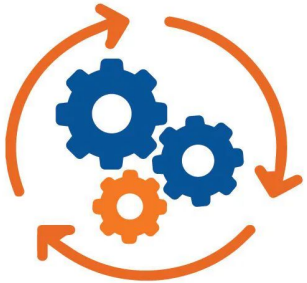
Chapter 3

Automated Test Tool Evaluation and Selection



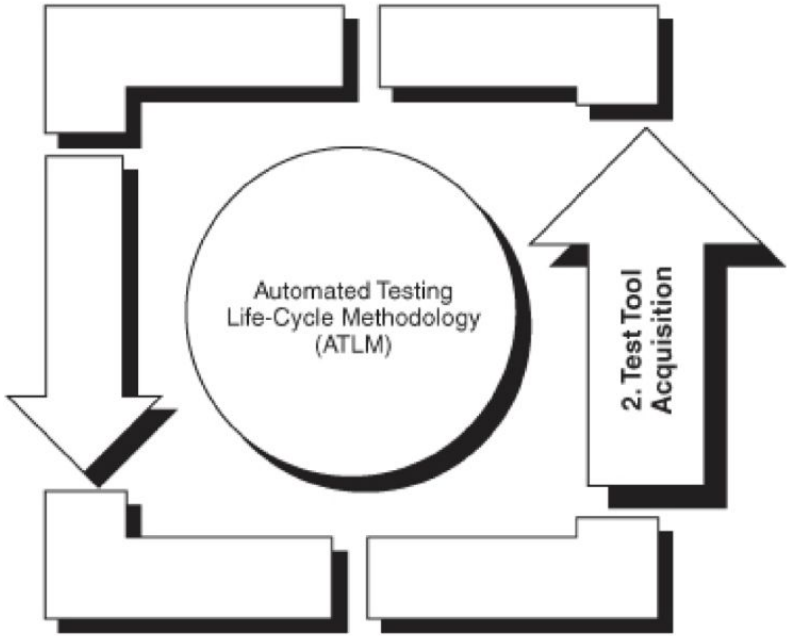
Lecturer: Nguyen Thanh Quan (MSc)
Email: tg_nguyenthanhquan_cntt@tdtu.edu.vn

Content

1. **The Organization's Systems Engineering Environment**
2. **Tools That Support the Testing Life Cycle**
3. **Test Tool Research**
4. **Evaluation Domain Definition**
5. **Hands-on Tool Evaluation**

Introduction

"If the only tool you have is a hammer, you tend to see every problem as a nail." —Abraham Maslow.

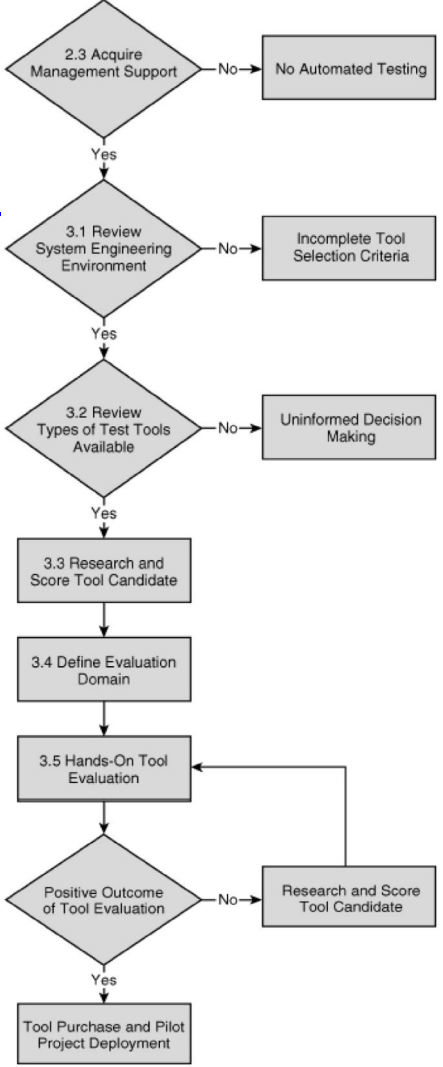


Introduction

- Frequently, the selection of an automated test tool is accomplished long after the development platform and development tools have been determined.
- In reality, a project often has a detailed system design in place before the concern for software test is addressed.
- Test tool cost and the formal and on-the-job training for the tool received by test team personnel represent an investment by the organization.
- The selected tool should fit the organization’s entire systems engineering environment.
- The test team needs to follow a structured approach for performing test tool evaluation and selection.

Introduction

- Once management has approved, the test engineer needs to take a methodical approach toward identifying the tool best suited for the situation.
- By reviewing the organization's systems engineering environment, the test engineer becomes familiar with the system and software architectures for each of the various projects within the organization.
- Defines the criteria for a tool evaluation.
- Identifies which of the various test tool types.
- Assesses the automated test tools available.
- Matches the test tool requirements with the types of test tools. Identifies an evaluation domain.



The Organization’s Systems Engineering Environment

The test engineer reviews the organization’s **systems engineering environment.**

- operating systems.
- programming languages.
- other aspects of the technical environment.

by addressing the questions and concerns.

Third-Party Input from Management, Staff, and End Users (1)

The following questions should be addressed:

- How will the tool be used within the organization?
- Will other groups and departments use the tool?
- What is the most important function of the tool?
- What is the least important function of the tool?
- How will the tool mainly be used?
- How portable must the tool be?

Third-Party Input from Management, Staff, and End Users (2)

During the survey, the test engineer identifies:

- the database architecture.
- technical application architecture (middleware, database, os)
- the languages used to develop the GUI.
- understanding of the detailed architecture design.
- performance under heavy loads.
- an overall idea of the organization’s engineering environment.
- automated test effort should accurately reflect customer quality concerns and priorities - state customers’ needs and expectations.

Tool Criteria Reflecting the Systems Engineering Environment (1)

The test engineer should review the system and software architectures within the organization

- Different projects may have a different set of test goals and objectives, and many types of tests can be performed.
- Which phase of the software development life cycle does the test team wish to apply automated testing?
- How will the test tool be used in the organization?
- Is there interest in a requirements management tool that could be applied during the requirements definition phase?
- Is there interest in performing usability testing during the design phase?

Tool Criteria Reflecting the Systems Engineering Environment (2)

The test engineer must identify which types of test tools are the most applicable

Several questions need to be posed:

- concentrate the automated test effort during the development phase?
- need a tool that supports memory leak testing?

- It is important to analyze the need for tools that support regression, stress, or volume testing.
- Should not limit the automated test tool selection criteria to a single project.

Level of Software Quality

Define the level of software quality expected for the project and determine crucial aspects of the software development

Whether the organization is seeking to comply with industry quality guidelines (ISO 9000, CMM)

- If 5 full time developers, too expensive for use of a variety of test tools.
- 30+ developers, justify the use of a greater variety of test tools.

High availability of hardware and software is a critical concern for financial situations.

Help Desk Problem Reports

- When an application or version of an application is **in operation**, the test team can monitor help desk trouble reports so as to r**eview the history of the most prevalent problems** of the application.

- If a new version of the application is **being developed**, the team can **focus** its **effort** on the most prevalent **problems** of the operational system, **identifying a test tool** that supports this kind of testing.

Budget Constraints

- When the test engineer has obtained management commitment for a test tool and must satisfy a large number of organizational test tool requirements on a limited budget:
    - The test engineer must be both selective and cost-conscious when looking for one or more tools to support requirements.
    - The test engineer may need to purchase a single tool that meets a majority of the requirements or one that best satisfies the most important requirements.

Types of Test

The types of tests to consider include regression testing, stress or volume testing, and usability testing

- What is the most important feature needed in a tool?
- Will the tool be used mainly for stress testing?
- Is this capability required for the particular project or set of projects?
- What is the goal?
- What functionality is desired?

Long-Term Investment Considerations

The use of one or more automated test tools should be contemplated for a longer period than just one year

- The tool should be viewed as a **long-term investment**, involving multiple tool **upgrades**.
- Who is the vendor? the test tool vendor
  - should be **available** to answer questions
  - should provide r**egular upgrades** to keep up with technological development.
- Does the product have a good track record?
- What is its industry acceptance?
- The test engineer should examine the entire organization's system engineering make-up.

Test Tool Process

When evaluating a test tool, keep in mind that the test team will need to introduce the test tool within the organization

- The test engineer needs to verify that **management** is **willing** to commit adequate resources to support the test tool introduction process.
- The test team needs to **ensure** that the **tool** is **implemented** in a manner that promotes its adoption.

If **no one** in the organization is **using the tool**, the **effort** to obtain and incorporate the tool will have been **wasted**.

Avoiding Shortcuts

To determine the test tool requirements, the test engineer will have to address additional questions

- Will I be supporting a large testing effort?
- Will there be enough time for the necessary people to learn the tool within the constraints of the schedule?

Given a situation where the project schedule **precludes** the introduction of an appropriate test tool for the organization, it may be **advisable not to** introduce an automated test tool.

Tools That Support the Testing Life Cycle

Programming Phase	Syntax checkers/ debuggers	Allow for syntax checking and debugging capability; usually come with built-in programming language compilers
	Memory leak and runtime error detection tools	Detect runtime errors and memory leaks
	Source code testing tools	Verify maintainability, portability, complexity, cyclomatic complexity, and standards compliance
	Static and dynamic analyzers	Depict quality and structure of code
	Various code implementation tools	Depending on the application, support code generation, among other things
	Unit test tools	Automates the unit testing process

Tools That Support the Testing Life Cycle

It is important to become familiar with the different types of tools available on the market

Metric Tools	Code (test) coverage analyzers or code instrumentors	Identify untested code and support dynamic testing
	Metrics reporters	Read source code and display metrics information
	Usability measurements	Provide user profiling, task analysis, prototyping, and user walkthroughs

- Should conduct a needs/improvement analysis.
- Tools could be most beneficial in improving the system development process.
- A tool evaluation should be performed.

Business Analysis Phase Tools

Numerous tools on the market support the business analysis phase

- Some tools support various methodologies, such as the Unified Modeling Language (UML).
- Business analysis tools have the ability to record process improvement opportunities and allow for data organization capabilities, thus improving the testing life cycle.

Business Modeling Tools

Business modeling tools support the creation of process models, organization models, and data models

- Business modeling tools allow for recording definitions of user needs and for automating the rapid construction of flexible, graphical client-server applications.
- Some business modeling tools integrate with other phases of the system/testing life cycle, such as the data modeling phase, design phase, programming phase, and testing and configuration management phase.
- Tools enhance process modeling throughout the system life cycle, while simultaneously supporting the production of testable systems.

Configuration Management Tools

Configuration management tools should be deployed early in the life cycle

- The purpose of early deployment is to manage change and institute a repeatable process.
- The final outputs of each system life-cycle phase should be baselined in a configuration management tool.

Defect Tracking Tools

It is important to use configuration management tools throughout the testing life cycle

- It is also important to use defect tracking tools from the beginning and throughout the testing life cycle.
- All defects or software problem reports encountered throughout the system life cycle should be documented and managed to closure.
- The identification of defects is the primary goal of testing and quality assurance activities.

Technical Review Management Tools

One of the best defect detection strategies relies on technical reviews or inspections

- Reviews allow defects to be discovered early in the life cycle.
- Reviews and inspections represent a formal evaluation technique applicable to software requirements, design, code, and other software work products.
- Technical review management tools allow for automation of the inspection process, while facilitating communication.
- The reviews support automated collection of key metrics, such as action items discovered during a review.

Documentation Generation Tools

- Simplify the testing life cycle by reducing the effort required to manually produce software documentation.
- Helpful throughout the entire testing life cycle.

Requirements Definition Phase Tools

Software needs to be assessed relative to an understanding of what the software is intended to do

- Within requirements specifications or use case definitions.
- The quality of defined requirements can make the test effort relatively painless or extremely arduous.
- When all of the information needed by a test engineer is usable, the requirements are said to be test-ready.
- If requirements are not test-ready or testable, test engineers must search for missing information (a tedious task).

Requirements Management Tools (1)

Requirements management tools permit requirements to be captured quickly and efficiently

- Requirements can be recorded in a natural language, such as English or in a formal language such as LOTOS or Z.
- Requirements can be modeled graphically, using tools such as Validator/Req
- One method of modeling requirements involves use cases. A UC defines:
  - the behavior of a system
  - semantic entity without revealing the entity's internal structure.
  - a sequence of actions, interacting with actors.

Requirements Management Tools (2)

Many requirements management tools support information traceability

- Traceability involves more than just tracking requirements.
- Linking all information together procedures is a critical factor in demonstrating project compliance and completeness.

➔ Requirements management tools can also automatically determine which test procedures are affected when a requirement is modified.

➔ Requirements management tools support information management.

➔ Such tools support lessons learned review activities to facilitate the management of issues and defects.

Requirements Verifiers (1)

Requirements verifiers are relatively new tools

In the past, recorded requirements information could be checked in two ways:

- By using functions of some requirements analysis tools to verify that information conformed to certain methodology rules.
- By performing manual reviews on the information.

➔ Assure that the requirements information represented a testable product.

Requirements Verifiers (2)

To be testable, requirements information must be unambiguous, consistent, quantifiable, and complete.

- Unambiguous if it has one, and only one, definition.
- Consistent if each of its terms is used in one, and only one, way.

➔ For instance, **report** as both a name and an action, however, would make the specification inconsistent.

- From a test engineer's point of view, **completeness** means that the requirements contain necessary and sufficient information for testing.
- An automated verifier cannot determine whether the collection of requirement statements is complete.

➔ Checking the completeness of the requirements specification must therefore be performed **manually**.

Tools for the Analysis and Design Phase (1)

- A requirements specification defines ***what*** a software system is expected to do.
- The design phase determines ***how*** these requirement specifications will be implemented.

Tools for the Analysis and Design Phase (2)

Visual Modeling Tools

- Used during the business analysis phase, helpful during the design phase.
- Improves communication among various team members.
- Improves quality.
- Increases visibility and predictability.
- Help reuse information between requirements phase and design phase.
- Enhance the design effort and thus the testing phase.

Tools for the Analysis and Design Phase (3)

Test Procedure Generators

A test procedure generator creates test procedures by statistical, algorithmic, or heuristic means.

- In statistical test procedure generation, the tool chooses input structures and values to form a statistically random distribution, or a distribution that matches the usage profile of the software under test.
- In algorithmic test procedure generation, the tool follows a set of rules or procedures and employ action-, data-, logic-, event-, and state-driven strategies.
- In heuristic or failure-directed means, the tool employs information from the test engineer (pass/failure in the past)

Tools for the Analysis and Design Phase (5)

Programming Phase Tools - Syntax Checkers/Debuggers

- Usually bundled within a high-level language compiler.
- Important in improving the testability of software during the programming phase.

Debugging can include setting breakpoints in the code to allow the executing program to be stopped for debugging, inserting a stop condition, allowing source code to be viewed during debugging, and allowing program variables to be viewed and modified.

Tools for the Analysis and Design Phase (6)

Programming Phase Tools - Memory Leak and Runtime Error Detection Tools

- Showing where memory has been allocated but to which no pointers exist.
- Runtime errors may be detected in third-party libraries, shared libraries, and other code.
- These tools can identify problems such as uninitialized local variables, stack overflow errors, and static memory access errors, just to name a few.
- These tools are very beneficial additions to the testing life cycle.

Tools for the Analysis and Design Phase (7)

Programming Phase Tools - Source Code Testing Tools

An early code checker test tool, called LINT, was provided to application developers as part of the UNIX operating system.

- Looks for misplaced pointers, uninitialized variables, and deviations from standards.
- Help identify miniscule problems prior to each inspection.

Tools for the Analysis and Design Phase (8)

Programming Phase Tools - Static and Dynamic Analyzers

Some tools allow for static and dynamic analysis of code.

- Perform static analysis, assessing the code in terms of programming standards, complexity metrics, unreachable code, and much more.
- Support dynamic analysis, which involves the execution of code using test data to detect defects at runtime, as well as detection of untested code, analysis of statement and branch execution, and much more.

➔ Reports in both textual and graphical form.

Metrics Tools (1)

Metrics Reporter

- Reads source code and displays metrics information, often in graphical format.
- Reports complexity metrics in terms of data flow, data structure, and control flow.
- Provides metrics about code size in terms of modules, operands, operators, and lines of code.
- Helps the programmer correct and groom code.
- Helps the test engineer determine which parts of the software code require the most test attention.

Metrics Tools (2)

Code Coverage Analyzers and Code Instrumentors

- Gives the development team insight into the effectiveness of tests and test suites (e.g McCabe's Visual Test Tool).
- Can quantify the complexity of the design.
- Measures the number of integration tests required to qualify the design.
- Produces the desired integration tests.
- Measures the number of integration tests that have not been executed.
- Measures multiple levels of test coverage, including segment, branch, and conditional coverage.

Metrics Tools (2)

Usability Measurement

These types of tools evaluate the usability of a client/server application.

Other Testing Life-Cycle Support Tools (1)

Test Data Generators

Test data can be used for all testing phases, especially during performance and stress testing, simplifying the testing process.

Other Testing Life-Cycle Support Tools (2)

File Compare Utilities

Comparisons are useful in validating that regression tests produce the same output files as baselined information, before code fixes were implemented.

Other Testing Life-Cycle Support Tools (3)

Simulation Tools

- Simulation modeling tools can simulate the behavior of application-under-test models.
- Using various modifications of the target application environment as part of “what-if” scenarios.
- These tools provide insight into the performance and behavior of existing or proposed networks, systems, and processes.

Testing Phase Tools (1)

Test Management Tools

- Support the testing life cycle by allowing for planning, managing, and analyzing all aspects of it (e.g Rational's TestStudio)
- Simplifying the entire testing life-cycle process.

Testing Phase Tools (2)

Network Testing Tools

- The advent of applications operating in a client-server, multitier, or Web environment has introduced new complexity to the test effort.
- The client-server architecture involves three components: the server, the client, and the network.

➔ Interplatform connectivity also increases the potential for errors.

Test tools allow the test engineer to monitor, measure, test, and diagnose performance across the entire network.

Testing Phase Tools (3)

GUI Application Testing Tools (Record/Playback Tools)

- Allows the test engineer to create (record), modify, and run (playback) automated tests across many environments.
- Recording is a computer program, which is referred to as a test “script” in a high-level language.
- It is necessary to work with its inherent scripting language.
- **Comparator** automatically compares actual outputs with expected outputs and logs the results.

Testing Phase Tools (4)

Load/Performance/Stress Testing Tools

- E.g Rational's PerformanceStudio allow for load testing, where the tool can be programmed to run a number of client machines simultaneously to load the client/server system and measure response time.
- Load testing typically involves various scenarios to analyze how the client/server system responds under various loads.
- Stress testing involves the process of running the client machines in highstress scenarios to see when and if they break.

Testing Phase Tools (5)

Environment Testing Tools

- Mainframe, UNIX, X-Windows, and Web.
- The number of test tools supporting Web applications is increasing.

Testing Phase Tools (6)

Year 2000 (Y2K) Testing Tools

- Such tools parse and report on mainframe or client-server source code regarding the date impact.
- Some Y2K tools support the baseline creation of Y2K data.
- Others allow for data aging for Y2K testing.
- Still other Y2K tools provide data simulation and simulate the Y2K testing environment.

Testing Phase Tools (7)

Product-Based Test Procedure Generators

- Reads and analyzes source code, and then derives test procedures from this analysis.
- Tries to create test procedures that exercise every statement, branch, and path (structural coverage).

➔ The problem with this tool is that it tries to achieve structural coverage by working from the code structure rather than from the requirements specification.

It cannot distinguish between good program code and bad program code (at present some IDE can).

Test Tool Research (1)

- The test engineer must translate the need for a test tool type into one or more specific test tool candidates.
- The test team needs to develop a test tool specification and evaluation form.
- The test engineer should investigate whether a requirements management tool will be used.

Several questions need to be posed:

- Do you need a capture/playback tool or a code coverage tool, or both?
- What are you trying to accomplish with the test tool?
- Do you need a tool for test management?
- Do you plan to use the tool for load testing or only regression testing?

Test Tool Research (2)

Evaluation Scorecard - Automated GUI Testing (Record/Playback) Tool

Test Tool Characteristic	Weight (1–10)	Score (1–5)	Value (1–50)
Ease of Use			
Learning curve	7	5	35
Easy to maintain the tool	5	5	25
Easy to install—tool may not be used if difficult to install	5	3	15
Tool Customization			
Can the tool be customized (can fields maintained by tool be added or deleted)?	7	4	28
Does the tool support the required test procedure naming convention?	8	4	32
Platform Support			
Can it be moved and run on several platforms at once, across a network (that is, cross-Windows support, Win95, and WinNT)?	8	4	32
Multiuser Access			
What database does the tool use? Does it allow for scalability?	8	5	40
Network-based test repository—necessary when multiple access to repository is required	8	5	40

Test Tool Research (3)

Evaluation Scorecard - Automated GUI Testing (Record/Playback) Tool

Defect Tracking (For more detail on evaluating defect tracking tools, see Chapter 8)			
Does the tool come with an integrated defect-tracking feature?	10	3	30
Tool Functionality			
Test scripting language—does the tool use a flexible, yet robust scripting language? What is the complexity of the scripting language: Is it 4 GL? Does it allow for modular script development?	9	5	45
Complexity of scripting language	9	5	45
Scripting language allows for variable declaration and use; allows passing of parameters between functions	9	5	45
Does the tool use a test script compiler or an interpreter?	9	5	45
Interactive test debugging—does the scripting language allow the user to view variable values, step through the code, integrate test procedures, or jump to other external procedures?	8	4	32
Does the tool allow recording at the widget level (object recognition level)?	10	5	50

Test Tool Research (4)

Evaluation Scorecard - Automated GUI Testing (Record/Playback) Tool

Does the tool allow for interfacing with external .dll and .exe files?	9	5	45
Published APIs—language interface capabilities	10	4	40
ODBC support—does the tool support any ODBC-compliant database?	10	4	40
Is the tool intrusive (that is, does source code need to be expanded by inserting additional statements)?	9	4	36
Communication protocols—can the tool be adapted to various communication protocols (such as TCP/IP, IPX)?	9	3	27
Custom control support—does the tool allow you to map to additional custom controls, so the tool is still compatible and usable?	10	3	30
Ability to kick off scripts at a specified time; scripts can run unattended	9	5	45
Allows for adding timers	10	5	50
Allows for adding comments during recording	7	5	35
Compatible with the GUI programming language and entire hardware and software development environment used for application under test (i.e., VB, Powerbuilder)	10	5	50
Can query or update test data during playback (that is, allows the use of SQL statements)	10	4	40
Supports the creation of a library of reusable function	10	5	50
Allows for wrappers (shells) where multiple procedures can be linked together and are called from one procedure	10	5	50
Test results analysis—does the tool allow you to easily see	10	3	30

Test Tool Research (5)

Evaluation Scorecard - Automated GUI Testing (Record/Playback) Tool

whether the tests have passed or failed (that is, automatic creation of test results log)?			
Test execution on script playback—can the tool handle error recovery and unexpected active windows, log the discrepancy, and continue playback (automatic recovery from errors)?	5	3	15
Allows for synchronization between client and server	5	10	50
Allows for automatic test procedure generation	8	5	40
Allows for automatic data generation	8	5	40
Y2K compliance	10	5	50
Reporting Capability			
Ability to provide graphical results (charts and graphs)	8	5	40
Ability to provide reports	8	5	40
What report writer does the tool use?	8	5	40
Can predefined reports be modified and/or can new reports be created?	8	5	40

Test Tool Research (6)

Evaluation Scorecard - Automated GUI Testing (Record/Playback) Tool

Performance and Stress Testing			
Performance and stress testing tool is integrated with GUI testing tool	9	5	45
Supports stress, load, and performance testing	10	3	30
Allows for simulation of users without requiring use of physical workstations	10	3	30
Ability to support configuration testing (that is, tests can be run on different hardware and software configurations)	10	3	30
Ability to submit a variable script from a data pool of library of scripts/data entries and logon IDs/password	10	3	30
Supports resource monitoring (memory, disk space, system resources)	10	3	30
Synchronization ability so that a script can access a record in database at the same time to determine locking, deadlock conditions, and concurrency control problems	10	5	50
Ability to detect when events have completed in a reliable fashion	9	5	45
Ability to provide client to server response times	10	3	30
Ability to provide graphical results	8	5	40
Ability to provide performance measurements of data loading	10	5	50

Test Tool Research (7)

Evaluation Scorecard - Automated GUI Testing (Record/Playback) Tool

Version Control			
Does the tool come with integrated version control capability?	10	4	40
Can the tool be integrated with other version control tools	8	3	24
Test Planning and Management			
Test planning and management tool is integrated with GUI testing tool	8	5	40

Test Tool Research (8)

Evaluation Scorecard - Automated GUI Testing (Record/Playback) Tool

Test planning and management tool is integrated with requirements management tool	8	5	40
Test planning and management tool follows specific industry standard on testing process (such as SEI/CMM, ISO)	7	4	28
Supports test execution management	10	5	50
Allows for test planning—does the tool support planning, managing, and analyzing testing efforts? Can the tool reference test plans, matrices, and product specifications to create traceability?	10	5	50
Allows for measuring test progress	10	5	50
Allows for various reporting activities	9	4	36
Pricing			
Is the price within the estimated price range?	10	4	40
What type of licensing is being used (floating, fixed)?	7	3	21
Is the price competitive?	9	4	36

Test Tool Research (9)

Evaluation Scorecard - Automated GUI Testing (Record/Playback) Tool

Vendor Qualifications			
Maturity of product	8	4	32
Market share of product	8	4	32
Vendor qualifications, such as financial stability and length of existence. What is the vendor's track record?	8	4	32
Are software patches provided, if deemed necessary?	8	4	32
Are upgrades provided on a regular basis?	8	5	40
Customer support	10	3	30
Training is available	9	4	36
Is a tool Help feature available? Is the tool well documented?	9	5	45
Availability and access to tool user groups	8	4	32
Total Value			2,638

Improvement Opportunities (1)

At the end of the test life cycle, test program review activities are performed

- Suggest the need for automated testing tools and identify those processes or products that need improvement.
- Outline how an automated tool could be expected to help improve the process or product.
- The test engineer needs to incorporate the results of test program review activities when selecting the criteria for a new test tool.
- Narrow down the search by eliminating tools that don’t meet the minimal expectations.

Improvement Opportunities (2)

Many questions need to be asked:

- How will the tool be used in the organization?
- What is its most important function?
- How portable must the tool be to support multiple platforms?
- With which system life-cycle phases should the tool integrate?

Evaluation Domain Definition (1)

Experience shows that often tools don't work as expected within the particular environment

- It is advantageous to first test the tool in an isolated test environment (test lab) before applying the test tool on a pilot project.
- The test environment should be similar enough to the pilot project environment.
- The hardware/software configuration within the test lab.
- A single or a broader evaluation domain is preferred.

The test team should be able to select an application development project as a pilot for applying the test tool.

Evaluation Domain Definition (2)

Experience shows that often tools don’t work as expected within the particular environment

- The organizational structure of the testing team will need to be considered.
- The structure of the test organization affects the characteristics of the desired test tool (A centralized/decentralized test team).
- It is necessary to define the role that each test engineer will play in the test tool evaluation process.

Hands-on Tool Evaluation (1)

The test engineer responsible for selecting an automated test tool

- Surveying the systems engineering environment.
- Using an evaluation scorecard to grade each candidate test tool.
- Identifying an isolated test environment.
- Defining a target evaluation domain,
- Identifying the individuals who will perform a hands-on evaluation of the test tool in the test environment.

Hands-on Tool Evaluation (2)

- Contact the test tool vendor to request a product demonstration.
- Consider the professionalism demonstrated by the representative.
- Assess whether the vendor representative will be supportive and easy to work with?
- Ask for a test tool evaluation copy from the vendor.
- Must clearly understand the duration of trial.
- Should avoid a purchase prior to shipping vendor's products.
- Should also understand the required functions of each candidate test tool.

➔ The goal is to ensure that the test tool performs as advertised and that the tool works within the required environment.

Hands-on Tool Evaluation (3)

Evaluation Report

During the test tool demonstration:

- Compare the test tool's performance with its rating on desired test tool characteristics.

If the test tool's rating significantly differs from the baseline score, the test engineer may need to reconsider whether that test tool represents the best product for the particular requirement.

Hands-on Tool Evaluation (3)

An example of a typical evaluation report document outline:

- **Introduction**: describes its purpose and scope, and provides some background information.
- **Summary**: summarize the process that has taken place as well as the roles and participation of particular groups.
  - How will this tool help?
  - Where is the return on the investment?
- **Background Information**: Include names, addresses, and contact information for all potential vendors, including information pertaining to test tools that were not formally evaluated.

Hands-on Tool Evaluation (4)

An example of a typical evaluation report document outline:

- **Technical Findings**: Summarize the results and highlight points of particular interest. Note which product received the best score and why.
- **Product Summaries**: summarize the results of the evaluation of each vendor and its product.
  - Provide the results of the evaluation scorecard for each test tool
  - Raise issues that go beyond the absolute score.
  - Price.
- **Conclusion**: Reiterate the objective and the evaluation team's recommendation.

License Agreement (1)

The test engineer needs to review the license agreement before it is accepted by the organization's Purchasing Department.

The topics below can be considered when negotiating a test tool license agreement:

- **Named Users versus Concurrent Users** (e.g number of copies that can be run simultaneously)
- **Extended Maintenance** (renew license, maintenance cost): the maintenance cost may increase from one year to the next.
- **Termination**: If the contract mentions termination due to "material breach," make sure that the licenser specifies what this phrase specifically entails or cites a governing law.

License Agreement (2)

The test engineer needs to review the license agreement before it is accepted by the organization's Purchasing Department.

The topics below can be considered when negotiating a test tool license agreement:

- **Upgrades:** the vendor may indicate that certain functionality will be released in a later version
  - Specific date.
  - Have them commit to providing the upgrades.

Chapter summary (1)

- ★ The organization’s test team will select a test tool that fits the criteria of the organization’s system engineering environment.
- ★ The test engineer needs to identify which of the various test tool types might potentially apply on a particular project.
- ★ The test engineer should define the level of software quality that is expected from the project, and determine which aspects of the software development are the most crucial to a particular project or effort.
- ★ The test engineer narrows down the test tool search.

Chapter summary (2)

- ★ An evaluation scorecard can be used to determine which tool best fits the particular requirements.
- ★ An optional evaluation scoring method involves sizing up the candidates using only the most important test tool characteristics.
- ★ The test engineer needs to contact the test tool vendor to request a product demonstration and ask for an evaluation copy.
- ★ An evaluation report should be prepared that documents the results of the first-hand examination of the test tool.

References

1. Poston, R. A Guided Tour of Software Testing Tools. San Francisco: Aonix, 1988. www.aonix.com.

2. Adapted from SQA Process “Cust_Chk.doc,” January 1996. See www.rational.com.

3. Greenspan, S. “Selecting Automated Test Tools During a Client/Server Migration.” Paper presented at STAR conference, Orlando, Florida, May 13–17, 1996.

4. Used with permission of Steven Greenspan. “Selecting Automated Test Tools During a Client/Server Migration.” Paper presented at STAR conference, Orlando, Florida, May 13–17, 1996.