Unit-6
Tool Support For SW Testing

Topics to be covered:

- Types of test tool
  - Test tool classification
  - Tool support for management of testing
  - Tool support for static testing
  - Tool support for test specification
  - Tool support for test execution and logging
  - Tool support for performance and monitoring

Topics to be covered:

- Effective use of tools: Benefits & Risks
  - Potential benefits of using tools
  - Risks of using tools

- Introducing a tool into an organization

Explain Tool support for management of testing and tests?

Test management tool

- a) A tool that provides support to the test management and control part of a test process. It often has several capabilities, such as test ware management, scheduling of tests, logging of results, progress tracking, incident management and test reporting.
- b) The features provided by test management tools include those listed below:
  - Management of tests (keeping track of associated data for given set of tests, number of tests planned, written, run, passed or failed)
  - setting up of tests to be executed (manually or by a test execution tool)
  - Management of testing activities

- Interfaces to other tools, such as:
  - Test execution tools (test running tools)
  - Incident management tools
  - Requirement management tools
  - Configuration management tools
- Traceability of tests, test results and defects to requirements or other sources;
- Logging test results (note that the test management tool does not run tests, but could summarize results from test execution tools that the test management tool interfaces with):
- preparing progress reports based on metrics (quantitative analysis), such as:
  - Tests run and tests passed:
  - Incidents raised, defects fixed and outstanding.

Requirements management tool:

- A tool that supports the recording of requirements, requirements attributes (e.g. priority, person responsible) and annotation and facilitates traceability through layers of requirements and requirements change management. Some requirements management tools also provide facilities for static analysis, such as consistency checking and violations to pre-defined requirements rules.
- Features or characteristics of requirements management tools include support for:
  - Storing requirement statements:
  - Storing information about requirement attributes
  - Checking consistency of requirements
  - Identifying undefined, missing or ‘to be defined later’ requirements

- Prioritizing requirements for testing purposes;
- Traceability of requirements to tests and tests to requirements, functions or features
- Traceability through levels of requirements:
- Interfacing to test management tools:
- Coverage of requirements by a set of tests (sometimes)

Incident management tool (defect management tool):

- A tool that facilitates the recording and status tracking of incidents. They often have workflow- oriented facilities to track and control the allocation, correction and re-testing of incidents (confirmation testing) and provide reporting facilities.

- Features or characteristics of incident management tools include support for:
  - storing information about attributes of incident (e.g.severity)
  - storing attachments (e.g. a screen shot)
  - Prioritizing incidents
  - Assigning actions to people (fix, confirmation test, etc.)
  - Status (e.g. open. rejected. duplicate, deferred, ready for confirmation test, closed)
  - Reporting of statistics metrics about incidents (e.g. average time open, number of incidents with each status, total number raised, open or closed).
- Incident management tool functionality may he included in commercial test management tools.

Configuration management tool :

- A tool that provides support for the identification and control of configuration items, their status over changes and versions, and the release of baselines consisting of configuration items.
- Features or characteristics of configuration management tools include support for:
  - Storing information about versions & builds of the software & test ware;
  - Traceability between software and test ware and different versions or variants:
  - Keeping track of which versions belong with which configurations (e.g. operating systems, libraries, browsers)
  - Build and release management
  - Base lining (e.g. all the configuration items that make up a specific release)
  - Access control (checking in and out).

Explain Tool support for static testing?

<u>Review process support tools:</u>

- <u>Review tool:</u>
  - A tool that provides support to the review process. Typical features include review planning and tracking support, communication support, collaborative reviews and a repository for collecting and reporting of metrics.
  - Features or characteristics of review process support tools include support for:
    - A common reference for the review process or processes to use in different situations
    - Storing and sorting review comments
    - Communicating comments to relevant people
    - Co-ordinating online reviews

- Keeping track of comments, including defects found, and providing statistical information about them
- Providing traceability between comments, documents reviewed and related documents
- A repository for rules, procedures and checklists to be used in reviews, as well as entry and exit criteria
- Monitoring the review status (passed. passed with corrections, requires re-review)
- Collecting metrics and reporting on key factors.

– Static analysis tool (static analyzer):
  - A tool that carries out static analysis. Static analysis is defined as Analysis of software artifacts, e.g. requirements or code, carried out without execution of those software artifacts.

- Features or characteristics of static analysis tools include support to:
- Calculate metrics such as cyclomatic complexity or nesting levels (which can help to identify where more testing may be needed due to increased risk)
- Enforce coding standards
- Analyze structures and dependencies
- Aid in code understanding
- Identify anomalies or defects in the code.

– Modeling tool :
  - A tool that supports the validation of models of the software or system.

- Features or characteristics of modeling tools include support for:
- Identifying inconsistencies and defects within the model
- Helping to identify and prioritize areas of the model for testing
- Predicting system response and behavior under various situations, such as level of load
- Helping to understand system functions and identify test conditions using a modeling language such as UML.

Explain Tool support for test specification?

Test design tool:

- A tool that supports the test design activity by generating test inputs from a specification that maybe held in a Computer Aided Software Engineering (CASE) tool repository, e.g. requirements management tool, or from specified test conditions held in the tool itself or from code.
- Features/characteristics of test design tools include support for:
  - Generating test input values from:
  - Requirements.
  - Design models (state, data or object)
  - code
  - Graphical user interfaces
  - Test conditions
- Generating expected results, if an oracle is available to the tool.

Test data preparation tool:

- A type of test tool that enables data to be selected from existing databases or created, generated, manipulated and edited for use in testing.
- Features or characteristics of test data preparation tools include support to:
  - Extract selected data records from files or databases;
  - Modify data records to make them anonymous or not able to be identified with real people (for data protection)
  - Enable records to be sorted or arranged in a different order:
  - Generate new records populated with pseudo-random data, or data set up according to some guidelines, e.g. an operational profile:
  - Construct a large number of similar records from a template, to give a large set of records for volume tests.

Explain Tool support for test execution and logging?

<u>Test execution tools:</u>

- A type of test tool that is able to execute other software using an automated test script, e.g.capture/playback. A capture/playback is a type of test execution tool where inputs are recorded during manual testing in order to generate automated test scripts that can be executed later (i.e. replayed). These tools are used for automated regression testing.
- Features or characteristics of test execution tools include support for:
  – Capturing (recording) test inputs while tests are executed manually
  – Storing an expected result in the form of a screen or object to compare to, the next time the test is run
  – Executing tests from stored scripts and optionally data files accessed by the script (if data-driven or keyword-driven scripting is used)

- Dynamic comparison (while the test is running) of screens, elements, links, controls, objects and values
- Ability to initiate post-execution comparison
- Logging results of tests run (pass/fail, differences between expected and actual results)
- Masking or filtering of subsets of actual and expected results, for example excluding the screen-displayed current date and time which is not of interest to a particular test
- Measuring timings for tests
- Synchronizing inputs with the application under test, e.g. wait until the application is ready to accept the next input, or insert a fixed delay to represent human interaction speed
- Sending summary results to a test management tool.

Test harness :

- A test environment comprised of stubs and drivers needed to execute a test.
- Unit test framework tool :

  A tool that provides an environment for unit or component testing in which a component can be tested in isolation or with suitable stubs and drivers. It also provides other support for the developer, such as debugging capacities.
- Stub :

  A skeletal or special-purpose implementation of a software component, used to develop and test a component that calls or is otherwise dependent on it. It replaces a called component.
- Driver:

  A software component or test tool that replaces a component that takes care of the control and/or the calling of a component or system.

- Features or characteristics of test harnesses and unit test framework tools include support for:
  - Supplying inputs to the software being tested;
  - Receiving outputs generated by the software being tested;
  - Executing a set of tests within the framework or using the test harness:
  - Recording the pass/fail results of each test (framework tools):
  - Storing tests (framework tools);
  - Support for debugging (framework tools):
  - Coverage measurement at code level (framework tools).

- Test comparator:
- A test tool to perform automated test comparison.
- Test comparison:

  The process of identifying differences between the actual results produced by the component or system under test and the expected results for a test. Test comparison can be performed during test execution (dynamic comparison) or after test execution
- Features/characteristics of test comparators include support for:
  - Dynamic comparison of transient events that occur during test execution
  - Post-execution comparison of stored data, e.g. in files or databases
  - Masking or filtering of subsets of actual and expected results

- <u>Coverage tool :</u>
- A tool that provides objective measures of what structural elements, e.g. statements, decisions, or branches have been exercised by a test suite.
- Features or characteristics of coverage measurement tools include support for:
  - Identifying coverage items (instrumenting the code)
  - Calculating the percentage of coverage items that were exercised b a suite of tests
  - Reporting coverage items that have not been exercised as yet
  - Identifying test inputs to exercise as et uncovered items (test design tool functionality)
  - Generating stubs and drivers (if part of a unit test framework).

- Security tool :
- A tool that supports operational security or provides support for testing security vulnerabilities
- Security testing :

  Testing to determine the security of the software product.
- Security testing tool:

  A tool that provides support for testing security characteristics and vulnerabilities.

  Features/characteristics of security tools include support for:
- Identifying viruses;
- Detecting intrusions such as denial of service attacks;
- Simulating various types of external attacks;
- Probing for open ports or other externally visible points of attack:
- Identifying weaknesses in password files and passwords:
- Security checks during operation, e.g. for checking integrity of files & intrusion detection. e.g. checking results of test attacks.

Explain Tool support for performance and monitoring?

Dynamic analysis tool:

- A tool that provides run- time information on the state of the software code. These tools are most commonly used to identify unassigned pointers, check pointer arithmetic, to monitor the allocation, use and de-allocation of memory and to flag memory leaks.
- Features or characteristics of dynamic analysis tools include support for:
  - Detecting memory leaks:
  - Identifying pointer arithmetic errors such as null pointers:
  - Identifying time dependencies.

<u>Performance-testing tool (load-testing tool):</u>

- A tool to support performance testing and that usually has two main facilities:
  - load generation
  - test transaction measurement.
- Load generation can simulate either multiple users or high volumes of input data. During execution, response time measurements are taken from selected transactions and these are logged. Performance-testing tools normally provide reports based on test logs and graphs of load against response times.
- <u>Load testing</u>

  A test type concerned with measuring the behavior of a component or system with increasing load, e g. the number of parallel users and/or numbers of transactions, to determine what load can be handled by the component or system.

- <u>Volume testing</u>

  Testing where the system s subjected to large volumes of data.

- <u>Stress testing</u>

  Testing conducted to evaluate a system or component at or beyond the limits of its specified requirements.

- Features or characteristics of performance-testing tools include support for:
  - Generating a load on the system to he tested:
  - Measuring the timing of specific transactions as the load on the system varies:
  - Measuring average response times:
  - Producing graphs or charts of responses over time.

Monitor (monitoring tool)

- A software tool or hardware device that runs concurrently with the component or system under test and supervises records and/or analyzes the behavior of the component or system.
- Features/characteristics of monitoring tools include support for:
  - Identifying problems and sending an alert message to the administrator (e.g. network administrator)
  - Logging ideal-time and historical information
  - Finding optimal settings
  - Monitoring the number of users on a network
  - Monitoring network traffic (either in real time or covering a given length of time of operation with the analysis performed afterwards).

End of Unit-6