Software Testing Vocabulary:

YouTube Channel Link: https://www.youtube.com/c/HaradhanAutomationLibrary?sub_confirmation=1

Affinity Diagram: A group process that takes large amounts of language data, such as a list developed by brainstorming, and divides it into categories.

Application: A single software product that may or may not fully support a business function.

Audit: This is an inspection/assessment activity that verifies compliance with plans, policies, and procedures, and ensures that resources are conserved. Audit is a staff function; it serves as the "eyes and ears" of management.

Backlog: Work waiting to be done; for IT this includes new systems to be developed and enhancements to existing systems. To be included in the development backlog, the work must have been cost-justified and approved for development.

Baseline: A quantitative measure of the current level of performance.

Benchmarking: Comparing your company's products, services, or processes against best practices, or competitive practices, to help define superior performance of a product, service, or support process.

Black-box Testing: A test technique that focuses on testing the functionality of the program, component, or application against its specifications without knowledge of how the system is constructed; usually data or business process driven.

Boundary Value Analysis: A data selection technique in which test data is chosen from the "boundaries" of the input or output domain classes, data structures, and procedure parameters. Choices often include the actual minimum and maximum boundary values, the maximum value plus or minus one, and the minimum value plus or minus one.

Brainstorming: A group process for generating creative and diverse ideas.

Branch Testing: A test method that requires that each possible branch on each decision point be executed at least once.

Bug: A general term for all software defects or errors.

Candidate: An individual who has met eligibility requirements for a credential awarded through a certification program, but who has not yet earned that certification through participation in the required skill and knowledge assessment instruments.

Cause-Effect Graphing: A tool used to derive test cases from specifications. A graph that relates causes (or input conditions) to effects is generated. The information in the graph is converted into a decision table where the columns are the cause-effect combinations. Unique rows represent test cases.

Certification: A voluntary process instituted by a nongovernmental agency by which individual applicants are recognized for having achieved a measurable level of skill or knowledge. Measurement of the skill or knowledge makes certification more restrictive than simple registration, but much less restrictive than formal licensure.

Checklists: A series of probing questions about the completeness and attributes of an application system. Well-constructed checklists cause evaluation of areas, which are prone to problems. It both limits the scope of the test and directs the tester to the areas in which there is a high probability of a problem.

Checkpoint Review: Held at predefined points in the development process to evaluate whether certain quality factors (critical success factors) are being adequately addressed in the system being built. Independent experts for the purpose of identifying problems conduct the reviews as early as possible.

Check sheet: A form used to record data as it is gathered.

Client: The customer that pays for the product received and receives the benefit from the use of the product.

Coaching: Providing advice and encouragement to an individual or individuals to promote a desired behaviour.

Code Comparison: One version of source or object code is compared to a second version. The objective is to identify those portions of computer programs that have been changed. The technique is used to identify those segments of an application program that have been altered as a result of a program change.

Compiler-Based Analysis: Most compilers for programming languages include diagnostics that identify potential program structure flaws. Many of these diagnostics are warning messages requiring the programmer to conduct additional investigation to determine whether or not the problem is real. Problems may include syntax problems, command violations, or variable/data reference problems. These diagnostic messages are a useful means of detecting program problems, and should be used by the programmer.

Complete Test Set: A test set containing data that causes each element of pre-specified set of Boolean conditions to be true. In addition, each element of the test set causes at least one condition to be true.

Completeness: The property that all necessary parts of an entity are included. Often, a product is said to be complete if it has met all requirements.

Complexity-Based Analysis: Based upon applying mathematical graph theory to programs and preliminary design language specification (PDLs) to determine a unit's complexity. This analysis can be used to measure and control complexity when maintainability is a desired attribute. It can also be used to estimate test effort required and identify paths that must be tested.

Compliance Checkers: A parse program looking for violations of company standards. Statements that contain violations are flagged. Company standards are rules that can be added, changed, and deleted as needed.

Condition Coverage: A white-box testing technique that measures the number of, or percentage of, decision outcomes covered by the test cases designed. 100% condition coverage would indicate that every possible outcome of each decision had been executed at least once during testing.

Configuration Management Tools: Tools that are used to keep track of changes made to systems and all related artifacts. These are also known as version control tools.

Configuration Testing: Testing of an application on all supported hardware and software platforms. This may include various combinations of hardware types, configuration settings, and software versions.

Consistency: The property of logical coherence among constituent parts. Consistency can also be expressed as adherence to a given set of rules.

Consistent Condition Set: A set of Boolean conditions such that complete test sets for the conditions uncover the same errors.

Control Flow Analysis: Based upon graphical representation of the program process. In control flow analysis, the program graph has nodes, which represent a statement or segment possibly ending in an unresolved branch. The graph illustrates the flow of program control from one segment to another as illustrated through branches. The objective of control flow analysis is to determine potential problems in logic branches that might result in a loop condition or improper processing.

Conversion Testing: Validates the effectiveness of data conversion processes, including field-to field mapping, and data translation.

Correctness: The extent to which software is free from design and coding defects (i.e., faultfree). It is also the extent to which software meets its specified requirements and user objectives.

Cost of Quality (COQ): Money spent beyond expected production costs (labor, materials, equipment) to ensure that the product the customer receives is a quality (defect free) product. The Cost of Quality includes prevention, appraisal, and correction or repair costs.

Coverage-Based Analysis: A metric used to show the logic covered during a test session, providing insight to the extent of testing. The simplest metric for coverage would be the number of computer statements executed during the test compared to the total number of statements in the program. To completely test the program structure, the test data chosen should cause the execution of all paths. Since this is not generally possible outside of unit test, general metrics have been developed which give a measure of the quality of test data based on the proximity to this ideal coverage. The metrics should take into consideration the existence of infeasible paths, which are those paths in the program that have been designed so that no data will cause the execution of those paths.

Customer: The individual or organization, internal or external to the producing organization that receives the product.

Data Dictionary: Provides the capability to create test data to test validation for the defined data elements. The test data generated is based upon the attributes defined for each data element. The test data will check both the normal variables for each data element as well as abnormal or error conditions for each data element.

DD (decision-to-decision) path: A path of logical code sequence that begins at a decision statement or an entry and ends at a decision statement or an exit.

Debugging: The process of analysing and correcting syntactic, logic, and other errors identified during testing.

Decision Coverage: A white-box testing technique that measures the number of, or percentage of, decision directions executed by the test case designed. 100% decision coverage would indicate that all decision directions had been executed at least once during testing. Alternatively, each logical path through the program can be tested. Often, paths through the program are grouped into a finite set of classes, and one path from each class is tested.

Decision Table: A tool for documenting the unique combinations of conditions and associated results in order to derive unique test cases for validation testing.

Defect: Operationally, it is useful to work with two definitions of a defect: 1. From the producer's viewpoint a defect is a product requirement that has not been met or a product attribute possessed by a product or a function performed by a product that is not in the statement of requirements that define the product; 2. From the customer's viewpoint a defect is anything that causes customer dissatisfaction, whether in the statement of requirements or not.

Defect Tracking Tools: Tools for documenting defects as they are found during testing and for tracking their status through to resolution.

Design Level: The design decomposition of the software item (e.g., system, subsystem, program, or module).

Desk Checking: The most traditional means for analysing a system or a program. Desk checking is conducted by the developer of a system or program. The process involves reviewing the complete product to ensure that it is structurally sound and that the standards and requirements have been met. This tool can also be used on artifacts created during analysis and design.

Driver: Code that sets up an environment and calls a module for test.

Dynamic Analysis: Analysis performed by executing the program code. Dynamic analysis executes or simulates a development phase product, and it detects errors by analysing the response of a product to sets of input data.

Dynamic Assertion: A dynamic analysis technique that inserts into the program code assertions about the relationship between program variables. The truth of the assertions is determined as the program executes.

Empowerment: Giving people the knowledge, skills, and authority to act within their area of expertise to do the work and improve the process.

Entrance Criteria: Required conditions and standards for work product quality that must be present or met for entry into the next stage of the software development process.

Equivalence Partitioning: The input domain of a system is partitioned into classes of representative values so that the number of test cases can be limited to one-per-class, which represents the minimum number of test cases that must be executed.

Error or Defect: 1. A discrepancy between a computed, observed, or measured value or condition and the true, specified, or theoretically correct value or condition. Human action that results in software containing a fault (e.g., omission or misinterpretation of user requirements in a software specification, incorrect translation, or omission of a requirement in the design specification).

Error Guessing: Test data selection technique for picking values that seem likely to cause defects. This technique is based upon the theory that test cases and test data can be developed based on the intuition and experience of the tester.

Exhaustive Testing: Executing the program through all possible combinations of values for program variables.

Exit Criteria: Standards for work product quality, which block the promotion of incomplete or defective work products to subsequent stages of the software development process.

File Comparison: Useful in identifying regression errors. A snapshot of the correct expected results must be saved so it can be used for later comparison.

Flowchart Pictorial: representations of data flow and computer logic. It is frequently easier to understand and assess the structure and logic of an application system by developing a flow chart than to attempt to understand narrative descriptions or verbal explanations. The flowcharts for systems are normally developed manually, while flowcharts of programs can be produced.

Force Field Analysis: A group technique used to identify both driving and restraining forces that influence a current situation.

Formal Analysis: Technique that uses rigorous mathematical techniques to analyse the algorithms of a solution for numerical properties, efficiency, and correctness.

Functional Testing: Application of test data derived from the specified functional requirements without regard to the final program structure.

Histogram: A graphical description of individually measured values in a data set that is organized according to the frequency or relative frequency of occurrence. A histogram illustrates the shape of the distribution of individual values in a data set along with information regarding the average and variation.

Infeasible Path: A sequence of program statements that can never be executed. Inputs Materials, services, or information needed from suppliers to make a process work, or build a product.

Inspection: A formal assessment of a work product conducted by one or more qualified independent reviewers to detect defects, violations of development standards, and other problems. Inspections involve authors only when specific questions concerning deliverables exist. An inspection identifies defects, but does not attempt to correct them. Authors take corrective actions and arrange follow-up reviews as needed.

Instrumentation: The insertion of additional code into a program to collect information about program behaviour during program execution.

Integration Testing: This test begins after two or more programs or application components have been successfully unit tested. It is conducted by the development team to validate the technical quality or design of the application. It is the first level of testing which formally integrates a set of programs that communicate among themselves via messages or files (a client and its server(s), a string of batch programs, or a set of online modules within a dialog or conversation.)

Invalid Input: Test data that lays outside the domain of the function the program represents.

Leadership: The ability to lead, including inspiring others in a shared vision of what can be, taking risks, serving as a role model, reinforcing and rewarding the accomplishments of others, and helping others to act.

Life Cycle Testing: The process of verifying the consistency, completeness, and correctness of software at each stage of the development life cycle.

Management: A team or individuals who manage(s) resources at any level of the organization.

Mapping: Provides a picture of the use of instructions during the execution of a program. Specifically, it provides a frequency listing of source code statements showing both the number of times an instruction was executed and which instructions were not executed. Mapping can be used to optimize source code by identifying the frequently used instructions. It can also be used to determine unused code, which can demonstrate code, which has not been tested, code that is infrequently used, or code that is non-entrant.

Mean: A value derived by adding several quantities and dividing the sum by the number of these quantities.

Metric-Based Test Data Generation: The process of generating test sets for structural testing based on use of complexity or coverage metrics.

Model Animation: It verifies that early models can handle the various types of events found in production data. This is verified by "running" actual production transactions through the models as if they were operational systems.

Model Balancing: Model balancing relies on the complementary relationships between the various models used in structured analysis (event, process, data) to ensure that modelling rules/standards have been followed; this ensures that these complementary views are consistent and complete.

Mutation Analysis: A method to determine test set thoroughness by measuring the extent to which a test set can discriminate the program from slight variants (i.e., mutants) of it.

Network Analysers: A tool used to assist in detecting and diagnosing network problems. Outputs Products, services, or information supplied to meet customer needs. Pass/Fail Criteria Decision rules used to determine whether a software item or feature passes or fails a test.

Path Expressions: A sequence of edges from the program graph that represents a path through the program.

Path Testing: A test method satisfying the coverage criteria that each logical path through the program be tested. Often, paths through the program are grouped into a finite set of classes and one path from each class is tested.

Performance Test: Validates that both the online response time and batch run times meet the defined performance requirements.

Performance/Timing Analyzer: A tool to measure system performance. Phase (or Stage) Containment A method of control put in place within each stage of the development process to promote error identification and resolution so that defects are not propagated downstream to subsequent stages of the development process. The verification, validation, and testing of work within the stage that it is created. Policy Managerial desires and intents concerning either process (intended objectives) or products (desired attributes).

Population Analysis: Analyses production data to identify, independent from the specifications, the types and frequency of data that the system will have to process/produce. This verifies that the specs can handle types and frequency of actual data and can be used to create validation tests.

Procedure: The step-by-step method followed to ensure that standards are met. The work effort that produces a product. This includes efforts of people and equipment guided by policies, standards, and procedures. 2. The process or set of processes used by an organization or project to plan, manage, execute, monitor, control, and improve its software related activities. A set of activities and tasks. A statement of purpose and an essential set of practices (activities) that address that purpose.

Process Improvement: To change a process to make the process produce a given product faster, more economically, or of higher quality. Such changes may require the product to be changed. The defect rate must be maintained or reduced.

Product: The output of a process: the work product. There are three useful classes of products: Manufactured Products (standard and custom), Administrative/Information Products (invoices, letters, etc.), and Service Products (physical, intellectual, physiological, and psychological). A statement of requirements defines products; one or more people working in a process produce them.

Product Improvement: To change the statement of requirements that defines a product to make the product more satisfying and attractive to the customer (more competitive). Such changes may add to or delete from the list of attributes and/or the list of functions defining a product. Such changes frequently require the process to be changed. Note: This process could result in a very new product.

Production Costs: The cost of producing a product. Production costs, as currently reported, consist of (at least) two parts; actual production or right-the-first time costs (RFT) plus the Cost of Quality (COQ). RFT costs include labor, materials, and equipment needed to provide the product correctly the first time.

Productivity: The ratio of the output of a process to the input, usually measured in the same units. It is frequently useful to compare the value added to a product by a process, to the value of the input resources required (using fair market values for both input and output).

Proof of Correctness: The use of mathematical logic techniques to show that a relationship between program variables assumed true at program entry implies that another relationship between program variables holds at program exit.

Quality: A product is a quality product if it is defect free. To the producer, a product is a quality product if it meets or conforms to the statement of requirements that defines the product. This statement is usually shortened to: quality means meets requirements. From a customer's perspective, quality means "fit for use."

Quality Assurance (QA): The set of support activities (including facilitation, training, measurement, and analysis) needed to provide adequate confidence that processes are established and continuously improved to produce products that meet specifications and are fit for use.

Quality Control (QC): The process by which product quality is compared with applicable standards, and the action taken when nonconformance is detected. Its focus is defect detection and removal. This is a line function; that is, the performance of these tasks is the responsibility of the people working within the process.

Quality Function Deployment (QFD): A systematic matrix method used to translate customer wants or needs into product or service characteristics that will have a significant positive impact on meeting customer demands.

Quality Improvement: To change a production process so that the rate at which defective products (defects) are produced is reduced. Some process changes may require the product to be changed.

Recovery Test: Evaluates the contingency features built into the application for handling interruptions and for returning to specific points in the application processing cycle, including checkpoints, backups, restores, and restarts. This test also assures that disaster recovery is possible.

Regression Testing: Testing of a previously verified program or application following program modification for extension or correction to ensure no new defects have been introduced.

Requirement: A formal statement of: 1. An attribute to be possessed by the product or a function to be performed by the product 2. The performance standard for the attribute or function; and/or 3. The measuring process to be used in verifying that the standard has been met.

Risk Matrix: Shows the controls within application systems used to reduce the identified risk, and in what segment of the application those risks exist. One dimension of the matrix is the risk, the second dimension is the segment of the application system, and within the matrix at the intersections are the controls. For example, if a risk is “incorrect input” and the systems segment is “data entry,” then the intersection within the matrix would show the controls designed to reduce the risk of incorrect input during the data entry segment of the application system.

Run Chart: A graph of data points in chronological order used to illustrate trends or cycles of the characteristic being measured to suggest an assignable cause rather than random variation.

Self-validating Code: Code that makes an explicit attempt to determine its own correctness and to proceed accordingly. Simulation Use of an executable model to represent the behaviour of an object. During testing, the computational hardware, the external environment, and even code segments may be simulated.

Software Feature: A distinguishing characteristic of a software item (e.g., performance, portability, or functionality). Software Item Source code, object code, job control code, control data, or a collection of these.

Special Test Data: Test data based on input values that are likely to require special handling by the program. Standardize Procedures that are implemented to ensure that the output of a process is maintained at a desired level.

Standards: The measure used to evaluate products and identify nonconformance. The basis upon which adherence to policies is measured.

Statement of Requirements: The exhaustive list of requirements that define a product. Note that the statement of requirements should document requirements proposed and rejected (including the reason for the rejection) during the requirement determination process.

Statement Testing: A test method that executes each statement in a program at least once during program testing. Static

Analysis: Analysis of a program that is performed without executing the program. It may be applied to the requirements, design, or code. Statistical Process Control The use of statistical techniques and tools to measure an ongoing process for change or stability.

Stress Testing: This test subjects a system, or components of a system, to varying environmental conditions that defy normal expectations. For example, high transaction volume, large database size or restart/recovery circumstances. The intention of stress testing is to identify constraints and to ensure that there are no performance problems.

Structural Testing: A testing method in which the test data is derived solely from the program structure. Stub Special code segments that when invoked by a code segment under testing, simulate the behaviour of designed and specified modules not yet constructed.

Supplier: An individual or organization that supplies inputs needed to generate a product, service, or information to a customer.

Symbolic Execution: A method of symbolically defining data that forces program paths to be executed. Instead of executing the program with actual data values, the variable names that hold the input values are used. Thus, all variable manipulations and decisions are made symbolically. This process is used to verify the completeness of the structure, as opposed to assessing the functional requirements of the program.

System: One or more software applications that together support a business function.

System Test: During this event, the entire system is tested to verify that all functional, information, structural and quality requirements have been met. A predetermined combination of tests is designed that, when executed successfully, satisfy management that the system meets specifications. System testing verifies the functional quality of the system in addition to all external interfaces, manual procedures, restart and recovery, and human-computer interfaces. It also verifies that interfaces between the application and the open environment work correctly, that JCL functions correctly, and that the application functions appropriately with the Database Management System, Operations environment, and any communications systems.

Test: 1. A set of one or more test cases. 2. A set of one or more test cases and procedures.

Test Case Generator: A software tool that creates test cases from requirements specifications. Cases generated this way ensure that 100% of the functionality specified is tested.

Test Case Specification: An individual test condition, executed as part of a larger test that contributes to the test's objectives. Test cases document the input, expected results, and execution conditions of a given test item. Test cases are broken down into one or more detailed test scripts and test data conditions for execution.

Test Cycle: Test cases are grouped into manageable (and schedulable) units called test cycles. Grouping is according to the relation of objectives to one another, timing requirements, and on the best way to expedite defect detection during the testing event. Often test cycles are linked with execution of a batch process.

Test Data Generator: A software package that creates test transactions for testing application systems and programs. The type of transactions that can be generated is dependent upon the options available in the test data generator. With many current generators, the prime advantage is the ability to create a large number of transactions to volume test application systems.

Test Data Set: Set of input elements used in the testing process.

Test Design Specification: A document that specifies the details of the test approach for a software feature or a combination of features and identifies the associated tests.

Test Driver: A program that directs the execution of another program against a collection of test data sets. Usually, the test driver also records and organizes the output generated as the tests are run. Test Harness A collection of test drivers and test stubs.

Test Incident Report: A document describing any event during the testing process that requires investigation.

Test Item: A software item that is an object of testing. Test Item Transmittal Report A document that identifies test items and includes status and location information.

Test Log: A chronological record of relevant details about the execution of tests.

Test Plan: A document describing the intended scope, approach, resources, and schedule of testing activities. It identifies test items, the features to be tested, the testing tasks, the personnel performing each task, and any risks requiring contingency planning.

Test Procedure: Specification A document specifying a sequence of actions for the execution of a test.

Test Scripts: A tool that specifies an order of actions that should be performed during a test session. The script also contains expected results. Test scripts may be manually prepared using paper forms, or may be automated using capture/playback tools or other kinds of automated scripting tools.

Test Stubs: Simulates a called routine so that the calling routine's functions can be tested. A test harness (or driver) simulates a calling component or external environment, providing input to the called routine, initiating the routine, and evaluating or displaying output returned.

Test Suite Manager: A tool that allows testers to organize test scripts by function or other grouping.

Test Summary Report: A document that describes testing activities and results and evaluates the corresponding test items.

Tracing: A process that follows the flow of computer logic at execution time. Tracing demonstrates the sequence of instructions or a path followed in accomplishing a given task. The two main types of trace are tracing instructions in computer programs as they are executed, or tracing the path through a database to locate predetermined pieces of information.

Unit Test: Testing individual programs, modules, or components to demonstrate that the work package executes per specification, and validate the design and technical quality of the application. The focus is on ensuring that the detailed logic within the component is accurate and reliable according to pre-determined specifications. Testing stubs or drivers may be used to simulate behaviour of interfacing modules.

Usability Test: The purpose of this event is to review the application user interface and other human factors of the application with the people who will be using the application. This is to ensure that the design (layout and sequence, etc.) enables the business functions to be executed as easily and intuitively as possible. This review includes assuring that the user interface adheres to documented User Interface standards, and should be conducted early in the design stage of development. Ideally, an application prototype is used to walk the client group through various business scenarios, although paper copies of screens, windows, menus, and reports can be used.

User: The customer that actually uses the product received.

User Acceptance Test: User Acceptance Testing (UAT) is conducted to ensure that the system meets the needs of the organization and the end user/customer. It validates that the system will work as intended by the user in the real world, and is based on real world business scenarios, not system requirements. Essentially, this test validates that the right system was built.

Valid Input: Test data that lie within the domain of the function represented by the program.

Validation: Determination of the correctness of the final program or software produced from a development project with respect to the user needs and requirements. Validation is usually accomplished by verifying each stage of the software development life cycle.

Values (Sociology): The ideals, customs, instructions, etc., of a society toward which the people have an affective regard. These values may be positive, as cleanliness, freedom, or education, or negative, as cruelty, crime, or blasphemy. Any object or quality desired as a means or as an end in itself.

Verification: 1. The process of determining whether the products of a given phase of the software development cycle fulfil the requirements established during the previous phase. 2. The act of reviewing, inspecting, testing, checking, auditing, or otherwise establishing and documenting whether items, processes, services, or documents conform to specified requirements.

Vision: A vision is a statement that describes the desired future state of a unit.

Walkthroughs: During a walkthrough, the producer of a product “walks through” or paraphrases the products content, while a team of other individuals follow along. The team’s job is to ask questions and raise issues about the product that may lead to defect identification.

White-box Testing: A testing technique that assumes that the path of the logic in a program unit or component is known. White-box testing usually consists of testing paths, branch by branch, to produce predictable results. This technique is usually used during tests executed by the development team, such as Unit or Component testing,