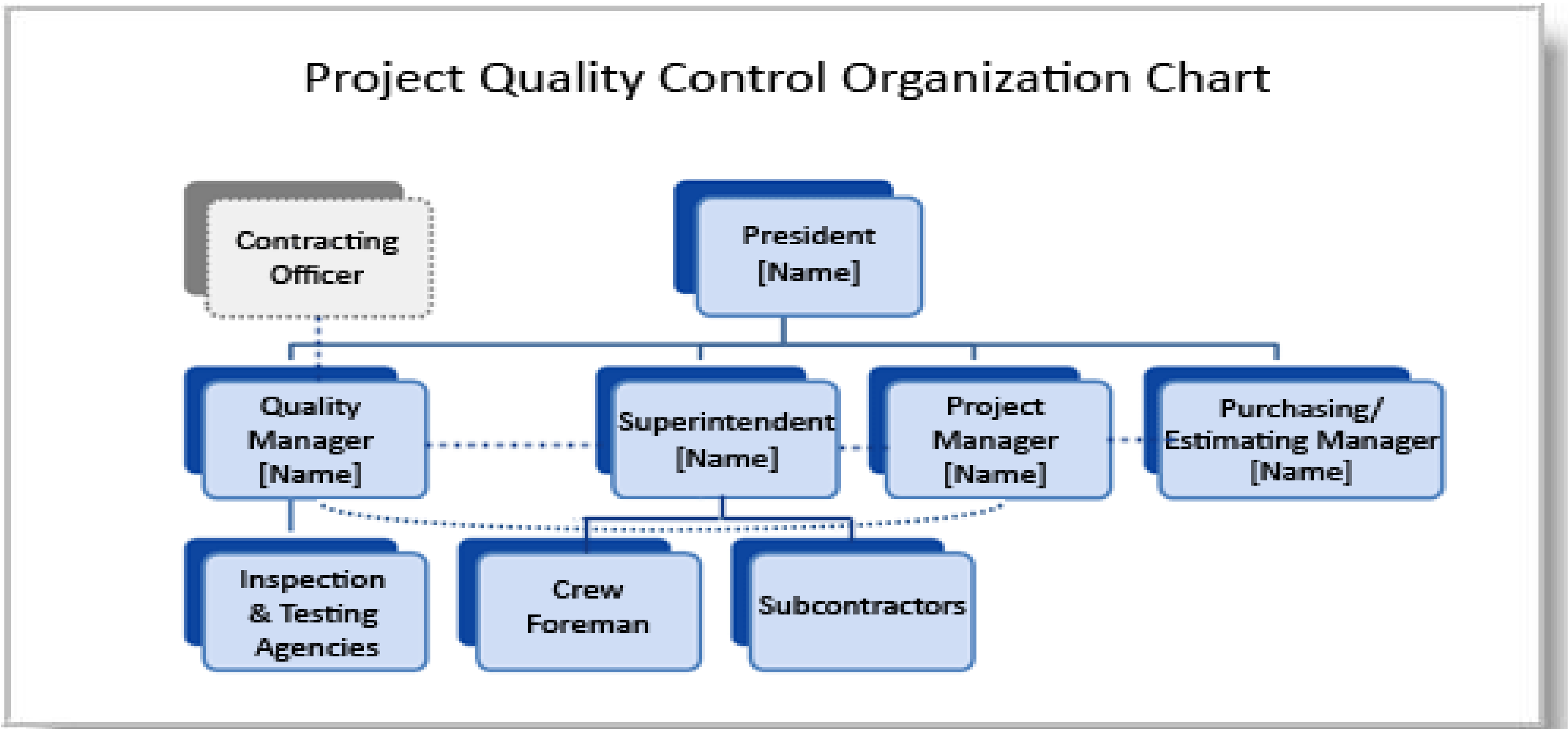
SQA Activities

- Formulating a quality management plan
- Applying software engineering techniques
- Conducting formal technical reviews
- Applying a multi-tiered testing strategy
- Enforcing process adherence
- Controlling change
- Measuring impact of change
- Performing SQA audits
- Keeping records and reporting

QA Organizational Structure

- The organizational structure has to provide the QA manager with direct organizational paths into every department.
- Small businesses can meet these requirements by assigning the QA responsibilities to someone in management, giving him the authority to manage QA matters throughout the company and creating a QA reporting path to the executive level.
- Employees continue to report to their department manager for disciplinary and non-QA matters, but report to the person responsible for QA on quality questions.

SQA Organizational Structure



SQA Plan

- Management section
  - describes the place of SQA in the structure of the organization
- Documentation section
  - describes each work product produced as part of the software process
- Standards, practices, and conventions section
  - lists all applicable standards/practices applied during the software process and any metrics to be collected as part of the software engineering work
- Reviews and audits section
  - provides an overview of the approach used in the reviews and audits to be conducted during the project

SQA Plan Continued…

- Problem reporting and corrective action section
  - defines procedures for reporting, tracking, and resolving errors or defects, identifies organizational responsibilities for these activities
- Other
  - tools, SQA methods, change control, record keeping, training, and risk management
- Test section
  - references the test plan and procedure document and defines test record keeping requirements

References

- http://www.sei.cmu.edu
- http://www.sei.cmu.edu/cmmi/
- http://www.zeepedia.com/read.php?software_quality_assurance_activities_software_project_management&b=18&c=19
- http://www.firsttimequality.com/Blog/?Tag=organization%20chart