ISTQB – Foundation Level

CHAPTER 6: Tool Support For Testing

Prepared by: Vu Nguyen
Feb 2010
Updated by: Minh Ha
Dec 2012

AGENDA

- 6.1 Types of Test Tools
- 6.2 Effective Use of tools
- 6.3 Selection of Test Tools
- 6.4 Introduction a Tool Into an Organization

Types of Test Tools

- Purpose of tool support for testing:
  - Improving the efficiency of test work
  - Making it possible to execute testing tasks that are impossible manually. For example, performance and load testing
  - Improving the reliability of testing by automating unreliable manual tasks
- There are tools for a single or multiple purposes
- An important area for tools is monitoring anything interesting to the tester, like memory use, network traffic, etc.
- CAST tools: tools exists for supporting or automating test activities (Computer Aided Software Testing)

Test Tool Classification

- Test tools are grouped by the testing activities or areas that supported by a set of tools.
- E.g
  - Tools that support management activities,
  - Tools to support static testing.
  - Tools to support configuration management of testware
  - Tools to support incident management,
  - Tool to support requirements management and traceability
  - Tool t support coverage measurement and test design support.
- Probe effect: the effect on a component or system while it is being measured/analyzed

Tools for Test Management and Control

- Test management tools:
  - Capturing, cataloging, and administration of test cases and their priorities
  - Allow status tracking of the test cases
  - Can additionally support project management aspects during testing (i.e., resource and schedule planning for the tests).
  - Help test manager to plan the tests, remain informed about the status of test cases.
  - Allow the capture of requirements and the linking of them with the test cases needed for validation -> RTM

Tools for Test Management and Control

- Requirements management tools
  - 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).

Tools for Test Management and Control

- Incident management tools
  - Storing information about the attributes of incidents
  - Prioritizing incidents
  - Status of incidents
  - Assigning actions to people
  - 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).

Tools for Test Management and Control

- Configuration management tools
  - Storing information about versions and builds of software and testware
  - Traceability between software and testware and different versions or variants;
  - Keeping track of which versions belong with which configurations (e.g. operating systems, libraries, browsers);
  - Build and release management;
  - Baselining (e.g. all the configuration items that make up a specific release);
  - Access control (checking in and out).

Tools support for static testing

- Review process support tools
  - Common reference for review process
  - Storing and sorting review comments
  - Communicating comments to relevant people
  - Coordinating online reviews
  - Keeping track of comments
  - 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
  - Collecting metrics and reporting on key factors

Tools support for static testing

- Static analysis tools (D)
  - Calculate metrics such as cyclomatic complexity or nesting levels
  - Enforce coding standard
  - Analyze structure and dependencies
  - Did in code understanding
  - Identify anomalies or defects in code
- Modeling tools (D)
  - Identify inconsistencies and defects within model
  - Identify and prioritize areas of model for testing
  - Predicting system response and behavior under various situations
  - Help to understand the system functions and identify test conditions

Tool support for test specification

- Test design tool
  - Generate test input value
  - Generate expected results
- Test data preparation tools
  - Extract selected data records
  - Enable records to be sorted or arranged
  - Generate new records populated with pseudo-random data
  - Construct a large number of similar records

Tool support for execution and logging

- Test execution tools
  - Tools for automating test execution
    - Capture/playback
    - Data driven
    - Keyword driven
- Debuggers
- Test harness/unit test framework tools (D):
  - Offering mechanisms for executing test objects through programming interface
  - Used with test objects without UI that are not directly accessible for a manual test
  - Mainly required during component and integration testing

Tool support for execution and logging

- Test comparators
- Dynamic analysis:
  - Acquire additional information on the internal state of the software (e.g., information on allocation, usage, and release of memory)
  - Memory leaks, pointer allocation, or pointer arithmetic problems can be detected.
- Coverage measurement tools
- Security tools

Tool Support for Performance & Monitoring

- Performance-testing, load-testing and stress-testing tools
  - Generating a load on the system to be 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.

Tool support for Performance & Monitoring

- Monitoring tools
  - Identifying problems and sending an alert message to the administrator (e.g. network administrator);
  - Logging real-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).

Tool Support for Specific Application Areas

- There are static analysis tools for specific development platforms and programming languages, since each programming language and every platform has distinct characteristics.
- There are dynamic analysis tools that focus on security issues, as well as dynamic analysis tools for embedded systems.
- Commercial tool sets may be bundled for specific application areas such as web-based or embedded systems.

Tool Support Using Other Tools

- E.g a word processor or a spreadsheet are often used to store test designs, test scripts or test data
- Testers may also use SQL to set up and query databases containing test data
- Tools used by developers when debugging, to help localize defects and check their fixes, are also testing tools
- It is a good idea to look at any type of tool available to you for ways it could be used to help support any of the testing activities
- For example, testers can use Perl scripts to help compare test results

6.2 Effective use of tools

- Objectives:
  - LO-6.2.1 Summarize the potential benefits and risks of test automation and tool support for testing. (K2)
  - LO-6.2.2 Recognize that test execution tools can have different scripting techniques, including data driven and keyword driven. (K1)

Potential Benefits and Risks

- Potential benefits of using tools include:
  - Repetitive work is reduced.
  - Greater consistency and repeatability.
  - Objective assessment (e.g. static measures, coverage).
  - Ease of access to information about tests or testing (e.g. statistics and graphs about test progress, incident rates and performance).

Potential Benefits and Risks

- Risks of using tools include:
  - Unrealistic expectations for the tool (including functionality and ease of use).
  - Underestimating the time, cost and effort for the initial introduction of a tool (including training and external expertise).
  - Underestimating the time and effort needed to achieve significant and continuing benefits from the tool.
  - Underestimating the effort required to maintain the test assets generated by the tool.
  - Over-reliance on the tool (replacement for test design or where manual testing would be better).

Selection of Test Tool

- Requirement specification for the tool application
- Market research (creating an overview of possible candidates)
- Tool demonstrations and creation of a short list
- Evaluating the tools on the short list
- Reviewing of the results and selection of the tool

Introducing a tool into an organization

- The main considerations in selecting a tool for an organization include:
  - Assessment of organizational maturity, strengths and weaknesses and identification of opportunities for an improved test process supported by tools.
  - Identify the areas of the organization where tool support will help
  - Proof-of-concept to see whether the product works as desired
  - Evaluate the tool against clear requirements and objective criteria
  - Evaluation of the vendor
  - Identification of internal requirements for coaching and mentoring in the use of the tool.

Introducing a tool into an organization

- Starts with a pilot project, which has the following objectives:
  - Learn more details about the tool.
  - Evaluate how the tool fits with existing processes and practices, and determine what would need to change.
  - Decide on standard ways of using, managing, storing and maintaining the tool and the test assets.
  - Assess whether the benefits will be achieved at reasonable cost.

Introducing a tool into an organization

- Success factors for the deployment of the tool within an organization include:
  - Rolling out the tool to the rest of the organization incrementally.
  - Adapting and improving processes to fit with the use of the tool.
  - Providing training and coaching/mentoring for new users.
  - Defining usage guidelines.
  - Implementing a way to learn lessons from tool use.
  - Monitoring tool use and benefits.

Introducing a tool into an organization

- Successful tool introduction follows these six steps:
  - Execute a pilot project
  - Evaluate the pilot project experiences
  - Adapt the processes and implement rules for usage
  - Train the users
  - Introduce the tool in a stepwise fashion
  - Offer accompanying coaching

References

- ISTQB Foundation Syllabus
- Foundation of Software Testing: ISTQB Certification

Q & A