Software validation and Testing

Belaynesh C.

Software Test Life Cycle (STLC)

1. Test planning: what, when and how to test.

• Test leaders and test managers use a project plan and functional requirements to produce test plan document. Some of the activities are resource identification, test estimation and preparation of test plans.

2. Test Designing:

• Most of test design activities are done by test engineers with some involvement of a team leader. The test plan obtained from the previous stage is used along with Project plan and functional requirements to produce test case documents, test scenario and traceability matrix (To demonstrate mapping between test cases with test scenarios).

3. Test Execution:

- Like test design, most tasks in this phase are done by test engineers. Previously obtained documents such as functional requirements, test plan and test cases along with program builds (software pre-release version) from programmers are considered as major resources to use at this specific phase.

- Important tasks are test case execution, test report/ test log preparation and defect identification. Test reports are obtained as a result.

Test Report is a document which contains a summary of all test activities and final test results of a testing project.

Test report is an assessment of how well the Testing is performed.

Based on the test report, stakeholders can evaluate the quality of the tested product and make a decision on the software release.

Test Report

Test Cycle System Test

EXECUTED	PASSED			130	
	FAILED			0	
	(Total) TESTS EXECUTED (PASSED + FAILED)				**130**
PENDING					0
IN PROGRESS					0
BLOCKED					0
(Sub-Total) TEST PLANNED					**130**
(PENDING + IN PROGRESS + BLOCKED + TEST EXECUTED)					

Functions	Description	% TCs Executed	% TCs Passed	TCs pending	Priority	Remarks
New Customer	Check new Customer is created	100%	100%	0	*High*	
Edit Customer	Check Customer can be edited	100%	100%	0	*High*	
New Account	Check New account is added	100%	100%	0	*High*	
Edit Account	Check Account is edit	100%	100%	0	*High*	
Delete Account	Verify Account is delete	100%	100%	0	*High*	
Delete customer	Verify Customer is Deleted	100%	100%	0	*High*	
Mini Statement	Verify Ministatement is generated	100%	100%	0	*High*	
Customized Statement	Check Customized Statement is generated	100%	100%	0	*High*	

4. Defect Reporting and Tracking:

• Test cases and test logs are used as inputs for this stage. Test engineers prepare defect reports and report defects to programmers.

5. Test closure:

• Test reports and defect reports are analyzed and exit criteria are also evaluated. Most of these tasks are handled by test lead or test managers.

Test Plan

- The test plan documents give information about scope of testing (software sections to be tested), resources required (hardware, software and human resources) , types of testing to be carried out, automation strategies if any, risk management, entry and exit criteria.

1. Scope: for example, a specific module or group of modules,
2. Hardware: devices required such as servers
3. Software: operating systems
4. Type of testing: functional and non-functional testing

Test plan as per IEEE 829-20083

1. Introduction
1.1. Document identifier
1.2. Scope
1.3. References
1.4. Level in the overall sequence
1.5. Test classes and overall test conditions
2. Details for this level of test plan
2.1. Test items and their identifiers
2.2. Test traceability matrix
2.3. Features to be tested
2.4. Features not to be tested
2.5. Approach
2.6. Item pass/fail criteria

2.7. Suspension criteria and resumption requirements

2.8. Test deliverables

3. Test management

3.1. Planned activities and tasks; test progression

3.2. Environment/infrastructure

3.3. Responsibilities and authority

3.4. Interfaces among the parties involved

3.5. Resources and their allocation

3.6. Training

3.7. Schedules, estimates, and costs

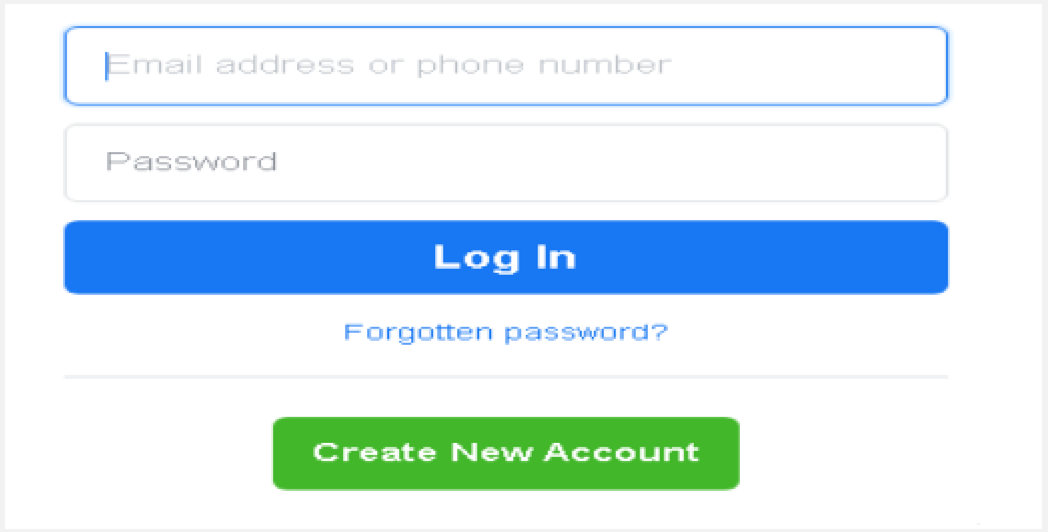
3.8. Risk(s) and contingency(s)

General
4.1. Quality assurance procedures
4.2. Metrics
4.3. Test coverage
4.4. Glossary
4.5. Document change procedures and history Templates and Models

Test Design

- Describe a given functionality in terms of possible input types expected from users.

- For example, look at a section from Facebook's home page.



Test scenarios

- The functionalities are the login, forgotten password and create new accounts. The login functionality can be tested with the following test scenarios.

- Test with valid email address/phone number and password .
- Test with *invalid* email address/phone number and *valid* password.
- Test with *valid* email address/phone number and *invalid* password.
- Test with *invalid* email address/phone number and *invalid* password.
- Test with blank email address/phone number and password.

For each test scenario, a test case is designed. The test case document contains details such as test case Id, project/module, Requirement number, scenario, test case title, steps to be followed, test data, expected results, actual results and finally the status of the test.

Why documenting test cases?

- **Track Test Coverage:** It assists the team to efficiently track and monitor the testing process and ensure 100% coverage. Also, it enables them to identify any deviances and perform effective requirement tests.
- **Maintain Consistency:** By preparing test case document during the early stages of testing, the team can maintain consistency throughout the testing process.
- **Re-usability:** Test case documents that are characterized by high accuracy, sufficient and relevant information, can be used by other testers to tackle similar issues and defects.

Easy to Automate: Test cases can be utilized to create automated test scripts. Moreover, these are easily automated, which allows the team to use them with any automated tool, such as test management tool or test case management tool.

- **Share Knowledge:** These can also be used by the team to share their knowledge and strategies of testing with their team as well as the stakeholders of the project.
- **Verification & Validation:** Most importantly, with the assistance of test case document, the team is able to verify the various features and aspects of the software product and validate their compliance against the specified requirements.

Test design document as per IEEE 829-2008

1. Introduction
1.1. Document identifier
1.2. Scope
1.3. References
2. Details of the level test design
2.1. Features to be tested
2.2. Approach refinements
2.3. Test identification
2.4. Feature pass/fail criteria
2.5 Test deliverables
3. General
3.1. Glossary

3.2. Document change procedures and history

Test case document as per IEEE 829-2008

1. Introduction (once per document)
1.1. Document identifier
1.2. Scope
1.3. References
1.4. Context
1.5. Notation for description
2. Details (once per test case)
2.1. Test case identifier
2.2. Objective
2.3. Inputs
2.4. Outcome(s)
2.5. Environmental needs
2.6. Special procedural requirements
2.7. Inter case dependencies
3. Global (once per document)
3.1. Glossary
3.2. Document change procedures and history

Test Execution

Test procedure document as per IEEE 829-20089

1. Introduction
1.1. Document identifier
1.2. Scope
1.3. References
1.4. Relationship to other procedures
2. Details
2.1. Inputs, outputs, and special requirements
2.2. Ordered description of the steps to be taken to execute the test cases
3. General
3.1. Glossary

3.2. Document change procedures and history

Test log as per IEEE 829-2008

1. Introduction
1.1. Document identifier
1.2. Scope
1.3. References
2. Details
2.1. Description
2.2. Activity and event entries
3. General
3.1. Glossary

Defect report as per IEEE 829-200813

1. Introduction
1.1. Document identifier
1.2. Scope
1.3. References
2. Details
2.1. Summary
2.2. Date anomaly discovered
2.3. Context
2.4. Description of anomaly
2.5. Impact
2.5.1. Inputs
2.5.2. Expected results
2.5.3. Actual results

2.5.4. Unexpected outcomes
2.5.5. Procedure step
2.5.6. Environment
2.5.7. Attempts to repeat
2.5.8. Testers
2.5.9. Observers
2.6. Originator’s assessment of urgency
2.7. Description of the corrective action
2.8. Status of the anomaly
2.9. Conclusions and recommendations
3. General
3.1 Document change procedures and history

Software Quality Attributes

While buying a mobile phone what are the quality attributes that you would consider?

Categories of software quality attributes

➢ *Functional attributes:* characterize the input/output behavior of the software product.
Correctness, robustness, reliability and safety

➢ Operational attributes: characterize the operational conditions of the software product

Latency: (or response time),

Throughput: Throughput is the volume of processing that the system can deliver per unit of operation time.

Efficiency: The efficiency of a software product is its ability to deliver its functions and services with minimal computing resources.

- Usability attributes: characterize the extent to which the software product can be used and adapted to user needs (*Ease of Use*, *Ease of Learning)*
- Business attributes: characterize the cost of developing, using, and evolving the software product(*Development Cost*, *Maintainability*, *Portability)*
- Structural attributes: characterize the internal structure of the software product (Testability, modularity, adaptability)

Software Testing tools

→Software testing is a labor-intensive task that is difficult to automate because it depends on a great deal of creativity and because it is difficult to cast in a systematic process.

→Still, there is much scope for automated support, to help with clerical or repetitive aspects of software testing.

→Test level: unit testing, integration testing, system testing, acceptance testing.

→Test data generation method: specification-based, code-based, scenario-based.

→Testing phase/activity: test data generation, test driver design, test execution, test outcome analysis.

→Target language: some tools are restricted to specific languages or are geared toward programs written in a specific language.

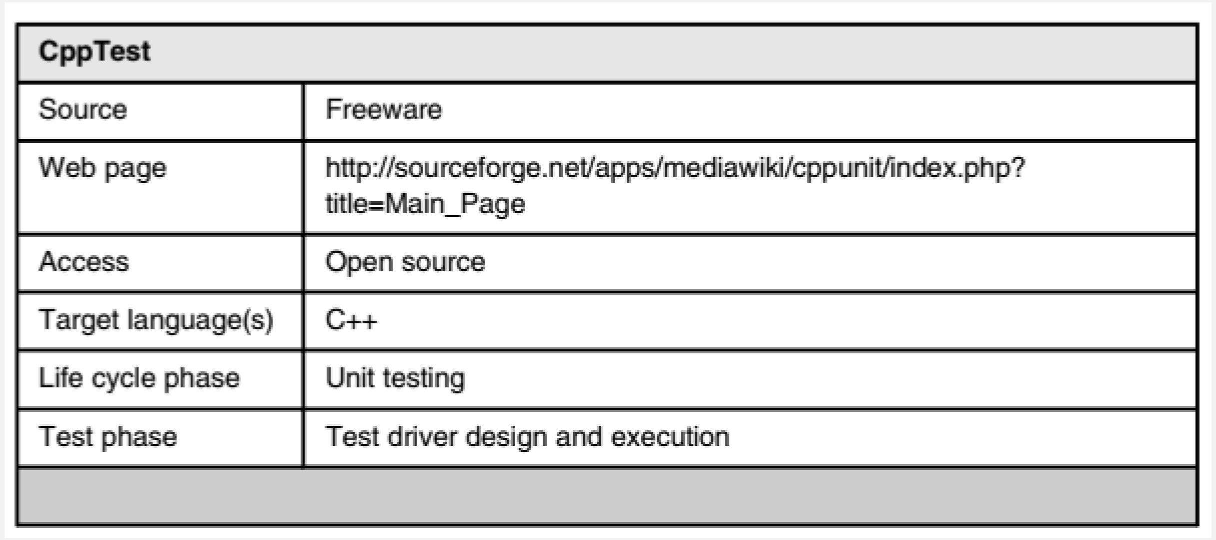
→Target development environment: some tools are compatible with a limited set of development environments.

1.Scripting Tools, which enable us to codify and store test data and test execution traces for repeated use; these tools are useful to test an application after each fault removal (in corrective maintenance, for example); they are not directly useful in adaptive maintenance because they cannot be reused if the specification changes.

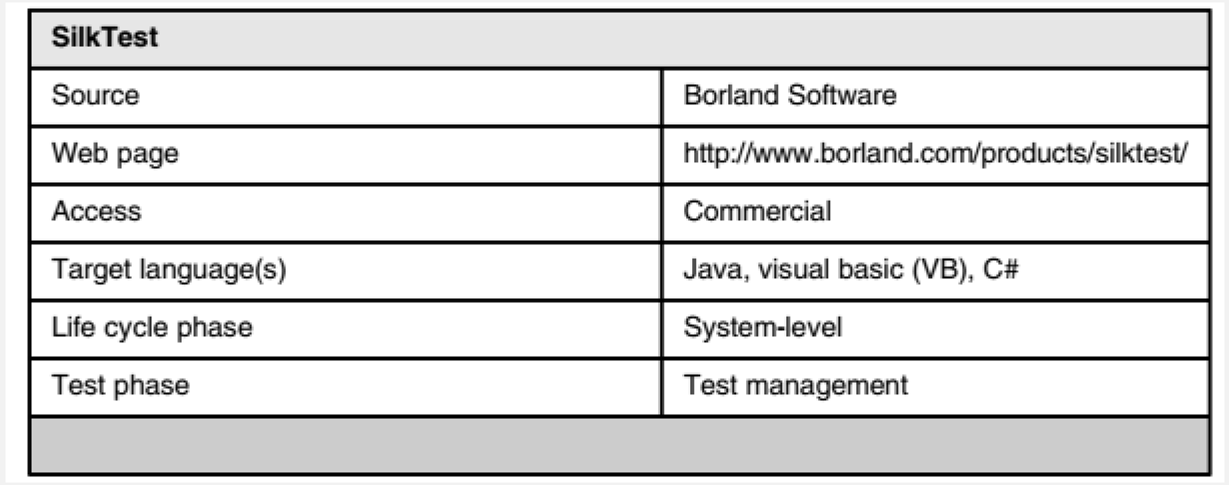
2. Record-and-Replay Tools, which can be used in applications with a complex interface (such as web applications) to record the scenario of interactions of the tester with the application. This scenario can be replayed whenever we want to reexecute the test (e.g., after each corrective maintenance operation to retest the application).

3. Performance-Testing Tools, which simulate workload conditions for applications under test to check how their performance evolves according to their workload or how they behave under specific workload conditions.
4. Oracle Design Tools, which enable the tester to write code to codify the oracle of a test for the sake of automation.
5. Exception Discovery Tools, which alert the tester to the possibility that the program may run into an exceptional condition, such as an infinite loop, an overflow/underflow, an array reference out of bounds, an illegal pointer reference, or a division by zero.
6. Collaborative Tools, which are tools whose main function is to serve as communication media between the stakeholders of a test, including end users, developers, and testers.

Scripting tools

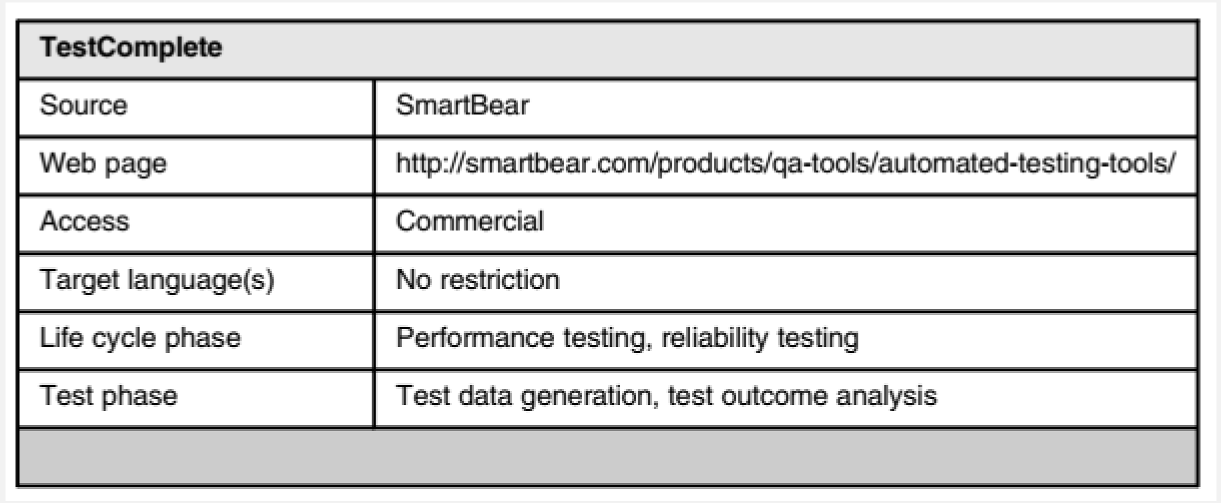


CppTest	
Source	Freeware
Web page	http://sourceforge.net/apps/mediawiki/cppunit/index.php?title=Main_Page
Access	Open source
Target language(s)	C++
Life cycle phase	Unit testing
Test phase	Test driver design and execution

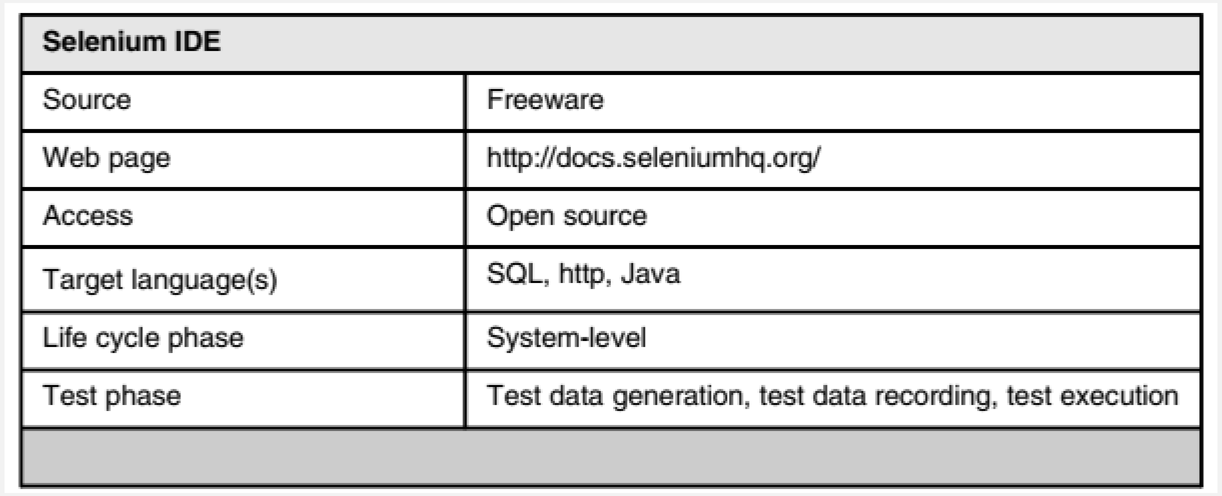


SilkTest	
Source	Borland Software
Web page	http://www.borland.com/products/silktest/
Access	Commercial
Target language(s)	Java, visual basic (VB), C#
Life cycle phase	System-level
Test phase	Test management

RECORD-AND-REPLAY TOOLS

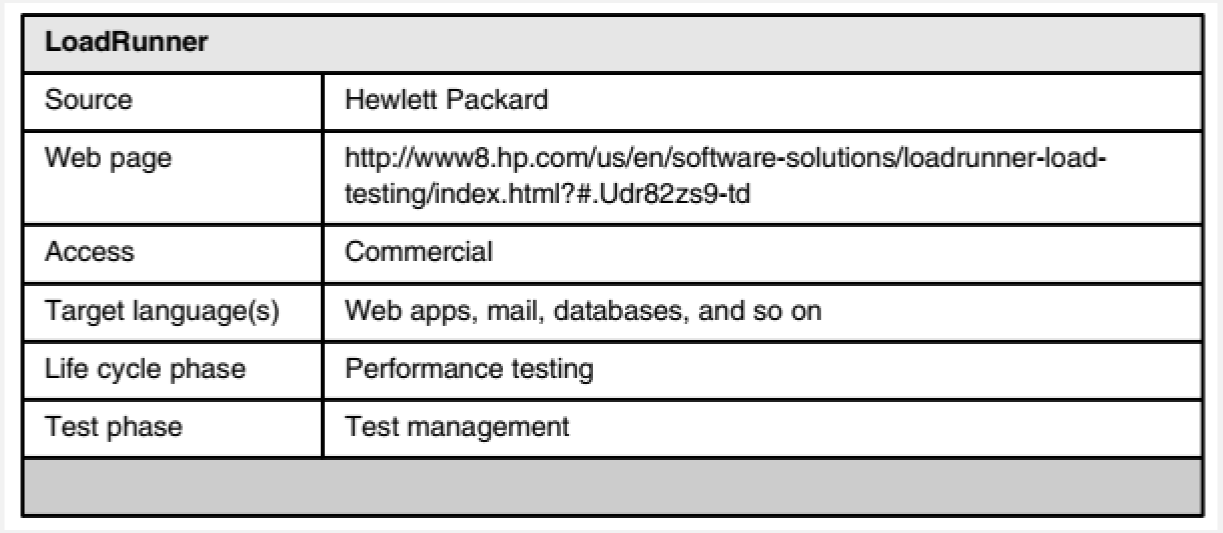


TestComplete	
Source	SmartBear
Web page	http://smartbear.com/products/qa-tools/automated-testing-tools/
Access	Commercial
Target language(s)	No restriction
Life cycle phase	Performance testing, reliability testing
Test phase	Test data generation, test outcome analysis

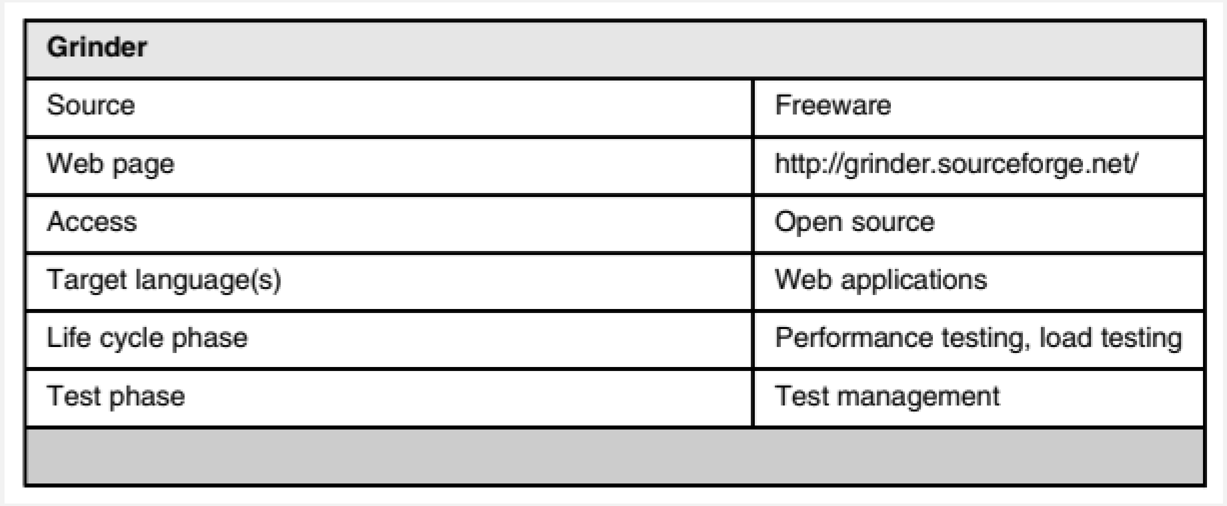


Selenium IDE	
Source	Freeware
Web page	http://docs.seleniumhq.org/
Access	Open source
Target language(s)	SQL, http, Java
Life cycle phase	System-level
Test phase	Test data generation, test data recording, test execution

PERFORMANCE-TESTING TOOLS

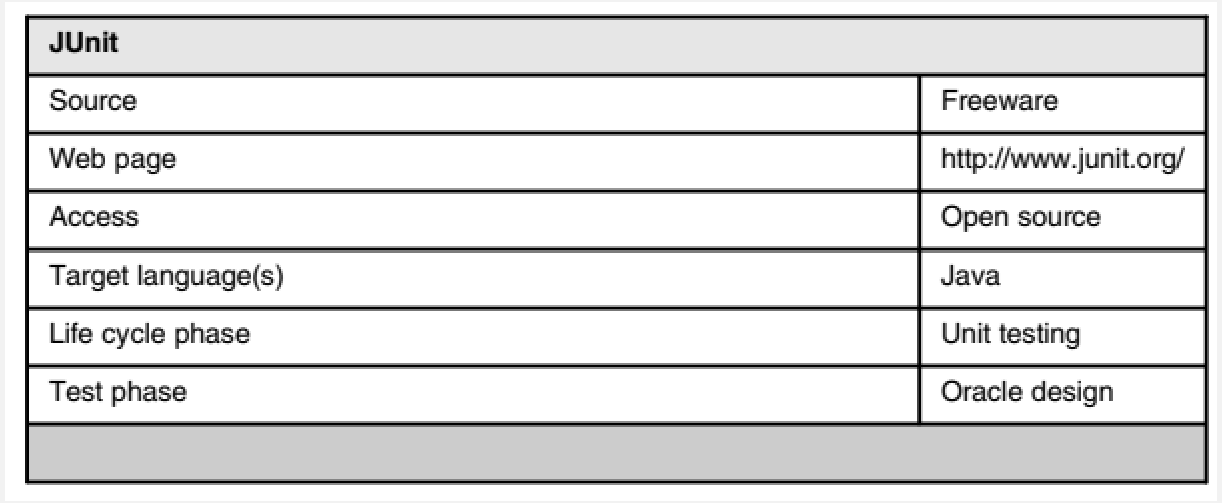


LoadRunner	
Source	Hewlett Packard
Web page	http://www8.hp.com/us/en/software-solutions/loadrunner-load-testing/index.html?#.Udr82zs9-td
Access	Commercial
Target language(s)	Web apps, mail, databases, and so on
Life cycle phase	Performance testing
Test phase	Test management

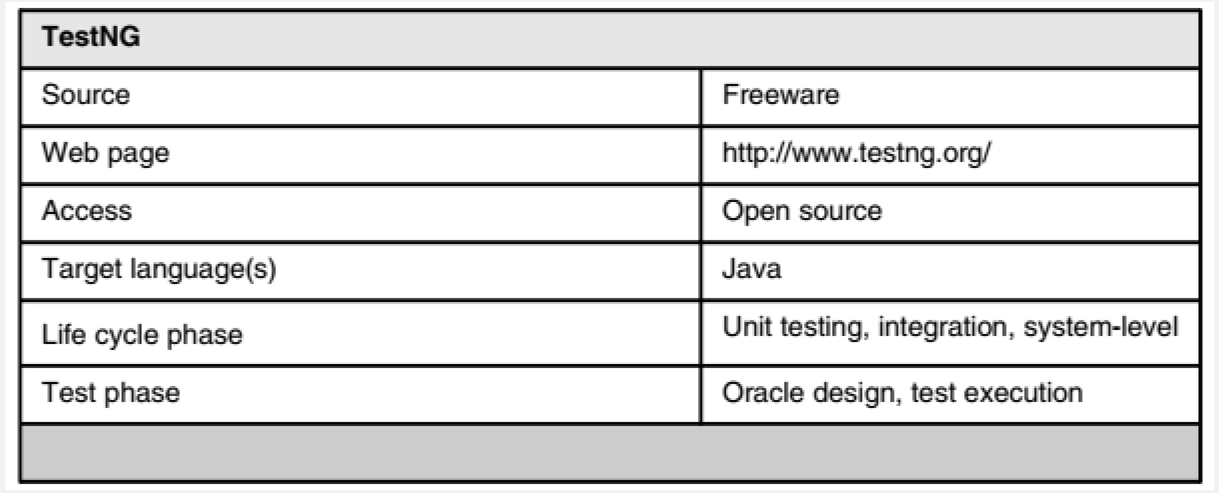


Grinder	
Source	Freeware
Web page	http://grinder.sourceforge.net/
Access	Open source
Target language(s)	Web applications
Life cycle phase	Performance testing, load testing
Test phase	Test management

ORACLE DESIGN TOOLS

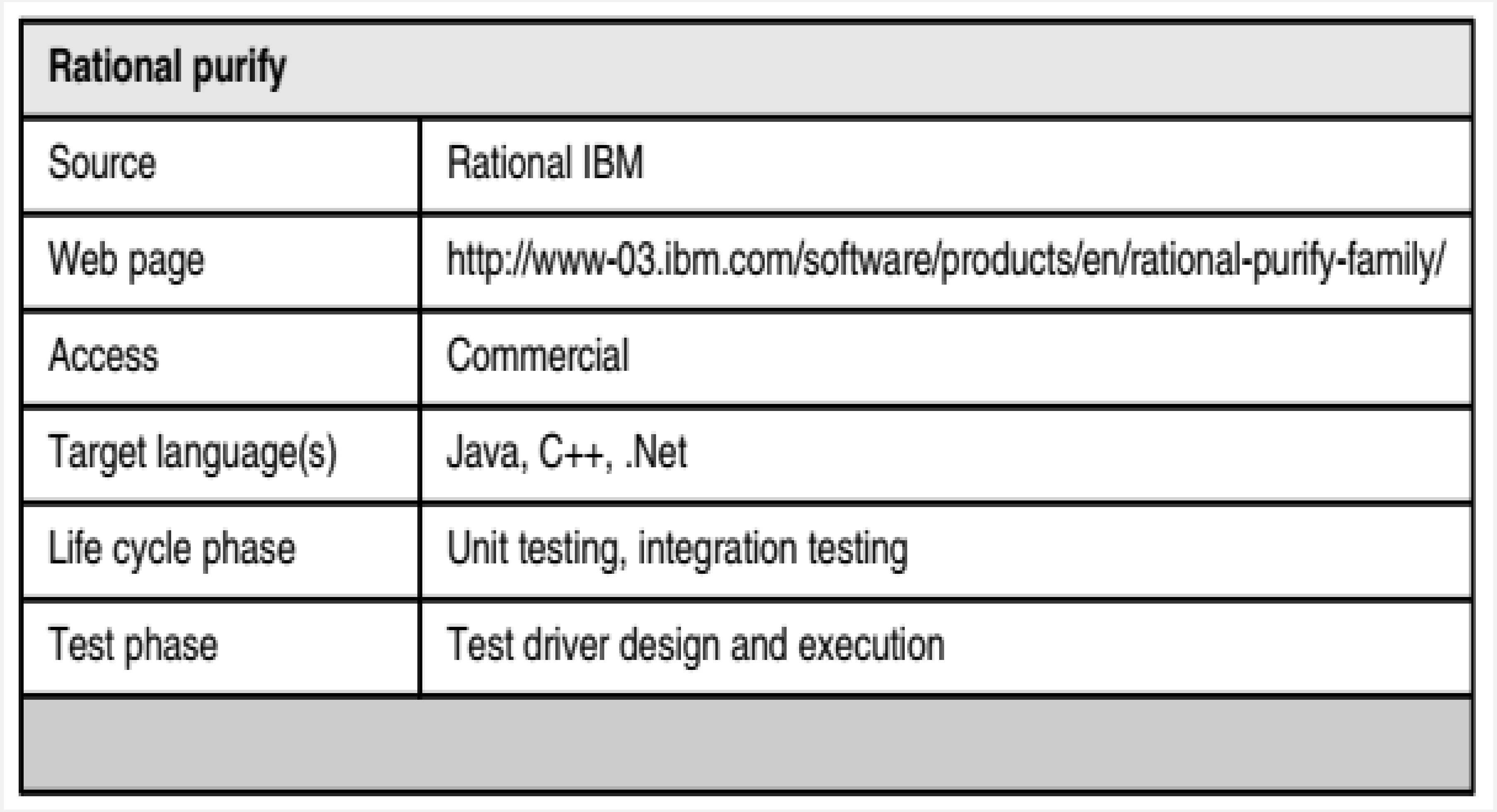


JUnit	
Source	Freeware
Web page	http://www.junit.org/
Access	Open source
Target language(s)	Java
Life cycle phase	Unit testing
Test phase	Oracle design



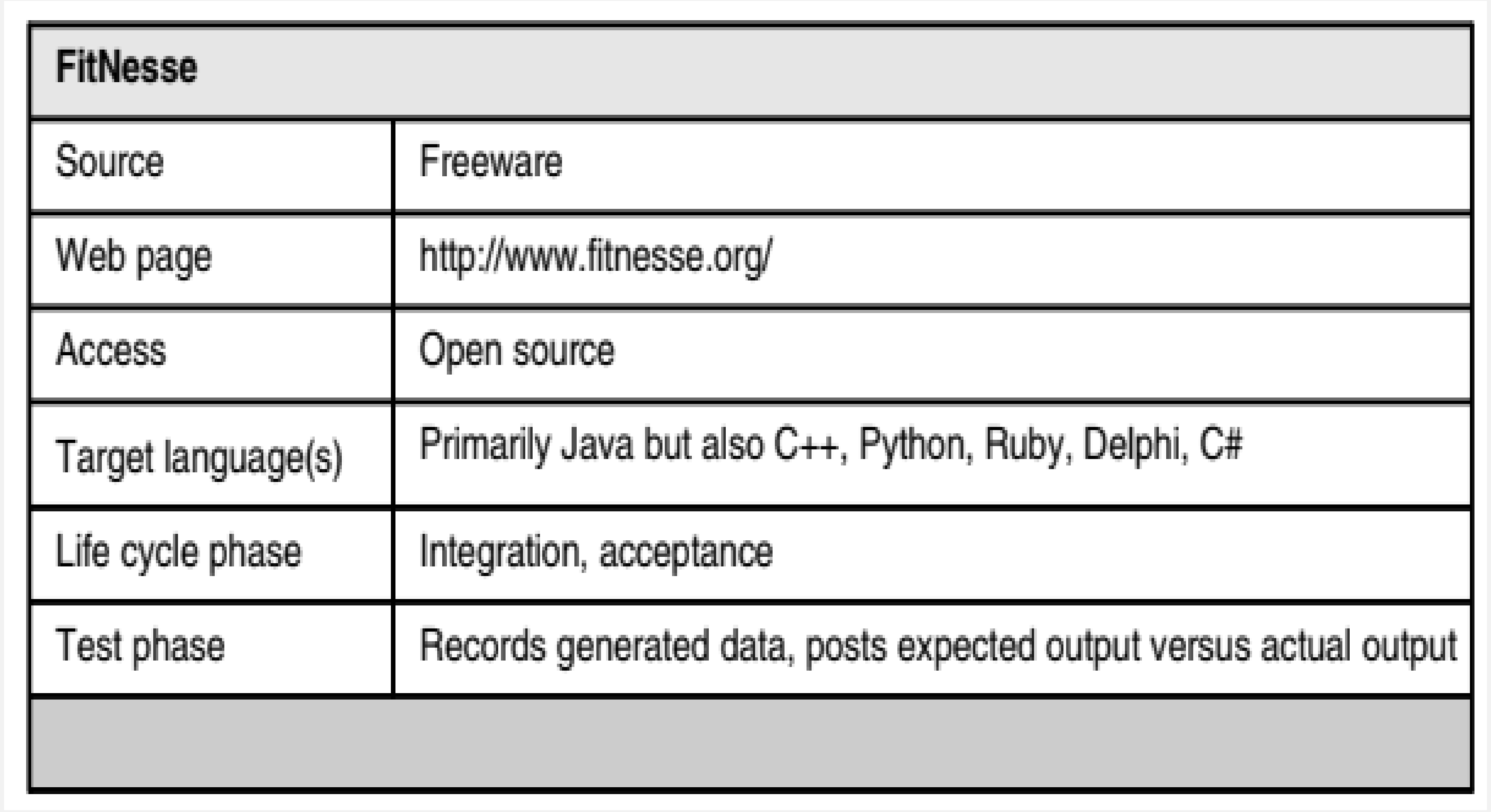
TestNG	
Source	Freeware
Web page	http://www.testng.org/
Access	Open source
Target language(s)	Java
Life cycle phase	Unit testing, integration, system-level
Test phase	Oracle design, test execution

EXCEPTION DISCOVERY



Rational purify	
Source	Rational IBM
Web page	http://www-03.ibm.com/software/products/en/rational-purify-family/
Access	Commercial
Target language(s)	Java, C++, .Net
Life cycle phase	Unit testing, integration testing
Test phase	Test driver design and execution

COLLABORATIVE TOOLS



FitNesse	
Source	Freeware
Web page	http://www.fitnesse.org/
Access	Open source
Target language(s)	Primarily Java but also C++, Python, Ruby, Delphi, C#
Life cycle phase	Integration, acceptance
Test phase	Records generated data, posts expected output versus actual output