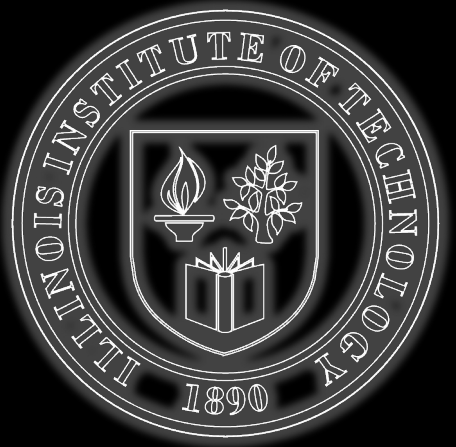
ILLINOIS TECH | College of Computing

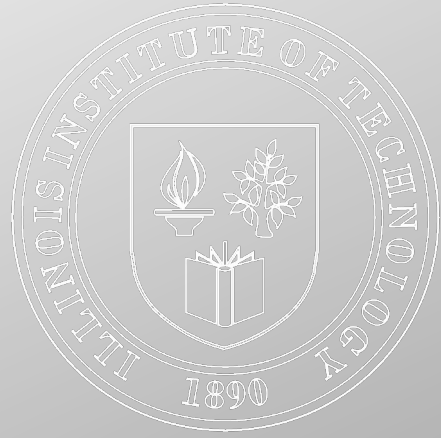
ITMD 536 Software Testing & Maintenance

Chapter 6
Tool Support for Testing



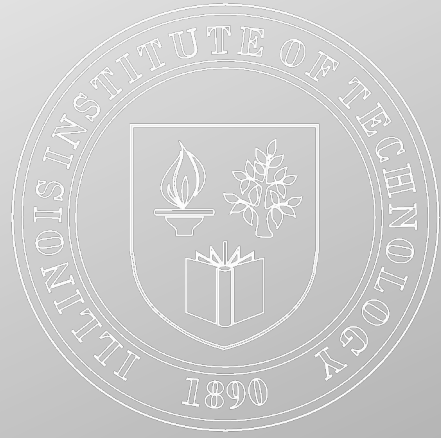
Objectives

- What is a tool?
- What is the use of a tool?
- Who needs tools?
- What are the benefit of using tools?
- What is use of tools in technology?
- What tools are used in testing?
- What are the risks of using test tools?
- How do you introduce a test tool?



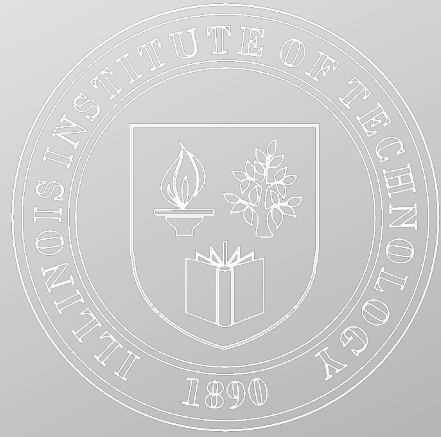
Tool Support for Testing

- What is a tool?
- What is the use of a tool?
- Who needs tools?
- What are the benefit of using tools?
- What is use of tools in technology?
- What tools are used in testing?
- What are the risks of using test tools?
- How do you introduce a test tool?



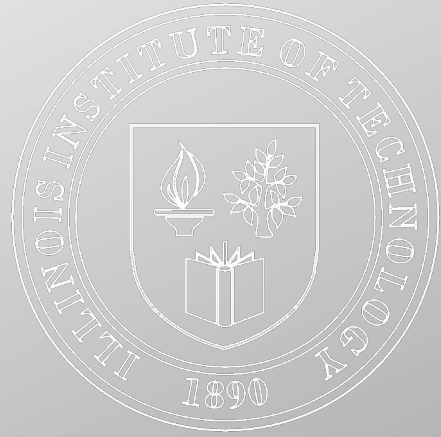
6.1 Types of Test Tools

- Types of test tools used in software testing are:
  - Test execution tools
  - Performance testing tools
  - Static analysis tools
  - Test management tools
  - Capture/playback tools



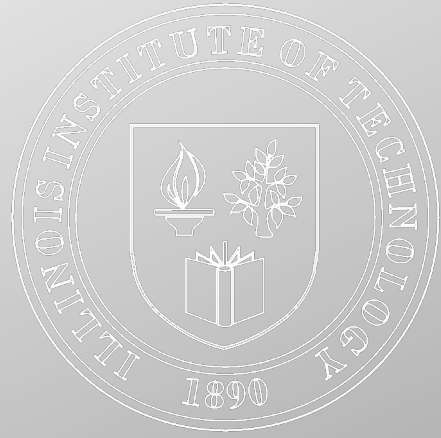
6.1 Types of Test Tools

- Configuration management tools
- Coverage tools
- Debugging tools
- Dynamic analysis tools
- Incident management tools
- Load testing tools
- Modelling tools



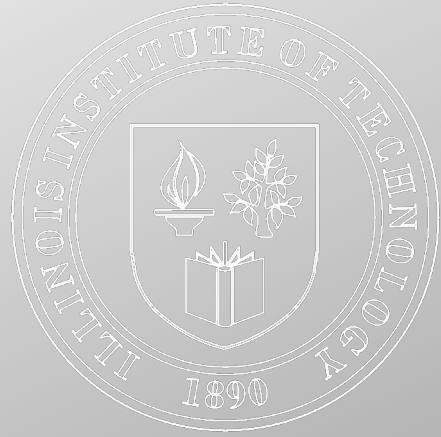
6.1 Types of Test Tools

- Monitoring tools
- Probe effect
- Requirement management tools
- Review tools
- Security testing tools
- Security tools
- Static code analysis



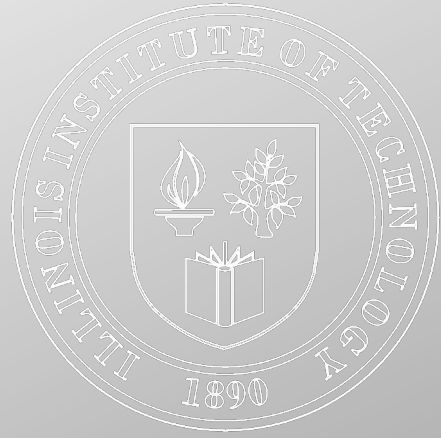
6.1 Types of Test Tools

- Static code analyzer
- Stress testing tools
- Test comparator
- Test comparison
- Test data preparation tools
- Test design tools
- Test framework
- Unit test framework tools and
- Volume testing tools



6.1.1 Meaning/Purpose of Tool Support for Testing

- Tools are used directly in testing.
- Tools helps manage testing process.
- Tools can monitor file activity for an application.
- Tools can be spreadsheets to manage test assets or progress document automated or manual tests.



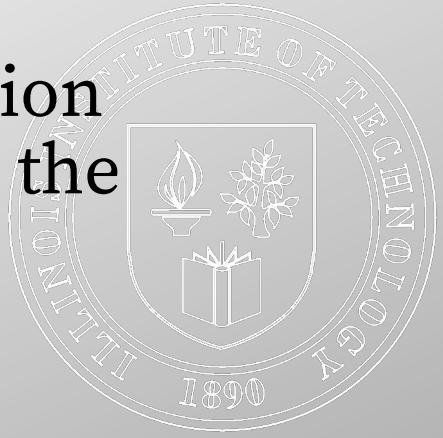
6.1.1 Meaning/Purpose of Tool Support for Testing

- Test Execution Tools (include via so-called test frameworks)
- **Test Framework:** It has three distinct meanings:

1. Reusable and extensible testing libraries that can be used to build testing tools (which are also called **test harness)**

2. A type of design of test automation (e.g., **data-driven** and, **keyword-driven**) and

3. An overall process of execution of testing. In the Foundation syllabus, and in this book, especially in this chapter, we use the term **test frameworks** in its first two meanings



6.1.2 Test Tool Classification

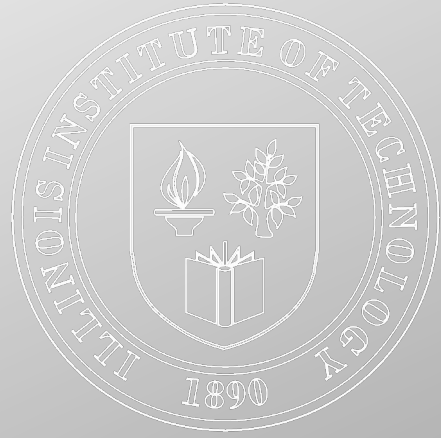
- Tools are grouped by the testing activities to support management activities, tools to support static testing
- There are certain things that computers can do much faster than humans such as add up 20 three-digit numbers quickly?
- A tool that measures some aspect of software may have unexpected side-effects on that software
- For example, a tool that measures timings for non-functional (performance) testing needs to interact very closely with that software in order to measure it
- A performance tool will set a start time will set a start time and a stop time for a given transaction in order to measure the response time, for example
- But the act of taking that measurement, i.e. storing the time at those two points, could actually make the whole transaction take slightly longer than it would do if the tool wasn't measuring the response time

6.1.2 Test Tool Classification

- **Probe Effect:** The effect on the component or system by the measurement instrument when the component or system is being measured, e.g. by a performance testing tool or monitor. For example performance may be slightly worse when performance testing tools are being used.

6.1.3 Tool Support for Management of Testing and Test

- **Test Management Tool:** A tool that provides support to the test management and control part of a test process.
- It often has several capabilities, such as testware management, scheduling of tests, the logging of results, progress tracking, incident management and test reporting
- **Interfaces to other tools**
- Test execution tools (test running tools)
- Incident management tools
- Requirement management tools
- Configuration management tools

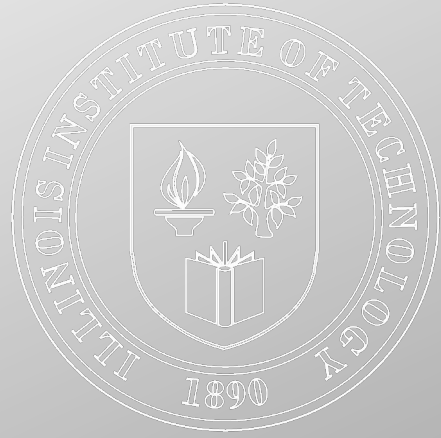


6.1.3 Tool Support for Management of Testing and Test

- **Requirement management tool:** A tool that supports the recording of requirements, requirements attributes and annotation, and facilitates traceability through layers of requirements and requirement change management. (Rational & Jira)
- Some requirements management tools also provide facilities for static analysis such as consistency checking and violations to pre-defined requirements rules
- Storing, checking consistency, gaps, prioritizing, traceability, interfacing and coverage of requirements by set of tests

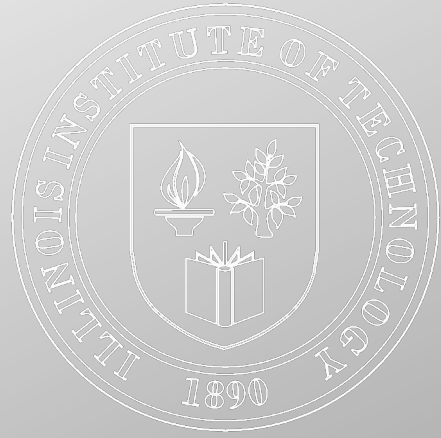
6.1.3 Tool Support for Management of Testing and Test

- **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 and provide reporting facilities BMC Remedy, Jira, Redmine & Bugzilla
- Storing incidents severity
- Storing attachments
- Prioritizing incidents
- Assigning actions to people
- Status
- Reporting statics/metrics



6.1.3 Tool Support for Management of Testing and Test

- **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.(CMDB, Clear Case…)
- Storing information about versions and builds of the software, testware
- Traceability between software and testware versions and variants
- Track which version belongs to which configuration (operating systems, libraries and browsers)
- Build and release management
- Baseline (e.g. all the configuration items that make up a specific release)
- Access control (checking in and out)

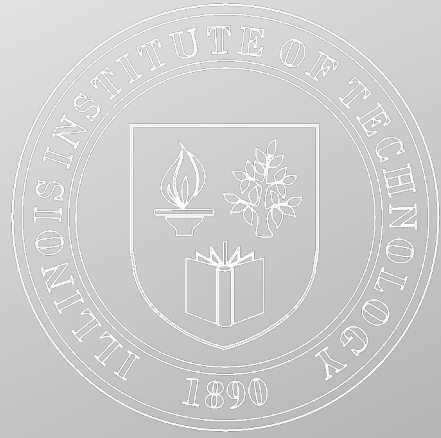


6.1.4 Tools for Static Testing

- **Static Testing:** Testing of a component or system at specification or implementation level without execution of that software, e.g. reviews or static analysis
- **Review Tool:** 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

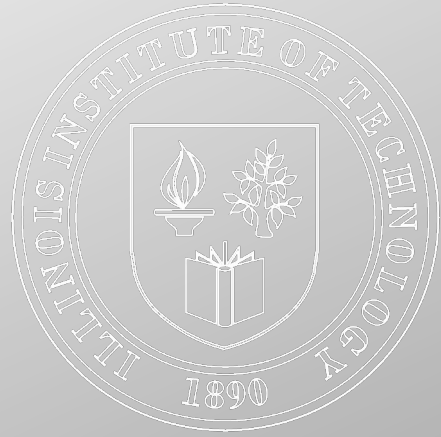
6.1.4 Tools for Static Testing

- A common reference for the review process or processes
- Storing and sorting review comments
- Keeping track of comments
- Providing traceability
- Repository for rules, procedures & checklists, entry and exit criteria
- Monitoring the reviews & collecting metrics



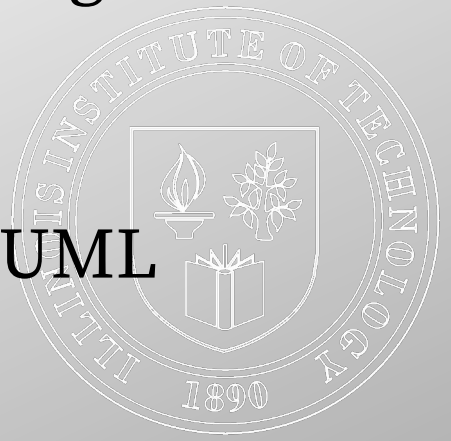
6.1.4 Tools for Static Testing

- **Static Analysis Tool (static analyzer):** A tool that carries out static analysis
- Static analysis tools are normally used by developers as part of the development and component testing process
- **Static Code Analyzer:** A tool that carries out static code analysis. The tool checks source code, for certain properties such as conformance to coding standards, quality metrics or data flow anomalies
- **Static Code Analysis:** Analysis of source code carried out without execution of that software
- Calculate metrics such as cyclomatic complexity or nesting levels
- Enforce coding standards
- Analyze structures and dependencies
- Aid in code understanding
- Identify anomalies or defects in the code



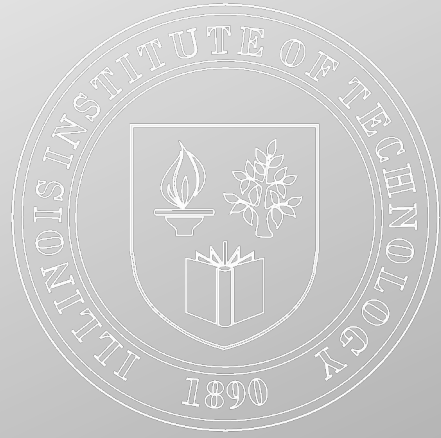
6.1.4 Tools for Static Testing

- **Modeling Tools (D):** A tool that supports the creation, amendment and verification of models of the software or system.
- This tool helps to validate models of the system or software
- Checks the consistency
- Identifies inconsistencies and defects within the model
- Helps to identify and prioritize areas of the model for testing
- Predicting system response and behavior under various situations, such as load
- Identify test conditions using modeling language such as UML



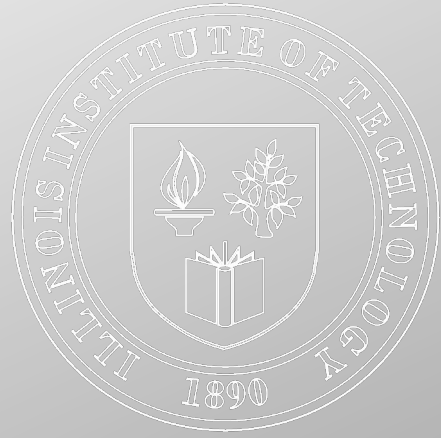
6.1.5 Tool Support for Test Specification

- **Test design tool:** A tool that supports the test design activity by generating test inputs from a specification that may be held in a CASE tool repository, e.g. requirements managements tool, from specified test conditions held in the tool itself, or from code.
- Generating test input values from
- Requirements
- Design models (state, data or object)
- Code
- Graphical user interfaces
- Test conditions
- Generating expected results



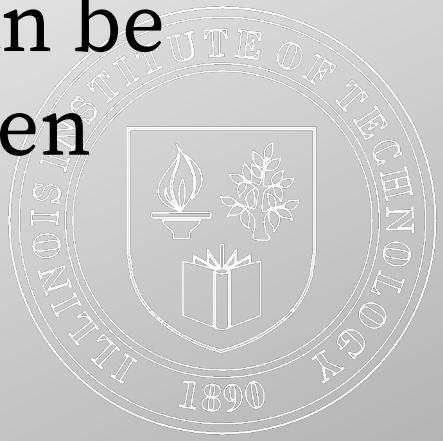
6.1.5 Tool Support for Test Specification

- **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
- Enables data to be selected from and existing database or created, generated, manipulated and edited for use in tests
- Extract selected data records from files of database
- ‘Message’ data records to make them anonymous (data protection)
- Enable records to be stored or arranged in different order
- Generate new records
- Construct a large number of similar records from a template



6.1.6 Tool Support for Test Execution and logging

- **Test Execution Tool:** A type of test tool that is able to execute other software using an automate test script, e.g. capture/playback.
- **Capture/Playback Tool:** A type of test execution tool where inputs are recorded during manual testing in order to generated automated test scripts that can be executed later (i.e. replayed). These tools are often used to support automated regression testing

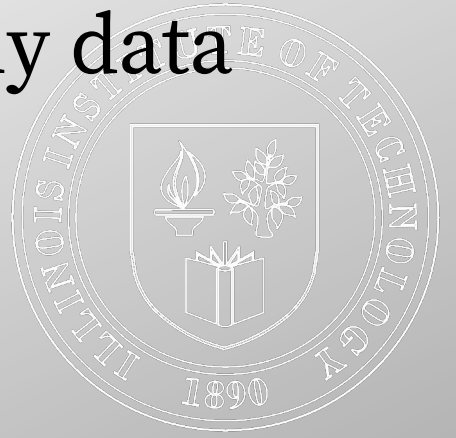


6.1.6 Tool Support for Test Execution and logging

- Test execution tools use a scripting language to drive the tool
- It is difficult to maintain the capture script because:
- It is closely tied to the flow and interface presented by the GUI
- It may rely on the circumstances, state and context of the system at the time the script was recorded
- The test input information is ‘hard-coded’, i.e. it is embedded in the individual script for each test

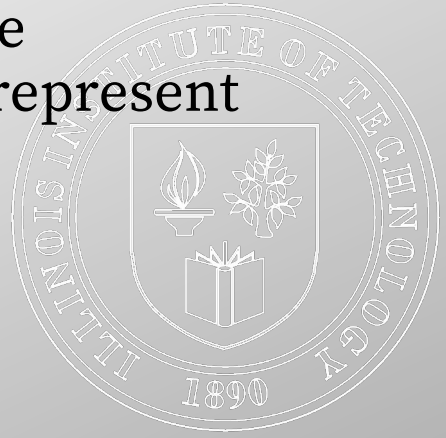
6.1.6 Tool Support for Test Execution and logging

- Capturing (recording) test inputs while inputs tests are executed manually
- Storing and 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



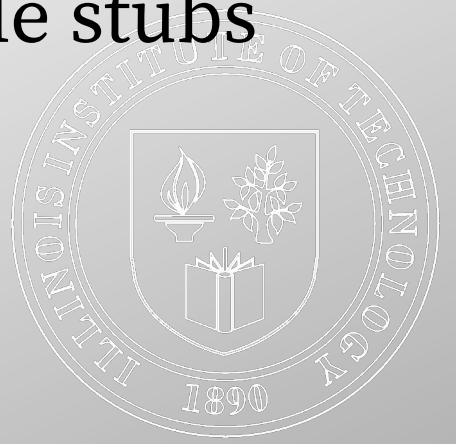
6.1.6 Tool Support for Test Execution and logging

- **Dynamic comparison** (while the test is running) of screens, elements, links controls, objects and values
- Ability to initiate post-execution comparison
- Logging results (pass/fail)
- Masking of filtering
- 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



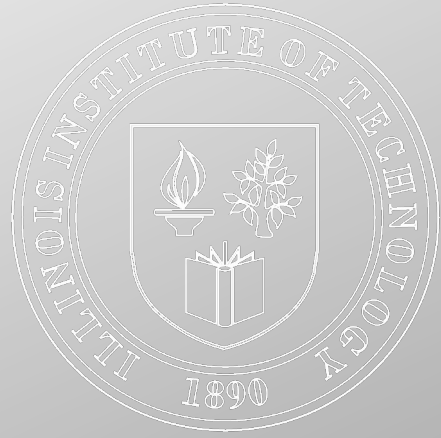
6.1.6 Tool Support for Test Execution and logging

- **Test harness/unit test framework tools (D):**
- A test harness provides stubs and drivers, which are small programs that interact with the software under 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 divers. It also provides other support for the developer, such as debugging capabilities



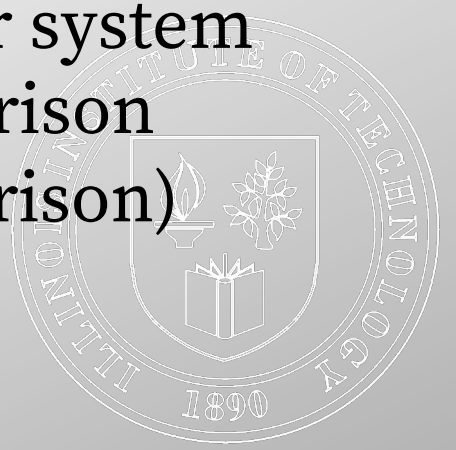
6.1.6 Tool Support for Test Execution and logging

- 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
- Storing tests
- Support for debugging
- Coverage measurement at code level



6.1.6 Tool Support for Test Execution and logging

- **Test comparators:** A test tool to perform automated test comparison of actual results with expected results
- A test comparator helps automated aspects of that comparison (stand-alone tool)
- **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. (batch run)



6.1.6 Tool Support for Test Execution and logging

- 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
- **Coverage measurement tools (D):**
- A tool that provides objective measures of what structural elements, e.g. statements, branches have been exercised by a test suite
- A coverage tool first identifies the elements or coverage items that can be counted

6.1.6 Tool Support for Test Execution and logging

- Identifying coverage items
- Calculating the percentage of coverage items that were exercised by a suite of tests
- Reporting coverage items that have not been exercised as yet
- Identifying test inputs to exercise as yet uncovered items (test design tool functionality)
- Generating stubs and drivers (if part of a unit test framework)
- The coverage tools only measure the coverage of the items that they can identify

6.1.6 Tool Support for Test Execution and logging

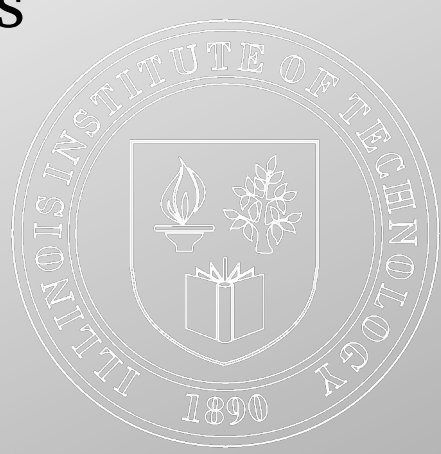
- **Security testing tool:** A tool that provides support for testing security characteristics and vulnerabilities
- **Security:** Attributes of software products that bear on its ability to prevent unauthorized access, whether accidental or deliberate, to programs and data
- **Security tool:** A tool that supports operational security.
- Security testing tool can be used to test security by trying to break into a system, whether or not its protected by a security tool.
- Identifying viruses

6.1.6 Tool Support for Test Execution and logging

- Detecting intrusions such as denial of service attacks;
- Simulating various types of external attacks
- Probing for open ports to other externally visible points of attack
- Identifying weaknesses in password files and passwords
- Security checks during operation, e.g. for checking integrity of files, and intrusion detection, e.g. checking results of test attacks

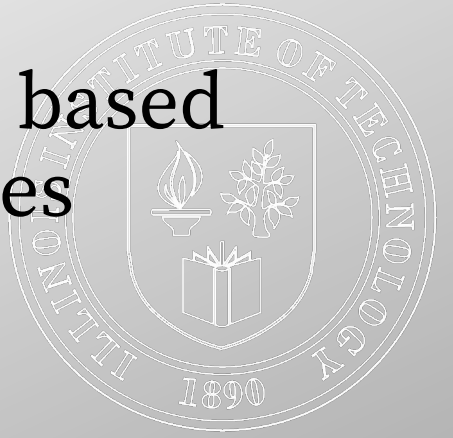
6.1.7 Tool Support for Performance and Monitoring

- **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 and to monitor the allocation, use and del-allocation of memory and to flag memory leaks
- Detecting memory leaks
- Identifying pointer arithmetic errors such as null pointers
- Identifying time dependencies
- It helps find dead links



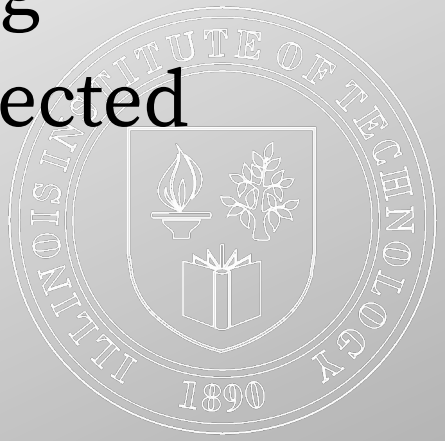
6.1.7 Tool Support for Performance and Monitoring

- **Performance-testing, load-testing and stress-testing tools:** A tool to support performance testing that usually has two main facilities load generation and 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



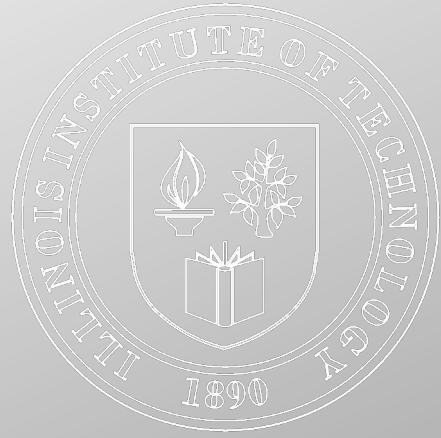
6.1.7 Tool Support for Performance and Monitoring

- A '***load***' test checks that the system can cope with its expected number of transactions
- **Volume testing:** Testing where the system is subjected to large volumes of data
- A volume test checks the large volume of data
- **Stress testing tool:** A tool that supports stress testing
- A stress test is one that goes beyond the normal expected usage of the system with respect to load or volume



6.1.7 Tool Support for Performance and Monitoring

- 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 response time

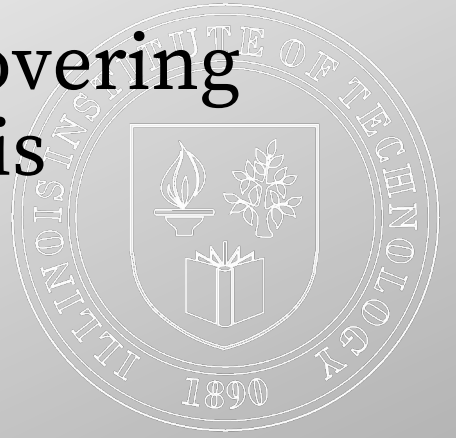


6.1.7 Tool Support for Performance and Monitoring

- **Monitoring tools:** 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
- Monitoring tools are used to continuously keep track of the status of the system in use, in order to have the earliest warning of problems and to improve services. Monitoring tools for servers, networks, databases, security, performance, website and internet usage, and applications

6.1.7 Tool Support for Performance and Monitoring

- Identifying problems and sending and alert message to the 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 operations with the analysis performed afterwards)

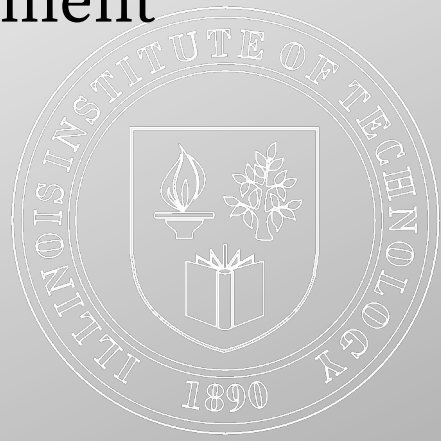


6.1.8 Tool Support for Specific Application Area

- Web-based performance-testing tools, as well as performance- testing tools for back-office systems.
- Dynamic analysis tools that focus on security issues, as well as dynamic analysis tools for embedded systems.
- Commercial tool sets may be bundled for web-based or embedded systems.

6.1.9 Tool Support Using other Tools

- Testing tools used by the testers are SQL to set up and query databases containing test data. Tools used by the developers when debugging, to help localize defects and check their fixes are also testing tools. Developers use debugging tools when identifying and fixing defects
- **Debugging tool:** A tool used by programmers to reproduce failures, investigate the state of programs and find the corresponding defect. Debugging enable programmers to execute programmers to execute programs step by step, to halt a program at any program statement and to set and examine program variables

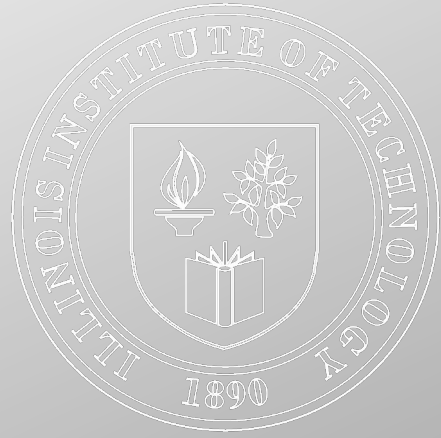


6.1.10 Tool Support for Specific Testing Needs

- Tools should be used for data quality assessment, including reviewing and verifying data conversion and migration rules
- Tools can help ensure that the processed data is correct, and complete, and that it complies to pre-defined, context-specification standards, even when the volume of data is large. You can also use tools for usability testing

6.2 Effective use of Tools – Benefits and Risks

- The reason for acquiring tools to support testing is to gain benefits, by using a software program to do certain tasks that are better done by a computer than by a person



6.2.1 Benefits of Using Tools

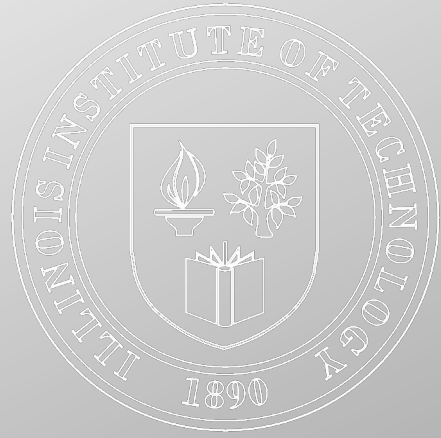
- **Benefits include:**
- Reduction of repetitive work
- Greater consistency and repeatability
- Objective assessment
- Ease of access to information about tests or testing

6.2.2 Risks of using Tools

- **Risks include:**
- Unrealistic expectations for the tool
- Underestimating the time, cost and effort needed to achieve significant and continuing benefits from the tool
- Understanding the effort required to maintain the test assets generated by the tool
- Over-reliance on the tool, including relying on tools for test design or test execution where manual testing would work better
- Failing to use proper configuration management and version control facilities for the test assets created for and by the tool

6.2.2 Risks of using Tools

- Failing to consider and mange issues related to relationships and interoperability between critical tools, such as requirements management tools, version control tools, incident management tools, defect tracking tools and tools from multiple vendors
- Possibility of the tool vendor going out of business, retiring the tool, or selling the tool to a different vendor, or in the open-source world, the possibility of orphanage of the tool or suspension of the project by the open-source community



6.2.2 Risks of using Tools

- Poor or completely non-existent vendor response, whether for support, upgrades, or defect fixes
- Various uncertainties and unforeseen problems, such as the inability to support a new platform
- The skill needed to create good tests
- The skill needed to use the tools well, depending on the type of tool

6.2.3 Special consideration for Types of Tools

- **Test execution tools:** The test execution tool must have some way of knowing what to do – this is the script for the tool. The script becomes a program, written in a programming language
- The scripting language may be specific to a particular tool, or it may be a more general language
- **Scripting language:** A programming language in which executable test scripts are written, used by a test execution tool (e.g. a capture/playback tool)

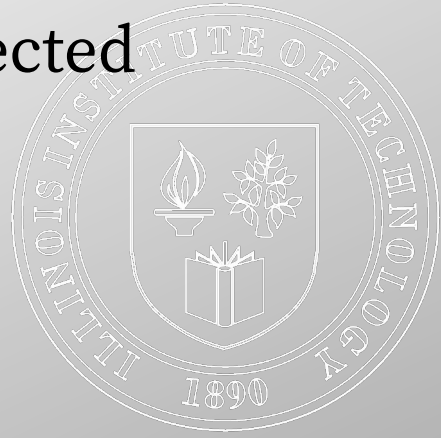
6.2.3 Special consideration for Types of Tools

- **Five levels of scripting by (Fewster and Graham):**

1. Linear scripts (created manually or captured by recording a manual test)
2. Structured scripts (using selection and iteration programming structures)
3. Shared scripts (where scripts can be called by other scripts so can be re-used – shared scripts also require a formal script library under configuration management)
4. Data-driven scripts (where test data is in a file or spreadsheet to be read by a control script)
5. Keyword-driven scripts (where all of the information about the test is stored in a file or spreadsheet, with a single control scripts that implements the tests described in the file using shared and keyword scripts)

6.2.3 Special consideration for Types of Tools

- The script doesn't know what the expected result is until you program it in – it only stores inputs that have been recorded, not test case
- A small change to the software may invalidate dozens or hundreds of scripts
- The recorded script can only cope with exactly the same conditions as when it was recorded. Unexpected events (e.g. a file that already exists) will not be interpreted correctly by the tool
- Data driven scripts allow the data, i.e. the test inputs and expected outcomes to be stored separately from the script



6.2.3 Special consideration for Types of Tools

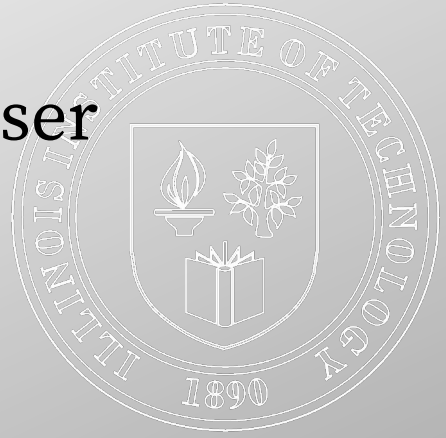
- **Data-Driven:** A scripting technique that stores test input and expected results in a table or spreadsheet, so that a single control script can execute all of the tests in the table. Data driven testing is often used to support the application of test execution tools such as capture/playback tools

6.2.3 Special consideration for Types of Tools

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

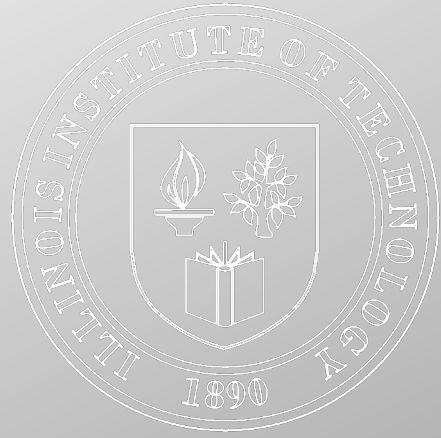
6.2.3 Special consideration for Types of Tools

- When using a performance testing tool we are looking at the transaction throughput, the degree of accuracy of a given computation, the computer resources being used for a given level of transactions, the time taken for certain transactions or the number of users that can use the system at once
- The design of the load to be generated by the tool (e.g. random input or according to user profiles)
- Timing aspects (e.g. inserting delays to make simulated user input more realistic)



6.2.3 Special consideration for Types of Tools

- The length of the test and what to do if a test stops prematurely
- Narrowing down the location of a bottleneck
- Exactly what aspects to measure (e.g. user interaction level or server level)
- How to present the information gathered

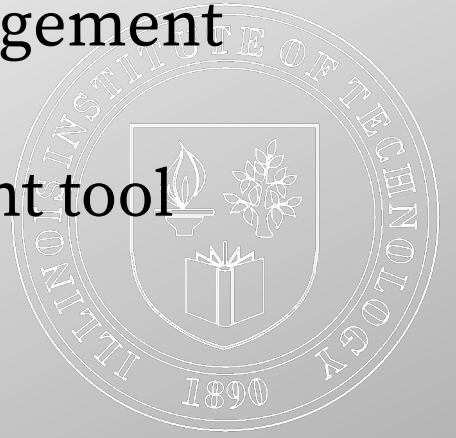


6.2.3 Special consideration for Types of Tools

- **Static Analysis Tool:** A tool that carries out static analysis.
- Developers use this tool to identify potential problems in code before the code is executed and they can also help to check the code is written to coding standards
- Static analysis tools can generate a large number of messages, for example by finding the same thing every few lines

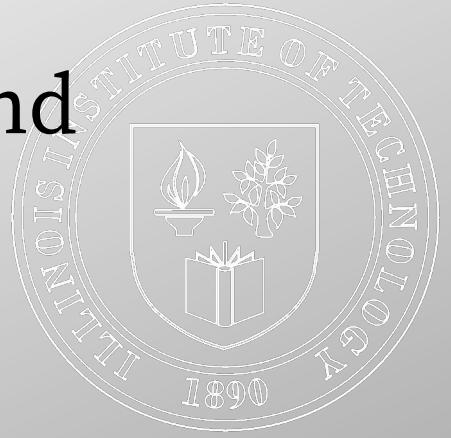
6.2.3 Special consideration for Types of Tools

- **Test management tool:** A tool that provides support to the test management and control part of a test process. It often has several capabilities, such as testware management, scheduling of tests, the logging of results, progress tracking incident management and test reporting
- It is important to have a defined test process before test management tools are introduced
- It is important to have a defined test process before test management tools are introduced
- If it’s working well manually it will work with test management tool



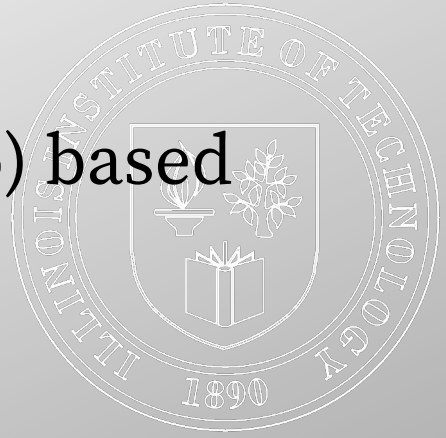
6.3 Introducing a Tool into an Organization

- **6.3.1 Main Principles:** Assessment of the organization's maturity (e.g. readiness for change)
- Identification of the areas within the organization where tool support will help to improve testing processes
- Evaluation of tools against clear requirements and objective criteria



6.3.1 Main Principles

- Proof-of-concept to see whether the product works as desired and meets the requirements and objectives defined for it
- Evaluation of the vendor (training, support and other commercial aspects) or open-source network of support
- Identifying and planning internal implementation (including training, coaching and mentoring for those new to the use of the tool)
- Estimation of the return on investment (cost-benefit ratio) based on a concrete and realistic business case

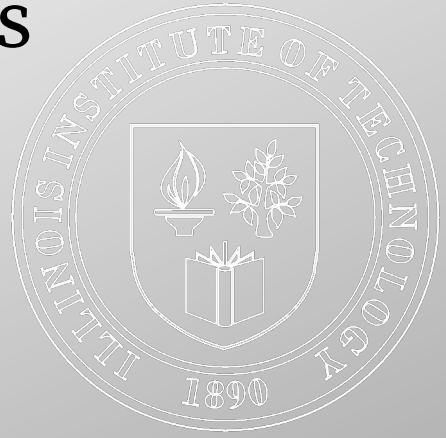


6.3.2 Pilot Project

- One of the ways to do a proof-of-concept is to have a pilot project as the first thing done with a new tool
- This will use the tool in earnest but on a small scale, with sufficient time to explore different ways of using the tool
- **Objectives for this new tool are:**
- To learn more about the tool (more detail, more depth)
- To see how the tool would fit with existing processes or documentation, how those would need to change to work well with the tool and how to use the tool to streamline existing processes

6.3.2 Pilot Project

- To decide on standard ways of using the tool that will work for all potential users (e.g. naming conventions, creation of libraries, defining modularity, where different elements will be stored, & maintained)
- To evaluate the pilot project against its objectives



6.3.3 Success Factors

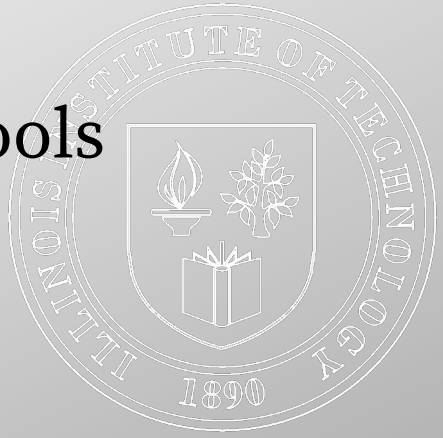
- Success is not guaranteed or automatic when implementing a testing tool, but many organization have succeeded.
- Incremental roll-out (after the pilot)
- Adapting and improving processes, testware and tool artefacts to get the best fit and balance between them and the use of the tool

6.3.3 Success Factors

- Providing adequate support, training, coaching and mentoring of new users
- Defining and communicating guidelines for the use of the tool, based on what was learned in the pilot
- Implementing a continues improvement mechanizing as a tool use spreads through more of the organizations

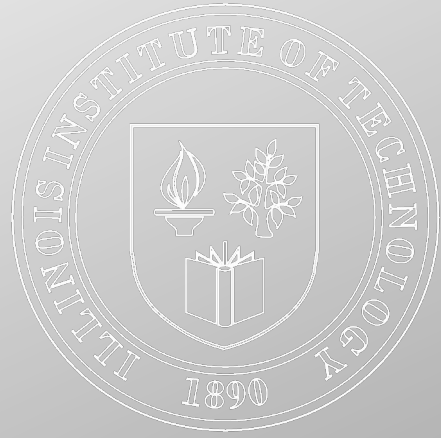
6.3.3 Success Factors

- Monitoring the use of the tool and the benefits achieved and adapting the use of the tool to take account of what is learned
- Provide continuing support for anyone using test tools, such as the test team; for example, technical expertise is needed to help non-programmer testers who use keyword-driven test automation
- Continuous improvement of tool use should be based on information gathered from all teams who are using test tools



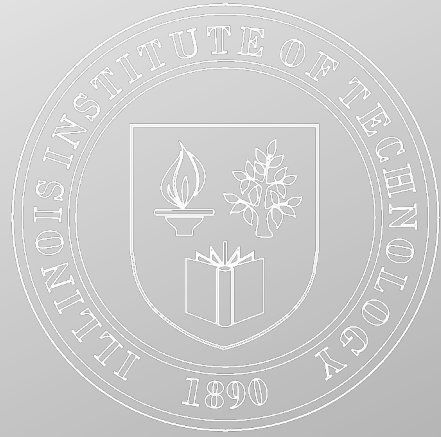
List of Testing Tools

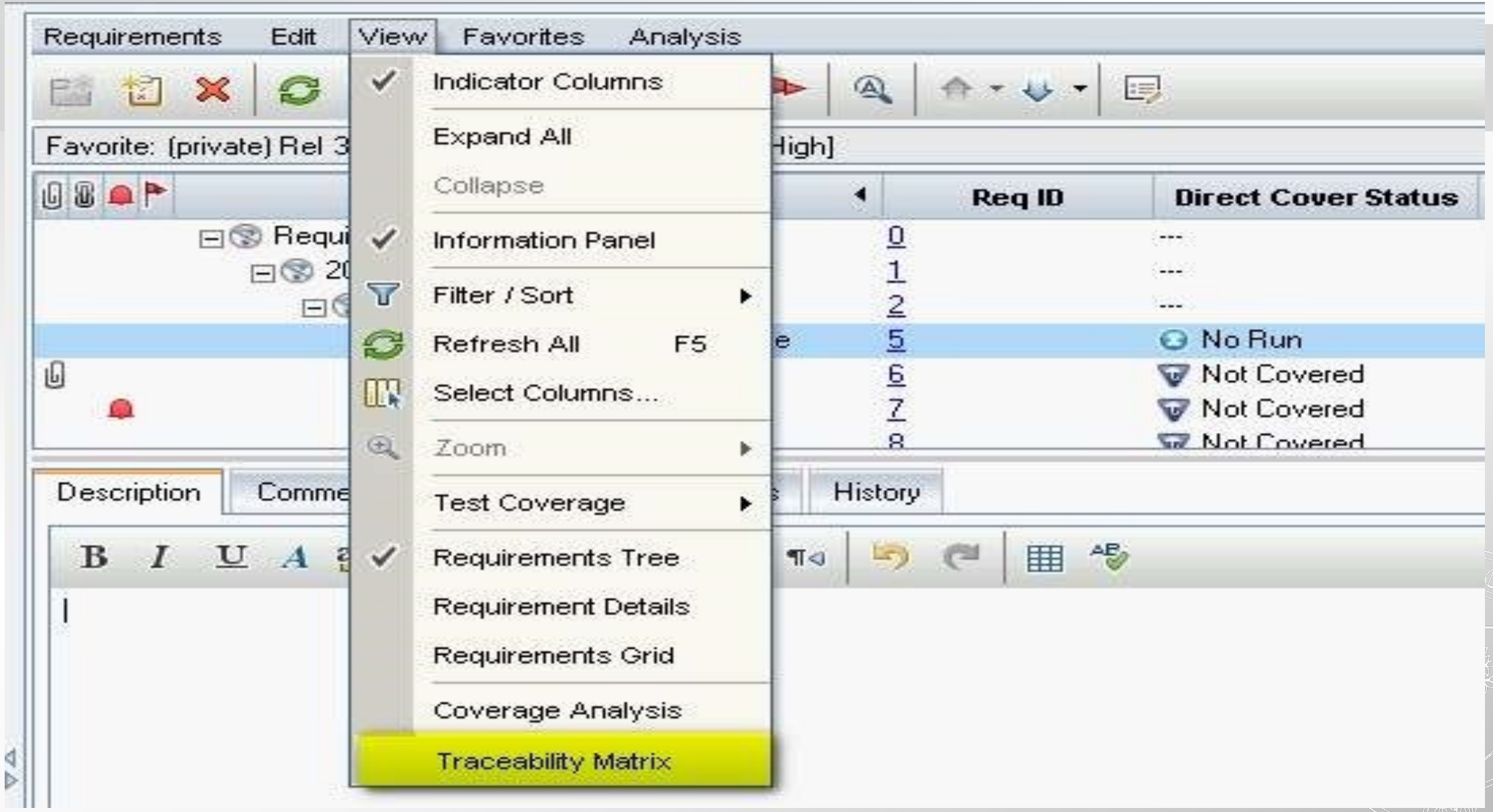
- Selenium
- Quick Test Professional (QTP)
- HP WinRunner
- Test Manager
- Rational Tester
- Load Runner
- Rational Quality Manager

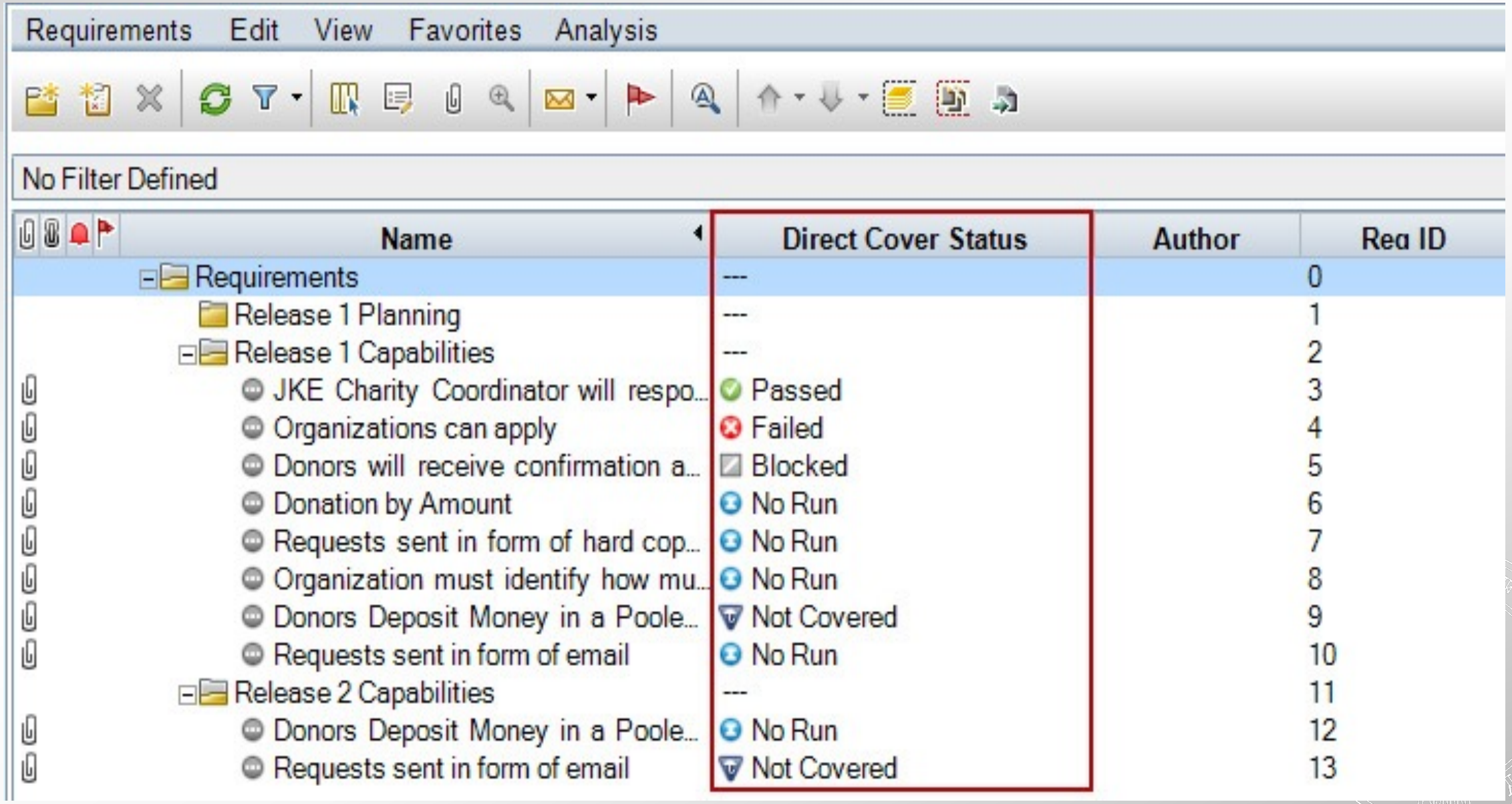


List of Defect Tracking Tools

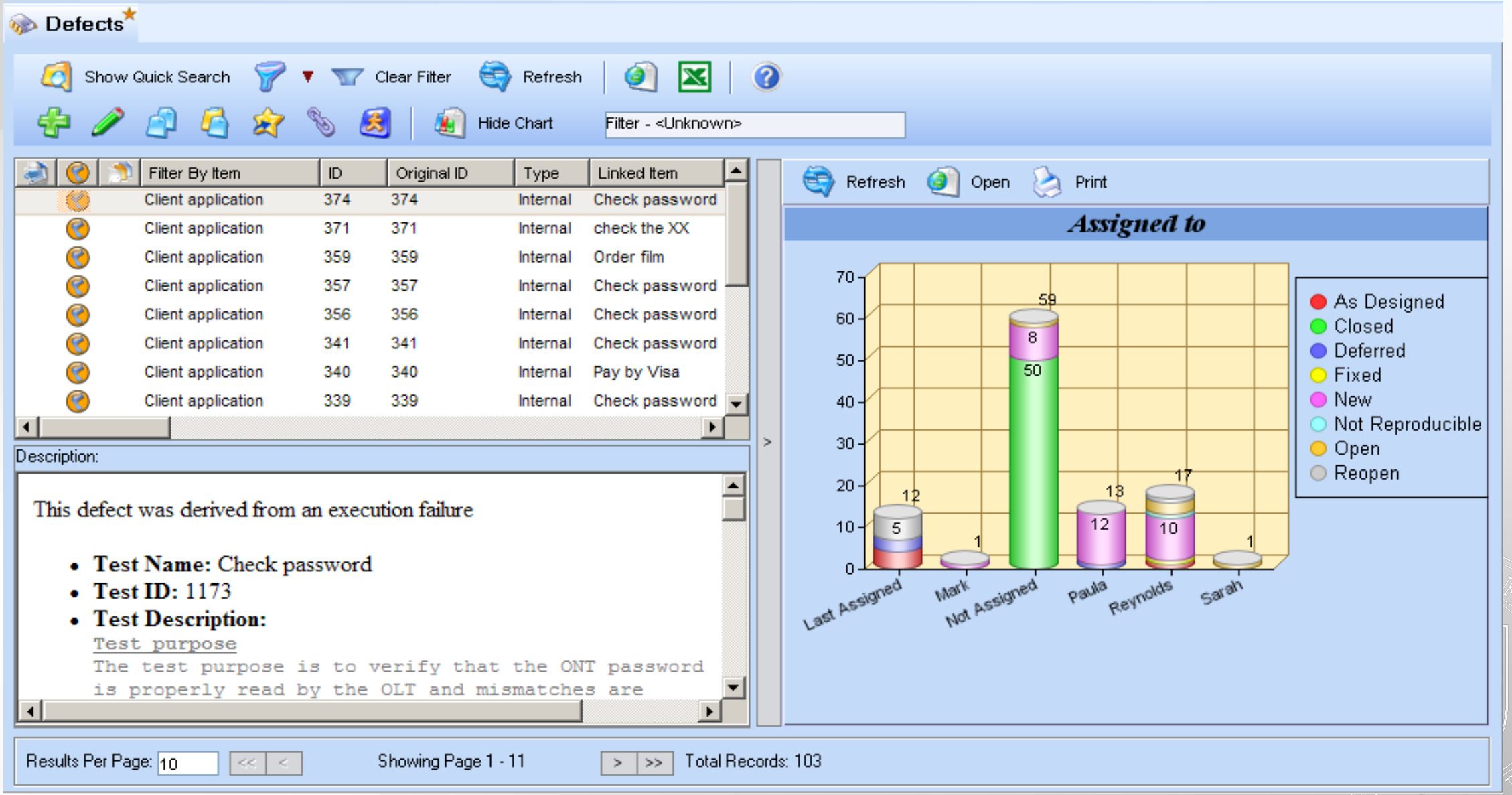
- Jira
- Remedy
- Bugzilla
- SMT
- Tracker
- Redmine




Requirements
Edit
View
Favorites
Analysis
Indicator Columns
Expand All
Collapse
Information Panel
Filter / Sort
Refresh All F5
Select Columns...
Zoom
Test Coverage
Requirements Tree
Requirement Details
Requirements Grid
Coverage Analysis
Traceability Matrix
Favorite: (private) Rel 3
High]
Req ID
Direct Cover Status
0 ---
1 ---
2 ---
5 No Run
6 Not Covered
7 Not Covered
Description
History



Name	Direct Cover Status	Author	Req ID
Requirements	---		0
Release 1 Planning	---		1
Release 1 Capabilities	---		2
JKE Charity Coordinator will respo...	Passed		3
Organizations can apply	Failed		4
Donors will receive confirmation a...	Blocked		5
Donation by Amount	No Run		6
Requests sent in form of hard cop...	No Run		7
Organization must identify how mu...	No Run		8
Donors Deposit Money in a Poole...	Not Covered		9
Requests sent in form of email	No Run		10
Release 2 Capabilities	---		11
Donors Deposit Money in a Poole...	No Run		12
Requests sent in form of email	Not Covered		13


Defects
Show Quick Search
Clear Filter
Refresh
Hide Chart
Filter - <Unknown>
Filter By Item
ID
Original ID
Type
Linked Item
Client application 374 374 Internal Check password
Client application 371 371 Internal check the XX
Client application 359 359 Internal Order film
Client application 357 357 Internal Check password
Client application 356 356 Internal Check password
Client application 341 341 Internal Check password
Client application 340 340 Internal Pay by Visa
Client application 339 339 Internal Check password
Description:
This defect was derived from an execution failure
Test Name: Check password
Test ID: 1173
Test Description:
Test purpose
The test purpose is to verify that the ONT password is properly read by the OLT and mismatches are
Refresh
Open
Print
Assigned to
70
60
50
40
30
20
10
0
12
5
1
59
8
50
13
12
17
10
1
Last Assigned
Mark
Not Assigned
Paula
Reynolds
Sarah
As Designed
Closed
Deferred
Fixed
New
Not Reproducible
Open
Reopen
Results Per Page: 10
Showing Page 1 - 11
Total Records: 103

Chapter 6 – 1. Questions and Answers

1. Which tools help to support static testing?

a. Static analysis tools, and test execution tools.

b. Review tools, static analysis tools, and coverage measurement tools.

c. Dynamic analysis tools, and modelling tools.

d. Review tools, static analysis tools, and modelling tools.

Chapter 6 -2. Questions and Answers

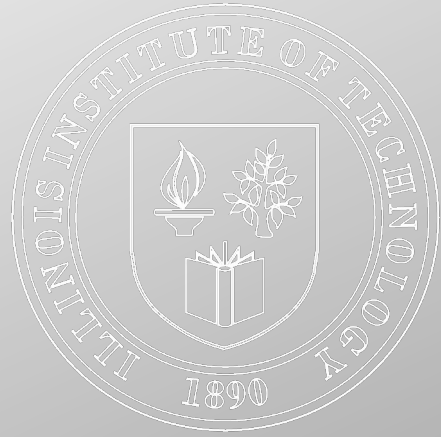
2. Which test activities are supported by test harness or unit test framework tools?

a. Test management and control.

b. Test specification and design.

c. Test execution and logging.

d. Performance and monitoring.



Chapter 6 – 3. Questions and Answers

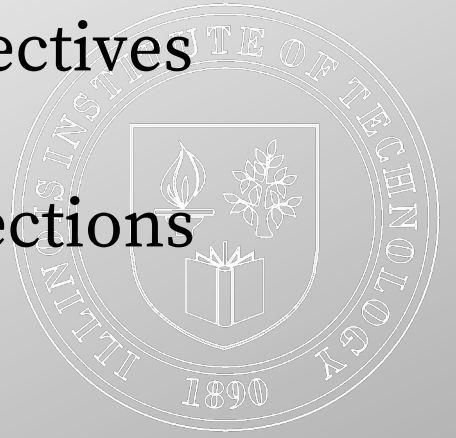
3. What are the potential benefits from using tools in general to support testing?

a. Greater quality of code, reduction in the number of testers needed, better objectives for testing.

b. Greater repeatability of tests, reduction in repetitive work, objective assessment.

c. Greater responsiveness of users, reduction of tests run, objectives not necessary.

d. Greater quality of code, reduction in paperwork, fewer objections to the tests.



Chapter 6 – 4. Questions and Answers

4. What is a potential risk in using tools to support testing?

a. Unrealistic expectations, expecting the tool to do too much.

b. Insufficient reliance on the tool, i.e. still doing manual testing when a test execution tool has been purchased.

c. The tool may find defects that aren't there.

d. The tool will repeat exactly the same thing it did the previous time.

Chapter 6 – 5. Questions and Answers

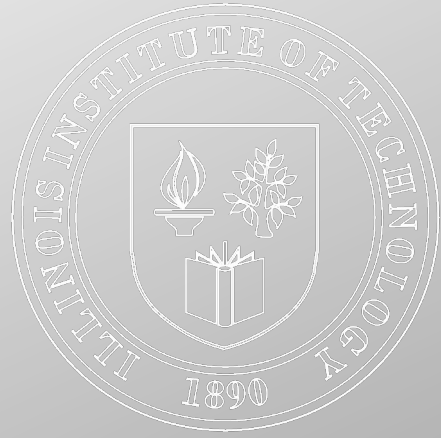
5. Which of the following are advanced scripting techniques for test execution tools?

a. Data-driven and keyword-driven.

b. Data-driven and captured-driven.

c. Capture-driven and keyhole-driven.

d. Playback-driven and keyword-driven.



Chapter 6 – 6. Questions and Answers

6. Which of the following would NOT be done as part of selecting a tool for an organization?

a. Assess organizational maturity, strengths, and weaknesses.

b. Roll out the tool to as many users as possible within the organization.

c. Evaluate the tool features against clear requirements and objective criteria.

d. Identify internal requirements for coaching and mentoring in the use of the tool.

Chapter 6 – 7. Questions and Answers

7. Which of the following is a goal for a proof-of-concept or pilot phase for tool evaluation?

a. Decide which tool to acquire.

b. Decide on the main objectives and requirements for this type of tool.

c. Evaluate the tool vendor including training, support, and commercial aspects.

d. Decide on standard ways of using, managing, storing, and maintaining the tool and the test assets.