

Crib Sheet of Main Terms/Concepts Used

Foundation-Level Software Testing Course

| Term | Description |
|---|---|
| Failure | A deviation of the software from its expected delivery or service A failure occurs when software does the 'wrong' thing. |
| Fault | A manifestation of human error in software, also known as a defect or bug. Faults may be caused by requirements, design or coding errors. |
| Error | A human action producing an incorrect result. Human error is inevitable in a complex activity. |
| Coverage | What we use to quantify testing. Defines an objective target for the amount of testing to perform. Measures completeness or thoroughness. |
| Acceptance Criteria (exit, closure or completion criteria) | Trigger to say: 'we've done enough '. Objective, non-technical for managers. E.g. all tests executed without failure, all faults corrected and re-tested, all outstanding incidents waived. |
| Expected outcome, expected result | The behaviour predicted by the specification of a component or system under test when a set of inputs are defined. |
| Baseline | Requirements, specs. etc. are baselines. They tell us what the software is meant to do. |
| Goal: locate faults | A successful test is one that locates a fault. Finding faults is good, perceived as improving quality; testers create 'tough' tests, not easy ones. |
| Re-testing | If a test finds a fault that we get fixed, we repeat the test that found the fault to ensure the fault has been corrected. This is a re-test. |
| Regression Testing | When we receive software that has had faults fixed, we may execute tests of unchanged software to make sure it has not been adversely affected by the change. This is regression testing. |
| Validation | Determination of the correctness of the products of software development with respect to the user needs and requirements. "Did we build the right system?" |
| Verification | The process of evaluating a system or component to determine whether the products of the given development phase satisfy the conditions imposed at the start of that phase. "Did we build the system right?" |
| Static testing | Reviews, walkthroughs, inspections of (primarily) documentation, including requirements, designs, code and test plans. |
| Dynamic testing | Test which involve executing the software. Component, link, system and acceptance tests are all dynamic tests. |
| Front-loading | The principle of starting test activities early. Front-loading includes static tests of documents and early test case preparation which 'tests ' the document on which the cases are based. Requirements, specification and design faults are detected earlier and are much less costly to fix. |
| Testability | The ease by which testers can plan and specify tests from a baseline document. Normally refers to requirements documents. |
| V-Model | A model for development and testing that promotes a staged approach to testing where the baselines are the products of earlier development stages. |
| Component (unit, module, program) testing | To demonstrate that a single program, module or component performs as described in its specification. Black and white box. Usually done by the component's author. |

Crib Sheet of Main Terms/Concepts Used

Foundation-Level Software Testing Course

| Term | Description |
|--|---|
| | |
| Link testing (integration testing in the small) | To demonstrate that a collection of components interface together as described in the physical design. White box tests of a sub-system or small group of components sharing an interface. Done by a member of the programming team. |
| Functional System Testing | To demonstrate that a sub-system or complete system performs as described in the design and requirements documentation. Testing is black box, mainly. Normally performed by a test team or group of independent testers. |
| Non-Functional System Testing | To demonstrate that the non-functional requirements (e.g. performance, volume, usability, security) are met. Normally performed after functional testing is complete. |
| Large scale integration testing | To demonstrate that a new or changed system interfaces correctly with other systems. Black and white box techniques may be used. Inter-project testers test multiple systems as an integrated whole. |
| User acceptance testing | To satisfy the users that the delivered system meets their requirements. Functional tests only. Users supported by test analysts test an entire system. |
| Integration strategies | Bottom up: low-level components are assembled into sub-systems and then complete systems. Top down: top-level components (e.g. menu structures) are assembled first, then lower level until eventually completing the whole system. Big bang: the entire system is assembled from its components in one go. |
| Driver | Code written to execute a component in isolation when its calling program doesn't yet exist. |
| Stub | Code written to substitute for features not yet written, so calling programs can be tested. |
| Alpha testing | Tests done by a supplier on early versions of a product. |
| Beta testing | Tests done by volunteer users on early versions of a product. |
| Performance testing | Tests to demonstrate that response time and throughput requirements can be met. |
| Stress testing | Tests to explore the behaviour of the system under extreme loads. |
| Security testing | Tests to demonstrate (CIA) Confidentiality, Integrity, Availability requirements are met. |
| Usability testing | Tests to ensure users can operate the system effectively and efficiently without being frustrated. |
| Storage testing | Test to ensure the system can accommodate planned amounts of data on disk or in memory. |
| Volume testing | Tests to ensure large tasks such as a large, end of month jobs (or smallest tasks such as empty jobs) can be run. |
| Installation testing | Tests to ensure the system can be installed and used correctly (and re-installed, de-installed). |
| Documentation testing | Tests to demonstrate that user manuals and guides are accurate, complete, consistent and helpful. |

Crib Sheet of Main Terms/Concepts Used

Foundation-Level Software Testing Course

| Term | Description |
|--|--|
| Backup and recovery testing | Tests to demonstrate that backup and recovery procedures are effective. |
| Maintenance Testing | Testing of changes and enhancements to existing systems. Dominated by regression testing. |
| Black-box tests (Functional) | Tests derived from the baseline document, without regard to the structure of the system or other implementation details. System/Acceptance tests are mainly black box. |
| White-box tests (Glass-box or structural) | Tests derived from the internal structure (i.e. code) of the system. Component testing is mainly White box. |
| Test Design (Technique) | The process of selecting test cases. A method used to derive or select test cases. |
| Test Measurement (technique) | A method used to measure test coverage. A technique identifies test coverage items that can be counted during test design and execution. |
| Equivalence partitioning | Where the inputs (or outputs) of a system are categorised using rules, and there is one test case per rule. |
| Boundary value analysis | Where values on and just outside the boundaries of equivalence partitions are used as test cases. |
| Statement testing/coverage | Tests are prepared and run to ensure that each statement of code exercised. |
| Decision or branch testing (and coverage) | Tests are prepared and run to ensure that each decision outcome or branch in code is exercised. |
| Error-guessing | Tests derived from knowledge and experience of previous problems. Used as a 'mopping up' approach. |
| Review | A method of examining documentation or code with a view to detecting, recording and tracking faults, communicating information and improving the development process. |
| Walkthrough | Aimed at communicating information, a document author leads reviewers through a document to explain the content. |
| Technical (Peer) review | Aimed at finding faults and achieving consensus on the way forward. Led by a reviewer and involving technical peers. |
| Inspection | Aimed at finding faults based on pre-defined rules, led by a trained moderator and involving technical peers. |
| Static analysis | Analysis of code to detect faults, without actually executing the software – normally done using a tool. |
| Configuration management | A disciplined approach to controlling the creation, organisation, versioning and build of software components. |
| Incident (observation) report | An unplanned event that (normally) documents a test failure or has an adverse effect on the testing. |
| Requirements Testing tool | Tool for capturing requirement into a repository, using e.g. UML. Provide limited consistency checking of requirements. |
| Static analysis tool | Used to analyse code and detect poor programming practices or over complex code. |
| Capture replay tool (test execution tool) | Used to execute prepared tests automatically. Normally have a programmable scripting language. |
| Dynamic analysis tool | Used to report on the state of the internal variables in code used while the test is still running. |
| Test Design tool | Used to capture and manage test cases and conditions and produce test documentation. |

Crib Sheet of Main Terms/Concepts Used

Foundation-Level Software Testing Course

| Term | Description |
|---|--|
| Data Preparation tool | Used to select data from production environments or create, edit and manipulate test data. |
| Source code coverage analysis tool | Used to report the statements or decisions covered by a test to help component testers to derive additional tests to meet a coverage target. |
| ISO 9000 | A standard for quality in software testing. Says that testing should be done. |
| DO178b | A standard for airborne software. Says how much software testing should be done. |
| IEEE 829 | The Standard for Software Test Documentation. |
| BS7925-1 | The Standard Glossary of Testing Terms. |
| BS7925-2 | The Standard for Software Component Testing. |
| IEEE 12207 | A standard for software engineering lifecycles. A framework to fit software development methodologies into. |