ISSN : 2231-0711

Ananya Shrivastava et al | International Journal of Computer Science Engineering and Technology(IJCSET) | July 2014 | Vol 4, Issue 7,217-222

Performance Testing and Improvement in Agile

Ananya Shrivastava
Research scholar
CSE Department
S.V.I.T.S., Indore, India

Dr. Dinesh C. Jain
Reader
CSE Department
S.V.I.T.S., Indore, India

Abstract- There is always scope for performance improvement in any application. But, how do we know where and how to improve?
This paper is meant to equip social control and technical persons with an end to end understanding and knowhow to implement performance testing of web applications concerning their project requirements. It explores and helps in deciding how and when a performance test is beneficial. From basic principles to regard to Agile platform to the sum of implementation with examples constitutes this paper. We have our dark areas for improvements. We pin pointed those areas. Now, how do we improve on those areas? How do we manage this whole performance improvement through process? Get on the interesting and information rich text read what awaits ahead!

Keyword: Performance Testing, Agile platform, Project requirement.

I. INTRODUCTION

Performance testing is a type of testing intended to determine the responsiveness, throughput, responsible-ness, and quantifiability of a system below a given workload. Performance testing is commonly conducted to accomplish the following:

- Assess production readiness
- Evaluate against performance criteria
- Compare performance characteristics of multiple systems or system configurations
- Notice the supply of performance issues
- Support system tuning
- Find throughput levels

II. PERFORMANCE MODEL

There are various well-known methodologies available for performance evaluation of computer systems. The overall performance of any system mainly depends on three models

- Workload model
- Performance model
- Cost model

➢ **Workload model**
The workload model captures the resource demands and workload intensity characteristics of the load brought to the system by the different types of transactions and requests.

➢ **Performance model**
The performance model is employed to predict response times, utilization, and throughputs, as a function of the system description and workload parameters.

➢ **Cost model**
The cost model accounts for software package, hardware, network-communications and support expenditures

III. WHY DO PERFORMANCE TESTING?

At the best level, performance testing is nearly continually conducted to handle one or additional risks associated with expense, chance prices, continuity, and/or company name. Specific reasons for conducting performance testing include:

- ***Assessing adequacy of developed software package performance by:***
  - Determining the application's desired performance characteristics (mar holf dozen) before and once changes to the software package.
  - Providing comparisons between the application's current and desired performance characteristics.
- ***Improving the potency of performance standardization:***
  - Analysing the behaviour of the applying at varied load levels.
  - Identifying bottlenecks within the application.
  - Providing info associated with the speed, quantifiability, and stability of a product prior to production release, so sanctioning you to make informed decisions about whether and when to tune the system

IV. PROJECT CONTEXT

For a performance testing project to achieve success, each the approach to testing performance and also the testing itself must be relevant to the context of the project. Performance testing is bound to focus on only those items that are relevant for achieving overall project success. This might embody, however is not restricted:

- The general vision or intent of the project
- Performance testing objectives
- Performance success criteria
- The project schedule
- The project budget
- Available tools and environments
- The ability set of the performance tester and also the team
- The V-shaped model should be utilized for small to medium sized projects where requisites are not vaguely defined and is fixed.
- The V-Shaped model should be opted when ample technical resources are present with needed technical expertise.

High confidence of customer is required for opting the V-Shaped model approach. Since, no prototypes are engendered, there is a very high peril involved in meeting customer expectations.

V. CORE ACTIVITIES OF PERFORMANCE TESTING

The performance testing approach consists of the following activities:

(i) Activity 1. Identify the Test Environment.

Identify the physical test environment and the production environment as well as the tools and resources available to the test team. The physical environment includes hardware, software, and network configurations. Having a radical understanding of the complete check setting at the outset enables more efficient test design and planning and helps you identify testing challenges early in the project. In some situations, this method should be revisited periodically throughout the project's life cycle.

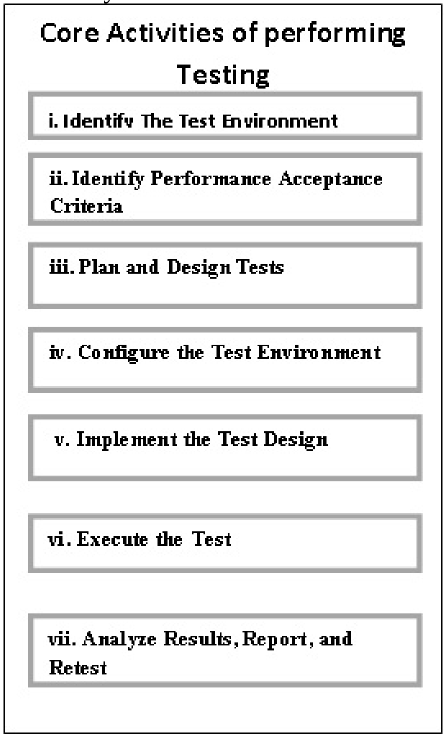

Core Activities of performing Testing
i. Identify The Test Environment
ii. Identify Performance Acceptance Criteria
iii. Plan and Design Tests
iv. Configure the Test Environment
v. Implement the Test Design
vi. Execute the Test
vii. Analyze Results, Report, and Retest

Fig:1 Core Activities Of Performing Testing

(ii) Activity 2. Identify Performance Acceptance Criteria.

Identify the response time, throughput, and resource utilization goals and constraints. In general, response time is a user concern, throughput is a business concern, and resource utilization is a system concern. Additionally, identify project success criteria that may not be captured by those goals and constraints; for example, using performance tests to evaluate what combination of configuration settings will result in the most desirable performance characteristics.

(iii) Activity 3. Plan and Design Tests.

Identify key scenarios, determine variability among representative users and how to simulate that variability, define test data, and establish metrics to be collected. Consolidate this information into one or more models of system usage to be implemented, executed, and analysed.

(iv). Activity 4. Configure the Test Environment.

Prepare the test environment, tools, and resources necessary to execute each strategy as features and components become available for test. Ensure that the test environment is instrumented for resource monitoring as necessary.

(v). Activity 5. Implement the Test Design.

Develop the performance tests in accordance with the test design.

(vi). Activity 6. Execute the Test.

Run and monitor your tests. Validate the tests, test data, and results collection. Execute validated tests for analysis while monitoring the test and the test environment.

(vii). Activity 7. Analyse Results, Report, and Retest.

Consolidate and share results data. Analyze the data both individually and as a cross-functional team. Reprioritize the remaining tests and re-execute them as needed. When all of the metric values are within accepted limits, none of the set thresholds have been violated, and all of the desired information has been collected, you have finished testing that particular scenario on that particular configuration.

VI. KEY TYPES OF PERFORMANCE TESTING

(i) Performance test:

- ➢ A performance test is a technical investigation done to determine or validate the responsiveness, speed, scalability, and/or stability characteristics of the product under test.
- ➢ To determine or validate speed, scalability, and/or stability

(ii) Load Test:

- ➢ Load testing is conducted to verify that your application can meet your desired performance objectives; these performance objectives are often specified in a service level agreement (SLA). A load test enables you to measure response times, throughput rates, and resource-utilization levels, and to identify your application's breaking point, assuming that the breaking point occurs below the peak load condition.
- ➢ Endurance testing is a subset of load testing. An endurance test is a type of performance test focused on determining or validating the performance characteristics of the product under test when subjected to workload models and load volumes anticipated during production operations over an extended period of time.
- ➢ To verify application behaviour under normal and peak load conditions.

(iii) Stress test:

- ➢ The goal of stress testing is to reveal application bugs that surface only under high load conditions. These bugs can include such things as synchronization issues, race conditions, and memory leaks. Stress testing enables you to identify your application's weak points, and shows how the application behaves under extreme load conditions.

- Spike testing is a subset of stress testing. A spike test is types of performance test focused on deter min-ing or validating the performance characteristics of the product under test when subjected to workload models and load volumes that repeatedly increase beyond an-ticipated production operations for short periods of time.
- To determine or validate an application's behaviour when it is pushed beyond normal or peak load condi-tions.

(iv)Capacity test:

- Capacity testing is conducted in conjunction with ca-pacity planning, which you use to plan for future growth, such as an increased user base or increased volume of data. For example, to accommodate future loads, you need to know how many additional re-sources (such as processor capacity, memory usage, disk capacity, or network bandwidth) are necessary to support future usage levels.
- Capacity testing helps you to identify a scaling strategy in order to determine whether you should scale up or scale out.
- To determine how many users and/or transactions a given system will support and still meet performance goals.

VII. AGILE PERFORMANCE TEST CYCLE

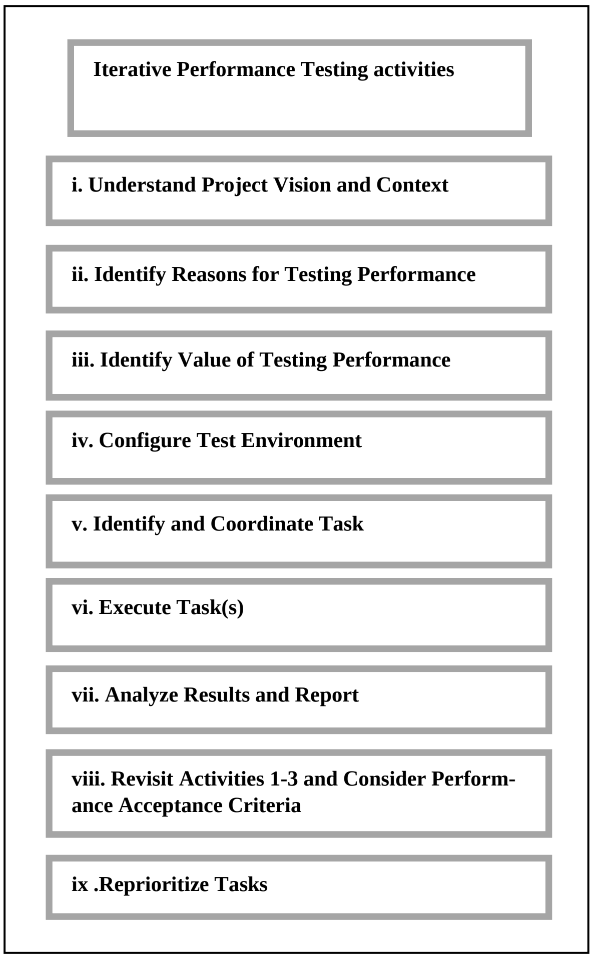

Iterative Performance Testing activities
i. Understand Project Vision and Context
ii. Identify Reasons for Testing Performance
iii. Identify Value of Testing Performance
iv. Configure Test Environment
v. Identify and Coordinate Task
vi. Execute Task(s)
vii. Analyze Results and Report
viii. Revisit Activities 1-3 and Consider Perform-ance Acceptance Criteria
ix .Reprioritize Tasks

Fig:2 Agile Performance Test Cycle

- **Activity 1. Understand the Project Vision and Con-text**.
  The outcome of this activity is a shared understanding of the project vision and context.
- **Activity 2. Identify Reasons for Testing Perform-ance**.
  Explicitly identify the reasons for performance testing.
- **Activity 3. Identify the Value Performance Testing Adds to the Project**.
  Translate the project- and business-level objectives into specific, identifiable, and manageable perform-ance-testing activities.
- **Activity 4. Configure the Test Environment**.
  Set up the load-generation tools and the system under test, collectively known as the performance test envi-ronment.
- **Activity 5. Identify and Coordinate Tasks**.
  Prioritize and coordinate support, resources, and schedules to make the tasks efficient and successful.
- **Activity 6. Execute Task(s)**.
  Execute the activities for the current iteration.
- **Activity 7. Analyze Results and Report**.
  Analyze and share results with the team.
- **Activity 8. Revisit Activities 1-3 and Consider Per-formance Acceptance Criteria**.
  Between iterations, ensure that the foundational in-formation has not changed. Integrate new information such as customer feedback and update the strategy as necessary.
- **Activity 9. Reprioritize Tasks**.
  Based on the test results, new information, and the availability of features and components, reprioritize, add to, or delete tasks from the strategy, and then re-turn to activity 5.

VIII. PERFORMANCE TESTING USING VISUAL STUDIO ULTIMATE 2012

Following the generic process and principals described above we can have a more specific flow of actions to be followed while conducting performance testing activity using Visual Studio Ultimate. Just like a very simple tool like HTTP Watch VS records the HTTP requests sent using methods like GET or POST. What VSTS offers is an auto-mated and end to end testing experience with further con-figurability and extensibility. Details are given below:

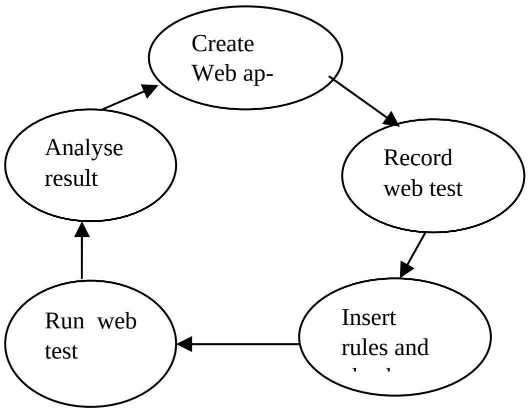


Fig:3 Process of Web testing

(i) Recording a Web Test

VSTS Web Testing supports activity recording, as long as there exists an internet site to record against. To record activities of an internet application, we are able to produce a brand new take a look at project or use an existing one. Adding a web test case is more specific to web testing. When it is added, an activity recorder is started in which we can browse the web application like browsing through a normal browser.

On the address bar, we can enter the URL of the personal website, including the port selected by the ASP.NET Development Server. Browsing to this location will be recorded in the Web Test Recorder explorer bar; as would any other URLs that are entered. Once the desired tests have been recorded, we can close the browser windows and save the test. The project will automatically include the Web test case file along with each of the recorded requests.

(ii)Customization

Selecting any of the nodes within the Web Test tree will allow you to modify the data inside the Properties window.

You can also group requests together using a transaction. Be careful not to confuse the term "transaction" with a programming concept in which state is committed or not committed as a unit. Within Web test cases, transactions only encapsulate actions into a group that can later be enumerated using code. Also, transactions are used when reporting on load—how many transactions per second, for example.

(iii)Using the Web Test Viewer to Verify a Web Test

Before adding a Web test to a load test, and running it for a long period of time, it is important to be sure that the test works exactly as intended. This is where the Web test viewer comes in to consideration. The Web test viewer allows you to watch a Web test as it runs, and to view all aspects of a previous test run.

Verifying a newly-created Web test goes beyond looking at the outcome of the test run and seeing whether it passed. For example, for a Web test without validation rules passed, means that no exceptions were thrown, no rules failed, and no HTTP errors occurred. Verification includes making sure the Web test exhibits the correct behavior on the target Web application, in addition to executing without any errors. It is important to review the response for each request to make sure that it is correct.

(iv)Running a Web Test Case

After recording a test you are ready to begin executing it. To execute all the tests within a project, simply run the project. This will open up the Test Results windows and mark each test as pending while it is in progress, and Passed/Failed once execution completes. Test selection and execution is also available from the Test Manager and Test View windows.

Individual Web test files (test cases or test fixtures) can also be run by opening them up and clicking the Run button.

Requests can also provide credentials for logging on to the targeted site using standard authentication methods. The dialogs for credentials allow for loading the login data from a data source.

Each result from a request is saved, and selecting each request allows you to navigate its detail, viewing the resulting page's HTML or raw request/response text.

As part of a test execution, the Web test engine verifies if all the URLs on the page are valid links. These links appear as child nodes below the request show the HTTP status returned by a request to each of the URLs.

(v)Request Rules

Although checking for valid hyperlinks on a response page is a useful feature, it is not sufficient in validating that the page is functioning correctly. For example, on entering valid credentials, the Web test needs to verify that the login was successful. Similarly, when the credentials are invalid, you need to check that an appropriate error message is displayed on the page. To support this, each Web request can include extraction rules and validation rules.

(vii)Binding Test Data to a Data Source(Parameterization)

Both validation and extraction rules provide for text entry within the Properties window of virtually every node. However, the fact that the text can be pulled from a database makes Visual Studio Web Test powerful. This means we can define a collection of inputs that do or don't conform to the specified requirements.

To test using a data source, we can click the Add Data Source button on the toolbar of the Web test. In the ensuing dialog box, specify an OLE DB Provider, perhaps using an *.mdf file that can also be added to the test project. After opening the database in the server explorer, we define a table that will contain the necessary test data.

Once a test has been configured with a data source, it is necessary to return to the Edit Run Settings dialog box and change the run count to one run per data source row. In this way, the test will repeat for each row in the newly configured data source, and during each run, the parameters associated with the data source will be assigned the value in the column for the particular row going to customize a test doing so provides a great starting point for customization of a particular Web test case, or even multiple cases, with a little refactoring.

IX. CONTINUOUS IMPROVEMENT AFTER TESTING

(i) Profilers:

After we have analysed test reports after the test runs for the performance we get to know where the performance is hit badly and needs improvements. What we get actually is the HTTP request turnaround time and reliability issues for a particular HTTP request which is a part of web test script. These requests translates to in part or whole to the scenarios that we have identified in the planning phase to be tested.

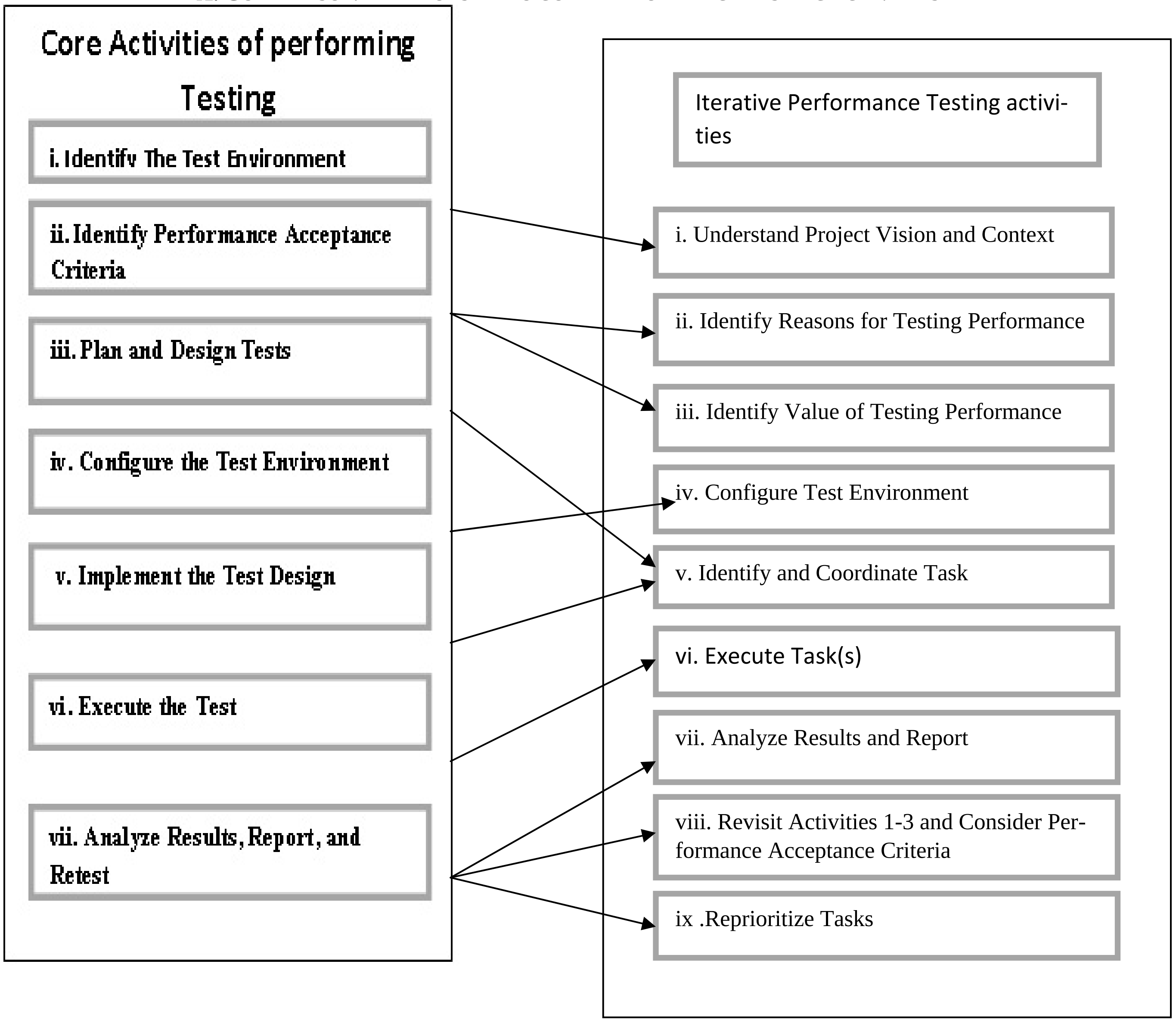
For the performance hit scenarios, there could be any reason: configuration of the application, network related issues, issue with core codebase, the APIs or the database querying and fetching results.

Profilers: There are third party Profilers for both database and code that can be employed to further zero down the cause of performance hit. Some platforms such as Visual Studio suit also provides inbuilt profilers for code part.

Hammer DB is another tool that can be employed to zero down the gap areas in the database.
Making use of these profilers Development team can quickly find and start improving on the code and database part.

(ii) Automated Builds:
Another action point that can be employed to further improve the productivity is to automate your triggering of web tests scenarios using a custom code piece. As evident with the case of Visual Studio, automated build definitions can be created to automatically trigger the web tests. This can further be integrated with every fresh build of your application and thus after analyzing the results from the test we have a better grip on what change is effecting the performance of the application.

X. Comparison/relationship to core performance testing activities



Core Activities of performing Testing
i. Identify The Test Environment
ii. Identify Performance Acceptance Criteria
iii. Plan and Design Tests
iv. Configure the Test Environment
v. Implement the Test Design
vi. Execute the Test
vii. Analyze Results, Report, and Retest
Iterative Performance Testing activities
i. Understand Project Vision and Context
ii. Identify Reasons for Testing Performance
iii. Identify Value of Testing Performance
iv. Configure Test Environment
v. Identify and Coordinate Task
vi. Execute Task(s)
vii. Analyze Results and Report
viii. Revisit Activities 1-3 and Consider Performance Acceptance Criteria
ix .Reprioritize Tasks

Fig:4 Relationship to core Performance activities

XI. Conclusion:

Web performance testing is a must for every application in almost all the cases where reliability, scalability and turnaround time is at stake. Not considering these parameters related to your application can translate to heavy losses in terms of brand and finances.

- The generic model of process to follow the performance testing is well co related to the new and widely popular Agile platforms.
- Web performance using Visual Studio not only gives the user an easy to use experience but also extended and configurability at their disposal.
- Various profilers can be employed to further zero down the dark areas for improvement. Integrated automated build definitions are further action points and advantage to finally get a near perfect application with respect to performance.

REFERENCES:

1. msdn Microsoft.com
2. Ron Patton ,Software Testing
3. Stackoverflow.com
4. Paul C.Jorgensen "Performance modle" Tim Koomen, Martin pol
5. http://www.testingreference.com/testinghistory.php Scott Loveland Geoffrey Miller, Richard Prewitt ,Michael Shannon
6. Marnie L. Hutcheson "Performing Testing"
7. vsptqrg.codeplex.comwww.xoriant.com/blog/type- performance-testing-and-key-issues "core activities of performing testing"
8. blog.nwcadence.com/web-performance-and geek-swithblogs.net/.../load-and-web-performance-testing-using-visual stload
9. Hp software developers Blogs"Agile Performance testing lifecycle " http://www.agileload.com/performance-testing/performance-testing-methodology/performance-testing-in-the-agile-process
10. Beck, Kent; et al. (2001). "Manifesto for Agile Software Develop-ment". Agile Alliance.Retrieved14 June 2010. ."Visual testing of software – Helsinki University of Technology"(PDF). Retrieved 2012-01-13"web test using visual studio"www.visualstudio.com/en-us/get-started/load...