

**DOKUZ EYLUL UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED
SCIENCES**

**RESEARCH OF MIND SHIFT IN TESTING
METHODOLOGIES AND APPROACHES IN THE
WORLD OF CLOUD TECHNOLOGIES**

**By
Tufan ERDİNÇ**

**March, 2013
İZMİR**

**RESEARCH OF MIND SHIFT IN TESTING
METHODOLOGIES AND APPROACHES IN THE
WORLD OF CLOUD TECHNOLOGIES**

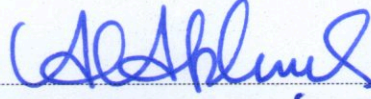
**A Thesis Submitted to the
Graduate School of Natural and Applied Sciences of Dokuz Eylul University
In Partial Fulfillment of the Requirements for the Degree of Master of Science
in Computer Engineering, Computer Engineering Program**

**by
Tufan ERDİNÇ**

**March, 2013
İZMİR**


M.Sc. THESIS EXAMINATION RESULT FORM

We have read the thesis entitled “RESEARCH OF MIND SHIFT IN TESTING METHODOLOGIES AND APPROACHES IN THE WORLD OF CLOUD TECHNOLOGIES” completed by TUFAN ERDİNÇ under supervision of ASST.PROF.DR. ADİL ALPKOÇAK and we certify that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

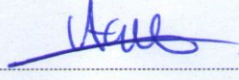


Asst. Prof. Dr. Adil ALPKOÇAK

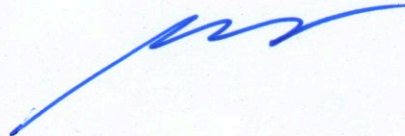
Supervisor



Y. Doc. Dr. Reyat YILMAZ
(Jury Member)



Prof. Dr. Aho KUT
(Jury Member)



Prof. Dr. Ayşe OKUR
Director
Graduate School of Natural and Applied Science

ACKNOWLEDGEMENTS

I would like to thank to my advisor Asst.Prof.Dr. Adil Alpköçak for his continuous help, suggestions, patience, guidance and support throughout all the phases of this thesis.

I would like to also thank my wife, and my two little sons, Emir and Ruzgar, who supported me all through this thesis preparation and for believing in me and being always with me.

Furthermore I would like to thank my family, my father, my sister who helped me registering for each semester, delivering documents and forms to school and being a power of attorney for me on these matters

Finally, I would like extend my sincere gratitude to all the staff and academic personnel of the Computer Engineering Department and Graduate School of Natural and Applied Sciences for their understandings and support in this, one of the longest lasting Master of Science degrees of the school history.

Tufan ERDİNÇ

RESEARCH OF MIND SHIFT IN TESTING METHODOLOGIES AND APPROACHES IN THE WORLD OF CLOUD TECHNOLOGIES

ABSTRACT

The cloud platform is getting more and more attractive to the computing world, enterprises and software developers. Existing software development frameworks and methodologies are being reshaped to fit this mind shift in computing world. Software development lifecycle activities are being morphed into actions that will support this mind shift. Testing tasks that are being placed within the software development methodologies should comply and support this new world of cloud computing.

This is intended to be a survey of existing test methodologies for cloud-based applications and services with pros and cons of each approach. We will touch base to cloud computing by telling the importance of cloud computing for enterprises, the types of cloud computing platforms used commonly and their attributes. Tools that can be used for cloud-based application testing are reviewed under different categories of testing. A sample Windows Azure application is also being tested with Visual Studio Web Test environment and Selenium Test Suite components to demonstrate the capabilities of these tools.

Finally, suggestions will be proposed in the conclusion chapter. New approaches such as Testing in Production (TiP), Synthetic Testing of Production code will be proposed as a better way to verify quality of cloud applications. Big enterprise software development companies, such as Microsoft is already integrating TiP methodologies into their cloud based service development cycles and this is suggested throughout the research.

Keywords: Cloud computing, Testing in cloud, Windows Azure, Visual Studio Web Testing, Selenium Testing Framework

BULUT BİLİŞİM TEKNOLOJİLERİNDE DEĞİŞEN TEST METODOLOJİLERİ VE YAKLAŞIMLARININ ARAŞTIRILMASI

ÖZ

Bulut bilişim platformları günümüz şirketleri ve yazılım geliştiricileri için gittikçe önem arz eden bir mahiyet kazanmaktadır. Var olan yazılım geliştirme yöntemleri ve kalıpları bu düşünce yapısına uygun olarak yeniden şekillenmektedir. Yazılım geliştirme hayat döngüleri bu yeni anlayışı destekleyecek biçimde değiştirilmekte ve uyarlanmaktadır. Yazılım geliştirme yöntemlerinin içerisinde geleneksel anlamda yer alan yazılım kalite kontrol uygulamaları da bu değişen ve yeniden şekillenen anlayışa göre hareket etmelidir.

Bu çalışmada, bulut bilişim tabanlı uygulamaların kalite kontrol yöntemlerinin ve yaklaşımlarının, her bir yaklaşımla elde edilen faydaların ve eksikliklerin incelenmesi hedeflenmiştir. Bu yaklaşımları desteklemek üzere yazılım kalite kontrol araçlarının nasıl kullanılabilceği, neleri karşılayıp, neleri karşılayamadığı, Windows Azure örnek bulut uygulaması üzerinden incelenerek, söz konusu araçları kapsayan örnek test uygulamaları ile beraber değerlendirilmiştir.

Anahtar sözcükler: Bulut bilişim, Yazılım kalite kontrol yönetimi, Bulut uygulamalarının kalite kontrolü, Windows Azure, Visual Studio Web Tools, Selenium Test Suite

CONTENTS

	Page
M.SC THESIS EXAMINATION RESULT FORM	ii
ACKNOWLEDGEMENTS	iii
ABSTRACT	iv
ÖZ	v
LIST OF TABLES	ix
LIST OF FIGURES	x
CHAPTER ONE- INTRODUCTION	1
1.1 What's Cloud Computing?	1
1.1.1 Benefits of Cloud Computing and Architecture	1
1.1.2 Example Cloud Platforms	3
1.1.2.1 Google Web Services	3
1.1.2.1.1 Google App Engine	4
1.1.2.1.2 Amazon Web Services	4
1.1.2.1.3 Windows Azure	6
1.1.2.3.1 Windows Azure Components	7
1.2 Goal of This Work	10
1.3 Thesis Organization	11
CHAPTER TWO – DESIGNING AND TESTING FOR CLOUD	12
2.1 Designing for Cloud Computing	12
2.1.1 Scalable Data Storage Techniques	12
2.1.2 MapReduce Programming	13
2.1.3 Rich Internet Applications (RIA)	15
2.1.4 Summary	17
2.2 Testing for Cloud Computing	18
2.2.1 Functional Testing	20

2.2.2 Non-Functional Testing.....	22
2.3 Testing in Production (TIP).....	25
2.4 Challenges of Testing Cloud Applications.....	27
2.5 Conclusion.....	28

CHAPTER THREE - TOOLS AND METHODS FOR CLOUD TESTING 29

3.1 A Sample Cloud Application to Be Tested.....	29
3.1.1 Preparing Development Environment.....	29
3.1.1.1 Install Windows Azure SDK for .NET.....	29
3.1.1.2 Set up a Windows Azure Account.....	30
3.1.1.3 Create a Windows Azure Storage Account.....	30
3.1.1.4 Creating Cloud Service in Windows Azure.....	33
3.1.1.5 Creating a Web Site in Windows Azure.....	35
3.1.2 Developing Windows Azure Application.....	37
3.1.3 Deploying Windows Azure Application.....	39
3.2 Testing the Sample App with Visual Studio.....	41
3.2.1 Web Performance Tests.....	41
3.2.2 Creating and Configuring Web Tests.....	42
3.2.2.1 Recording a Web Performance Test.....	43
3.2.3 Running and Observing Results for Web Performance Tests.....	46
3.2.4 Coded Web Performance Tests.....	46
3.2.4 Load Tests.....	48
3.2.4.1 Creating and Configuring Load Tests.....	48
3.2.4.1.1 Scenarios and Think Times.....	49
3.2.4.1.2 Load Patterns.....	50
3.2.4.1.3 Test Mix Model.....	51
3.2.4.1.4 Test Mix.....	52
3.2.4.1.5 Network Mix.....	53
3.2.4.1.6 Browser Mix.....	53
3.2.4.1.6 Performance Counter Sets.....	54
3.2.4.1.7 Run Settings.....	54

3.2.4.2 Executing Load Tests and Viewing Results.....	56
3.3 Summary for Visual Studio Tools.....	57
3.4 Testing the Sample App with Selenium.....	58
3.4.1 Brief history of the Selenium Project.....	58
3.4.2 Selenium RC Architecture.....	59
3.4.2.1 Proxy Injection.....	60
3.4.3 Building a Web Test with Selenium IDE.....	62
3.4.4 Example of Selenium Web Driver API.....	65
3.4.5 Creating and Running Selenium Test.....	66
3.5 Summary for Selenium Suite.....	71
CHAPTER FOUR - CONCLUSION.....	73
REFERENCES.....	75

LIST OF TABLES

	Page
Table 2.1 Silverlight XAML code for “Hello World” application.....	16
Table 2.2 Code Behind for “Hello World” application.....	17
Table 2.3 Test environment in cloud.....	24
Table 2.4 TiP methodologies.....	25
Table 2.5 Main Testing challenges for Cloud applications.....	27
Table 3.1 XML code behind for web performance test.....	45
Table 3.2 Coded Web performance test in Visual Studio.....	47
Table 3.3 Sample Selenium WebDriver code.....	65
Table 3.4 Selenium WebDriver code to test cloud application.....	69

LIST OF FIGURES

	Page
Figure 1.1 Six phases of computing paradigm.....	2
Figure 1.2 Windows Azure platform.....	7
Figure 1.3 Windows Azure provide compute and storage services in the cloud....	8
Figure 2.1 RIA technologies.....	15
Figure 2.2 RIA technologies fill the gap between rich and reach applications....	16
Figure 2.3 A new approach required to testing cloud applications.....	20
Figure 3.1 Web Platform Installer to install Windows Azure SDK for Visual Studio.....	30
Figure 3.2 Storage screen on Windows Azure.....	31
Figure 3.3 Creating storage account in Windows Azure.....	32
Figure 3.4 Managing keys for storage account in Windows Azure.....	32
Figure 3.5 “Primary/Secondary Storage Keys” in Windows Azure.....	33
Figure 3.6 Creating a cloud service in Windows Azure.....	34
Figure 3.7 Options for creating cloud service in Windows Azure.....	35
Figure 3.8 Creating a new web site in Windows Azure.....	36
Figure 3.9 Create web site options in Windows Azure.....	36
Figure 3.10 Creating a Web Application in Visual Studio 2012.....	37
Figure 3.11 Internet project template selections in Visual Studio 2012.....	38
Figure 3.12 Sample application home screen.....	38
Figure 3.13 Screenshot of the “Web Sites” tab in Windows Azure.....	39
Figure 3.14 Screenshot of the “Web Site Dashboard” in Windows Azure.....	40
Figure 3.15 Screenshot for Visual Studio publish web wizard.....	36
Figure 3.16 Creating new Test Project in Visual Studio.....	43
Figure 3.17 Result of Web Performance Test recording in Visual Studio.....	44
Figure 3.18 Results of the execution of Web Performance Test.....	46
Figure 3.19 First page of “New Load Test Wizard”.....	49
Figure 3.20 Load Pattern options in “New Load Test Wizard”.....	50
Figure 3.21 Test mix model in “New Load Test Wizard”.....	52
Figure 3.22 Network mix selection in “New Load Test Wizard”.....	53

Figure 3.23 Browser distribution page for New Load Test Wizard	54
Figure 3.24 Run settings page for “New Load Test Wizard”	55
Figure 3.25 Test Results for Load Test	56
Figure 3.26 Load Test monitor in Visual Studio	56
Figure 3.28 Selenium RC Architecture	61
Figure 3.29 Selenium Project download page	63
Figure 3.25 Test Results for Load Test	56
Figure 3.26 Load Test monitor in Visual Studio	56
Figure 3.28 Selenium RC Architecture	61
Figure 3.29 Selenium Project download page	63
Figure 3.30 Snapshot of Selenium IDE	63
Figure 3.31 Available commands for Selenium context-menu	67
Figure 3.32 Selenium IDE after the recording stopped	68
Figure 3.33 Running tests in Selenium IDE	69

CHAPTER ONE

INTRODUCTION

1.1 What's Cloud Computing?

Cloud computing is the new way of presenting services over the Internet in a dynamically scaled and mostly virtualized way (Furht & Escalante, 2010). It is most of the times recognized as resources are virtual and limitless and the details of the physical systems on which software runs are abstracted from the user. The term "cloud" makes reference to the two essential concepts as described by Sosinsky (chap 1, 2011).

Abstraction: Cloud computing abstracts the details of system implementation from users and developers. Applications run on physical systems that aren't specified, data is stored in locations that are unknown, administration of systems is outsourced to others, and access by users is ubiquitous.

Virtualization: Cloud computing virtualizes systems by pooling and sharing resources. Systems and storage can be provisioned as needed from a centralized infrastructure, costs are assessed on a metered basis, multi-tenancy is enabled and resources are scalable with agility.

1.1.1 Benefits of Cloud Computing and Architecture

Basically two types of cloud computing is identified, one based on where the services are deployed and the other based on the type of the service model. According to the deployment model, the type of cloud is defined on where the cloud is located and for what purpose, i.e. public, private and hybrid clouds.

The service model defines the type of service that the service provider is offering, best known are Software as a Service, Platform as a Service and Infrastructure as a

Service. The service model builds on one another and defines what a vendor must manage and what the client's responsibility is (Sosinsky, 2011).

Cloud computing has been an umbrella term to describe a category of sophisticated on-demand computing services. Figure 1.1 shows six phases of computing paradigms as described by Furht & Escalante (chap. 1, 2010).

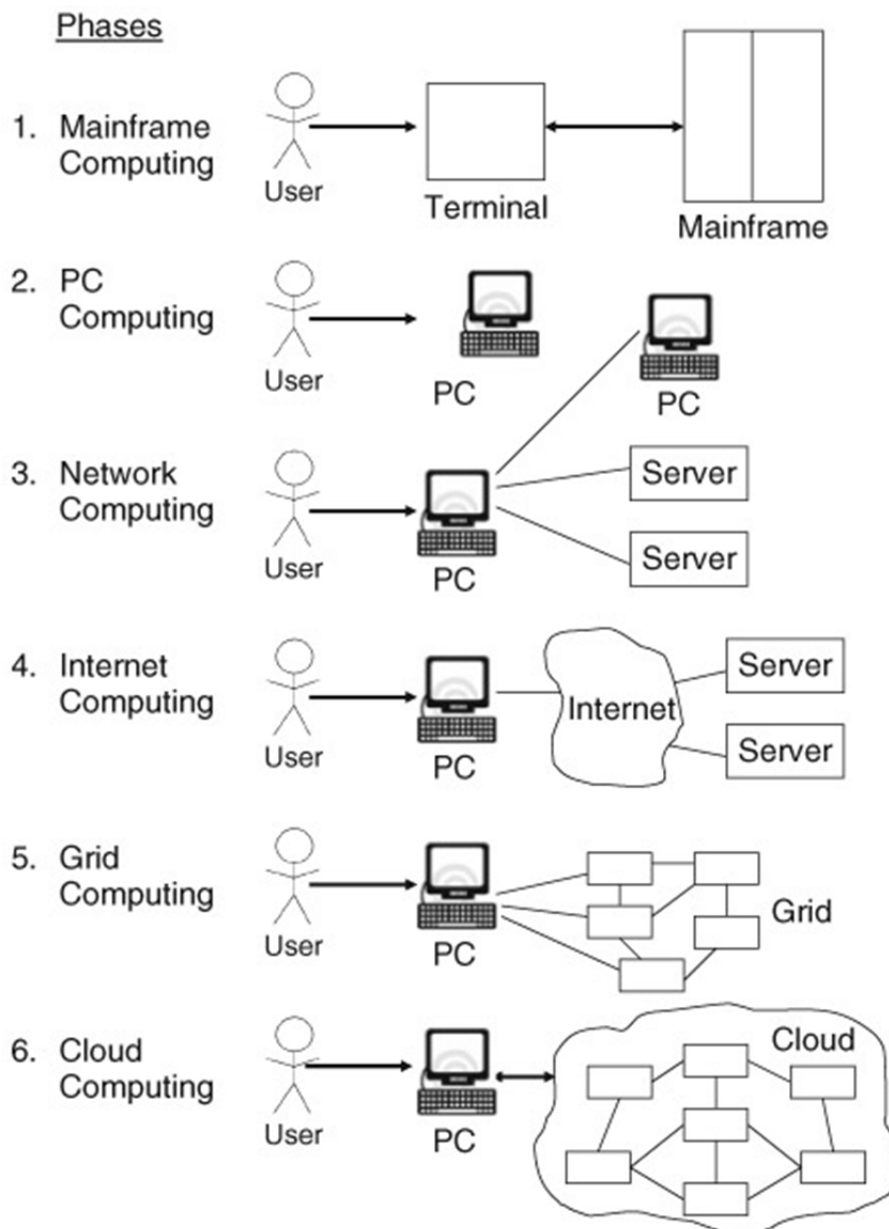


Figure 1.1 Six phases of computing paradigm

In the first phase users shared powerful mainframes using dummy terminals, in phase 2, stand-alone PCs became so powerful that it met majority of users' needs. In the next phase, PCs, laptops and servers were connected together through local networks to share resources and increase performance. In Phase 4, local networks were connected to other local networks forming a global network such as the internet to utilize remote applications and resources. In Phase 5, grid computing provided shared computing power and storage through a distributed computing system. In phase 6, cloud computing further provides shared resources on the Internet in a scalable and simple way.

1.1.2 Example Cloud Platforms

There are a couple of illustrative examples of cloud computing services which will be referred throughout this thesis. These examples illustrate the fact that 'cloud computing' is becoming a mode of IT delivery. The cloud stack comprises layers from infrastructure up to business process in the same way the IT stack does.

The aim is to direct the aim of this work thesis towards tangible examples and to enable a quicker understanding of cloud computing by examining these services.

1.1.2.1 Google Web Services

Google is the prototypical company of cloud computing services and supports some of the largest web sites and services in the world. At the center of Google's core business is the search technology. Google uses automated technology to index the Web and makes the search service available to users as a standard search engine and to developers as a collection of special search tools limited to various areas of content.

Google is offering Google Apps for Business which includes a bundle of services, such as e-mail and cloud storage and operates services such as Google Maps, Google Finance, Google Voice and Google App Engine.

Google App Engine lets you host your own web applications on Google's infrastructure. However your application is not hosted on a single server and there are no servers to maintain. The background of hosting task is invisible to the user so that the user is free of the infrastructure, capacity manager, and load balancing tasks that enterprise typically have to manage. You can serve your app from your own domain name and share your application or limit access to members of your organization.

As with most cloud-hosting services you pay for what you use in terms of storage, bandwidth and computational resource.

1.1.2.1.1 Google App Engine. Google App Engine lets you run your own web application on Google's infrastructure. Application doesn't run on a single server, it might even involve more than one server. The infrastructure is totally abstracted from the application and Google App Engine is totally free of the infrastructure, capacity management, and load balancing tasks that enterprise typically has to manage. Application can be served from owned domain name using Google Apps or served from a free sub-domain owned by Google. It supports applications written in several programming languages. Application developers have access to persistent storage technologies such as the Google File System (GFS) and Bigtable, a distributed storage system for unstructured data. The Java version supports asynchronous non-blocking queries using the Twig Object Datastore interface.

1.1.2.2 Amazon Web Services

Amazon.com is one of the most important and heavily trafficked Web Sites in the world. It provides a vast selection of products using an infrastructure based on Web services. As Amazon.com has grown, it has dramatically grown its infrastructure to accommodate peak traffic times. Over time the company has made its network resources available to partners and affiliates, which also has improved its range of products.

Starting in 2006, Amazon.com made its Web service platform available to developers on a usage-basis model. Through hardware virtualization on Xen hypervisors, Amazon, com has made it possible to create private virtual servers that you can run worldwide. These servers can be provisioned with almost any kind of application software you might envisage, and they tap into a range of support services that not only make distributed cloud computing applications possible, but make them robust. Some very large Web sites are running on Amazon.

Amazon Web Services is based on SOA standards, including HTTP, REST, and SOAP transfer protocols, open source and commercial operating systems, application servers, and browser-based access. Virtual private servers can provision virtual private clouds connected through virtual private networks providing for reasonable security and control by the system administrator.

AWS has a great value proposition: You pay for what you use. While you may not save a great deal of money over time using AWS for enterprise class Web applications, you encounter very little barrier to entry in terms of getting your site or application up and running quickly and robustly. It represents the largest pure Infrastructure as a Service (IAAS). It is comprised of the following components.

Amazon Elastic Compute Cloud (EC2): It is the central application in the AWS portfolio and enables the creation, use and management of virtual private servers running the Linux or Windows operating system over a Xen hyper-visor. Amazon Machine Instances are sized at various levels and rented on a computing/hour basis. Spread over data centers worldwide, EC2 applications may be created that are highly scalable, redundant and fault tolerant.

Amazon Simple Storage System (S3): It is online backup and storage system and allows you to store data objects ranging in size from 1 byte up to 5GB in a flat namespace.

Amazon Elastic Block Store (ESB): It is a system for creating virtual disks (volume) or block level storage devices that can be used for Amazon Machine Instances in EC2.

Amazon SimpleDB: It is a structured data store that supports indexing and data queries to both EC2 and S3. SimpleDB isn't a full database implementation; it stores data in "buckets" and without requiring the creation of a database schema.

Amazon Relational Database Service (RDS): It allows you to create instances of the MySQL database to support your Web sites and the many applications that rely on data-driven services. RDS provides features such as automated software patching, database backups and automated database scaling via an API call.

Amazon CloudFront: It is an edge-storage or content-delivery system that caches data in different physical locations so that user access to data is enhanced through faster data transfer speeds and lower latency.

1.1.2.3 Windows Azure

Windows Azure is a cloud services operating system that serves as the development, service hosting, and service management environment for the Azure Services Platform, which provides developers with on-demand compute and storage to host, scale, and manages Web applications on the Internet through Microsoft data centers.

It is an operating system in the cloud providing services for hosting, management, and scalable storage with support for simple blobs, tables, and queues, as well as a management infrastructure for provisioning and geo-distribution of cloud-based services, and a development platform for the Azure Services layer.

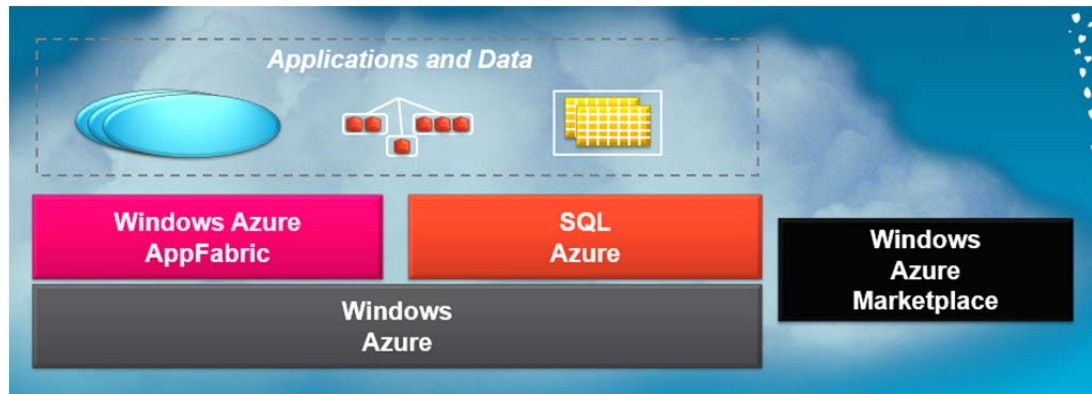


Figure 1.2 Windows Azure platform

As Figure 1.2 shows The Windows Azure platform today can be seen as having four parts:

- Windows Azure: A Windows environment for running applications and storing data on computers in Microsoft data centers.
- SQL Azure: Relational data services in the cloud based on SQL Server.
- Windows Azure-based Services: Cloud-based infrastructure for application running in the cloud or on premises.
- Windows Azure Marketplace: An online service for finding and purchasing cloud-based applications and data.

1.1.2.3.1 Windows Azure Components. At a high level it runs windows applications and stores data in the cloud. Figure 1.3 shows basic components.

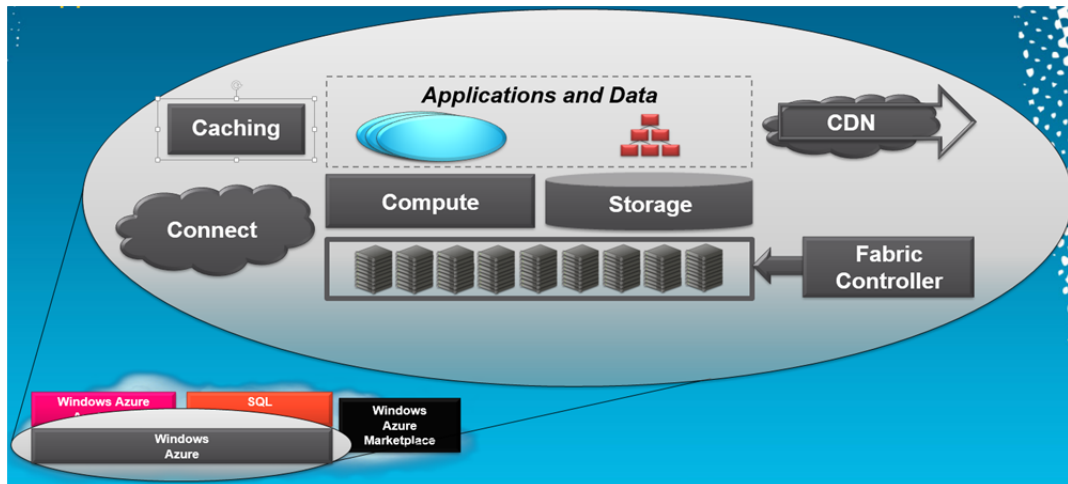


Figure 1.3 Windows Azure provide compute and storage services in the cloud

Compute: The Windows Azure compute service runs applications on a Windows Server foundation. These applications can be created using the .NET Framework in languages such as C# and Visual Basic, or they can be built without .NET in C++, Java and other languages. Developers can use Visual Studio or other development tools, and they are free to use technologies such as ASP.NET, Windows Communication Foundation (WCF), and PHP.

Storage: This service allows storing binary large objects (blobs) provides queues for communication between components of Windows Azure applications and even offers a form of tables with a simple query language. Both Windows Azure applications and on-premises applications can access the Windows Azure storage service, and both do it in the same way: using a Restful approach.

Fabric Controller: Windows Azure runs on a large number of machines. The fabric controller's job is to knit the machines in a single Windows Azure data center into a cohesive whole. The Windows Azure compute and storage services are then built on top of this pool of processing power.

Caching: It's common for applications to access the same data over and over. One way to speed up this kind of application is to cache frequently accessed information in memory, reducing the number of times the application must query a

database. The caching service provides this—and the performance boost it brings—for Windows Azure applications.

Connect: It's often useful for organizations to interact with cloud applications as if they were inside the organization's own firewall. Windows Azure Connect allows this, making it easier for, say, a Windows Azure application to access an on-premises database.

Content delivery network (CDN): Caching frequently accessed data closer to its users speeds up access to that data. The Windows Azure CDN can do this for blobs, maintaining cached copies at sites around the world.

Windows Azure is a general platform that can be used in a broad set of scenarios. Here are a few examples.

- An independent software vendor (ISV) creating a software-as-a-service (SaaS) version of an existing on-premises Windows application might choose to build it on Windows Azure. Because Windows Azure mostly provides a standard Windows environment, moving the application's business logic to this cloud platform won't typically pose many problems. And once again, building on an existing platform lets the ISV focus on business logic—the thing that makes them money—rather than spending time on infrastructure.
- An enterprise creating an application for its customers or employees might choose to build it on Windows Azure. Because Windows Azure supports .NET, developers with the right skills aren't difficult to find, nor are they prohibitively expensive. Running the application in Microsoft data centers frees the enterprise from the responsibility and expense of managing its own servers, turning capital expenses into operating expenses. And especially if the application has spikes in usage, letting Microsoft maintain the large server base required for this can make economic sense.

- A start-up creating a new Web site could build its application on Windows Azure. Because the platform supports both Web-facing services and background processes, the application can provide an interactive user interface as well as executing work for users asynchronously. Rather than spending time and money worrying about infrastructure, the start-up can instead focus solely on creating code that provides value to its customers and investors. The company can also start small, incurring low costs while its application has only a few users. If the application catches on and usage increases, Windows Azure can let the application scale as needed.

1.2 Goal of This Work

In this study a research around methodologies and tools for testing cloud-computing applications will be carried out. In the broader terms, software testing refers the activities carried out in order to ensure a higher quality, fully-functional and free of most of the bugs software is being shipped. It is part of the software lifecycle development and there are different methodologies developed over the years for the classical software testing notion.

However in most of the cases, software is being developed for a pre-defined time period with all engineering processes being applied. On the other side, in case of cloud-computing applications, the release cycle is shorter, therefore the process, including all testing activities must be agile and generating results in a shorter time.

And also due to nature of cloud computing, mainly as a result of abstraction and virtualization, the environment where the software will be run cannot be predefined.

Adding up all these, it is obvious that classical testing processes and notions should be tweaked to better apply to cloud-computing.

Main idea behind this study is investigating existing methodologies and presenting different researches which target testing of cloud-computing software.

Different tools, ideas and frameworks will be presented including applications where possible.

We will be mostly focusing Windows Azure applications and also tools from Visual Studio family. But we will also be touch basing other cloud platforms and other testing tools.

This is not intended to be a comprehensive, deep-dive research of all cloud platforms and the tools required to test applications on cloud platforms. But rather it serves as a starting point to open up that discussion, as testing cloud-based applications is still a new area and we are expecting more tooling and framework support coming in the future.

1.3 Thesis Organization

This thesis is organized as follows.

In Chapter 2, in classical software development terms software testing is explained. It starts overviewing different types of software testing and how each traditional type of software testing can be applied to testing cloud based applications

In Chapter 3, existing tools and methodologies will be evaluated in terms of testing cloud based applications. These tools are mainly targeting testing software in the traditional way of software development. Strength and weaknesses of each tool will be presented in terms of testing cloud based applications

In Chapter 4 some of the open issues and current and future developments will be concluded for testing cloud based applications

CHAPTER TWO

DESIGNING AND TESTING FOR CLOUD

2.1 Designing for Cloud Computing

There are new design paradigms that have evolved for cloud computing applications; it requires different concepts and theoretical knowledge to develop efficient cloud applications on these new platforms. New methodologies such as enabling scalable data storage using database partitioning and key-value stores are required for this kind of application development. Platform as a Service (PaaS) technologies enabled developers to create efficient, scalable applications to be hosted on the Web.

New concepts such as handling large scale data storage during application execution use of both relational databases and NoSQL data stores to store cloud-hosted data, The Hadoop platform – which brings a new notion of MapReduce programming etc. requires rethinking on the application design.

2.1.1 Scalable Data Storage Techniques

Cloud Applications may have data storage requirements that exceed those of enterprise applications. High capacities of this kind far exceed the needs of enterprise storage systems. In addition high throughput may also be a reason why conventional technologies cannot scale to the cloud. The basic technique to scale storage systems to cloud-scale is to partition and replicate the data over multiple independent storage systems. Partitioning and replication increases the overall throughput of the system, since the total throughput of the combined system is the aggregate of the individual storage systems.

The other technology for scaling storage is known as NoSQL. NoSQL was developed as a reaction to the perception that conventional databases, focused on the need to ensure data integrity for enterprise applications, were too rigid to scale to

cloud levels. As an example, conventional databases enforce a schema on the data being stored, and changing the schema is not easy. However, changing the schema may be a necessity in a rapidly changing environment like the cloud. NoSQL storage systems provide more flexibility and simplicity compared to relational databases. The disadvantage, however, is greater application complexity. The applications have to be written to deal with data records of varying formats.

Partitioning and replication also increase the storage capacity of a storage system by reducing the amount of data that needs to be stored in each partition. However, this creates synchronization and consistency problems.

2.1.2 MapReduce Programming

MapReduce is a popular paradigm of programming for the Cloud, which particularly works well for large-scale data processing. It is very effective for massively data-parallel applications that can be parallelized to crunch data on hundreds or thousands of CPUs. Traditional ways of writing parallel and distributed programs require the developer to explicitly split the tasks as multiple processes, deploy these processes on multiple CPUs and also manage the communication among the processes (through communication APIs) to exchange intermediate data values or final results. Writing such distributed applications is not very easy for a developer who has programmed for sequential machines. The MapReduce programming model makes development of such parallel applications very easy. The programmer just specifies a map function and a reduce function for the application and the MapReduce framework does automatic parallelization and distribution of data to result in efficient parallel execution of the Cloud application. Furthermore, the platform ensures that the application is fault tolerant. (Sitaram & Manjunath, 2012)

The Map function takes as input a key-value pair and generates a set of intermediate key-value pairs. The MapReduce platform then collates all the intermediate values from parallel Map function execution into groups that

correspond to a single key and sends them to the Reduce function. The reduce, on the other hand, takes this intermediate key and the set of values corresponding to that key and combines these values to form a smaller number of key value pairs (typically one or zero values) as the overall result of the computation.

The processing flow for a MapReduce program is as follows:

- The input data is split into chunks, each of which is sent to different Mapper processes to execute in parallel. The parallel execution is achieved when the Map function just reads the relevant key-value pairs given to it.
- The result of the Mapper process is partitioned based on the Key and is sorted locally. The user can also provide the comparator operator here. This sorting is done by the MapReduce platform and is referred to as Shuffle.
- Result from the different Mappers that have the same key will be given as input to the same Reduces instance. The Reduce function (provided by the user) processes this sorted key-value data to generate the output.

From the programming model perspective, the MapReduce abstract model is based on the following simple concepts:

- Iteration over the input;
- Computation of key/value pairs from each piece of input;
- Grouping of all intermediate values by key;
- Iteration over the resulting groups;
- Reduction of each group.

The MapReduce paradigm provides a clean abstraction for programmers to easily develop data parallel applications. However, developers need to learn this new paradigm of programming that borrows a lot from functional programming concepts.

2.1.3 Rich Internet Applications (RIA)

Since cloud computing services have to be accessed over the internet, end user applications need to have good interface that is user friendly and rich in experience. Such applications are called Rich Internet Applications (RIA). These provide very good user experience rather than just presenting the desired information.

A traditional application may display data in tables, for example sales figures. For any statistical computation or displaying charts etc., the application needs to go back to the server. With RIA, these all can be done right on the client side which makes the webpage look “rich” in content and has lower delays during interaction (Sitaram & Manjunath, 2012).

RIA applications can either run within a web browser with client-side scripts (JavaScript) and a browser plug-in or execute within a secure sandbox as desktop applications (e.g. Flash applications). RIA platforms have their own runtime libraries, execution engines and rendering mechanisms. For example, Flex by Adobe runs on the Flash runtime and Microsoft Visual Studio runtime is Silverlight. These runtimes run on the client side independent of the server.



Figure 2.1 RIA technologies

The development of an RIA starts with an Integrated Development Environment (IDE) such as Flash Builder or Microsoft Visual Studio. The developer uses this IDE to develop the application using design view and code view of the IDE. Generally,

the code is written in some extension of XML, in the case of Flex this is MXML or in case of Silverlight this is XAML. However the source code can contain ActionScript code in Flex case, or any .NET language in Silverlight case. In the case of AJAX (Asynchronous JavaScript and XML), the script is written in JavaScript, which is directly interpreted by the browser which doesn't require any browser plug-ins.

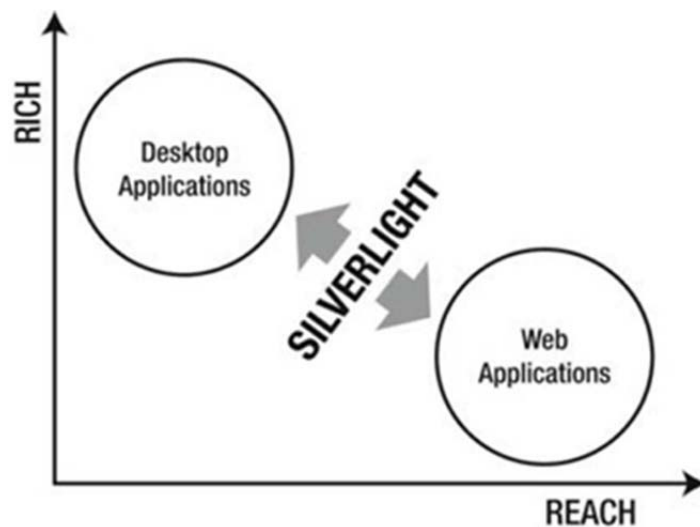


Figure 2.2 RIA technologies fill the gap between rich and reach applications

In the following table there is a simple example RIA application using Silverlight as an illustrative platform. This code builds a simple Hello World application which uses a TextBlock control to display the message. The code behind for this changes the text in TextBlock in C#.

Table 2.1 Silverlight XAML code for “Hello World” application

```

1. <UserControl x:Class="HelloWorld.MainPage"
2.   xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
3.   xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
4.   xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
5.   xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
6.   mc:Ignorable="d"
7.   d:DesignHeight="300" d:DesignWidth="400">
8.
9.   <Grid x:Name="LayoutRoot" Background="White">
10.     <TextBlock x:Name="HelloMessage" Text="Hello World!" FontSize="30" />
11.   </Grid>
12. </UserControl>

```

The code behind for the above Silverlight application is shown in Table 2.2. In this code behind, the contents of the “TextBlock” is set to another value at run time.

Table 2.2 Code Behind for “Hello World” application

```
1. namespace HelloWorld
2. {
3.     public partial class MainPage : UserControl
4.     {
5.         public MainPage()
6.         {
7.             InitializeComponent();
8.             this.HelloMessage.Text = "Hello Universe!";
9.         }
10.    }
11. }
```

2.1.4 Summary

As described in the sections above there are multiple deep technical concepts that will aid in writing efficient cloud applications. All these technical concepts deviate from traditional concept of designing and developing software applications. For example, instead of traditional, heavily-regulated strict-consistent ruled relational databases an alternative flexible schema, simple interface storage systems are evolving. An emerging paradigm of developing cloud applications using the MapReduce technique helps with basic concepts of data parallelism and functional programming. There is need to develop new types of algorithms to leverage the data parallel computation platform enabled by MapReduce platforms.

Finally, Rich Internet Technology relies on running the application either within the sandbox of the browser or as a desktop application which eases the load on server. Therefore this results both in richer applications which costs less to run against cloud platforms and still has very enhanced user experiences.

All these new techniques of developing code also requires newer and adopted notions of testing which will be covered in the next section.

2.2 Testing For Cloud Computing

With all the benefits of Cloud Computing, it also presents risks which must be mitigated effectively if Cloud Computing is to be a viable option for businesses.

- Lack of control: When IT infrastructure is outsourced to an external third party, how does the business control their data, impact down time, drive technology change or influence decisions which may impact their solution?
- Security: How can business ensure the potentially sensitive information which traverses the cloud is safe and secure?
- Privacy: What can business do to maintain the privacy of their users and information when using the cloud?
- Data Integrity: With flexible NoSQL storage how does businesses assure their valuable data remains intact?
- Availability: Cloud computing solutions rely on the availability of their infrastructure to be able to function. Should a business critical Cloud solution be unavailable for any time, what is the business impact?
- Business acceptability: How sure can a business be that their third party solution is suitable for its intended use?

Testing is important to enhance user agreement and reduce the maintenance and cost. With the applications running on a diverse set of platforms, from mobile devices, to tablets, to desktops and virtual environments, validating applications based on functional and non-functional aspect on a broad range of devices requires a variety of tools.

Testing of cloud services has some familiar aspects and some new challenges. The quality of the business services being build and deployed in cloud should be assured. There are some factors to be considered in this sense.

- Software performance on the application, and the way it's hosted in the client environment
- Database performance
- Network performance
- Server performance

The advantage of application development testing in the cloud allows you to more closely mirror a production environment to a staged/test system so that risk is mitigated when applications are deployed live. The use of cloud computing means lower cost and lower capital expenditure.

The challenges that surround testing applications in cloud environments include.

- Testing cloud applications and networks demands a wide mix of application traffic, current security coverage and incredibly high-performance and throughput.
- There is a need to create a realistic testing environment with an ever-changing mix of applications and increasingly sophisticated security attacks. This makes delivering high performance a moving target.
- Traditional testing tools were simply not designed for this dynamic, complex and high performance computing environments.
- Predictability of software becomes more important.
- Functional testing using tools like Selenium, Windmill, twill, Visual Studio setup and execution is another hallmark of using the cloud for application testing.

All of these challenges results in an outcome of a new approach to testing. We need a good understanding of what makes a cloud computing application and

distributed architecture. We also need to better understand what testing tools are available, what their strengths and weaknesses are for testing different types of cloud applications.

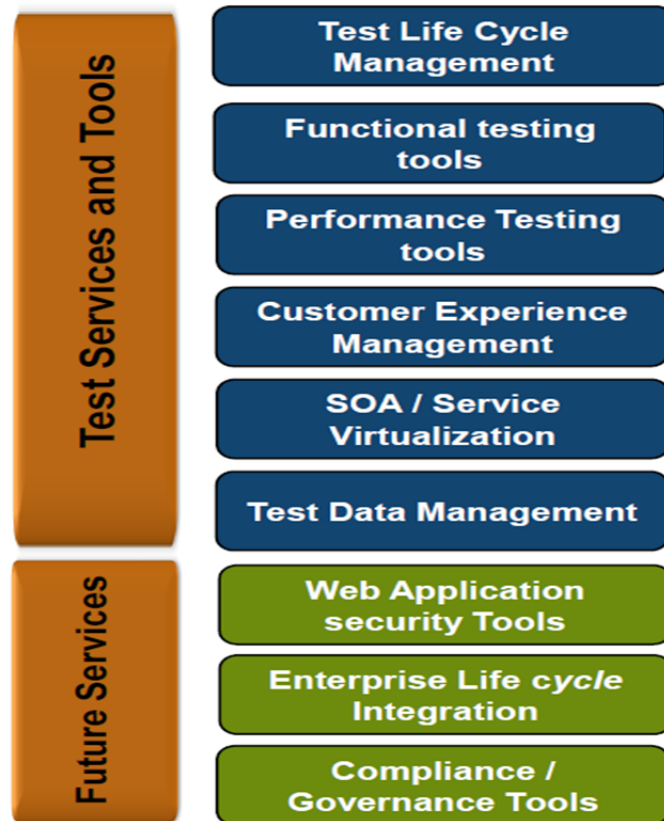


Figure 2.3 A new approach required to testing cloud applications

2.2.1 Functional Testing

Functional testing refers to gathering more in-depth information to deconstruct the product's feature sets and test the discrete functional attributes and capabilities of the various components. Functional testing techniques increase the effectiveness of testing and gather important information about certain aspects of the product. They are systematic procedures that can help to perform a comprehensive investigation of the software's functional attributes and capabilities.

Some main functional testing techniques involve boundary value analysis, equivalence class partitioning, combinatorial analysis, and state transition testing.

Equivalence Class Partitioning is a tool that enables the tester to evaluate input or output variables systematically for each parameter in a feature. It requires that a comprehensive analysis of the variable data for each parameter in the context of the specific system is performed. Before designing the ECP tests, variable data for each input/output parameter in discrete subsets of valid and invalid classes should be decomposed and modeled. Then ECP tests are derived from creating unions of valid class subsets until all valid class subsets have been used in a test, and then each invalid data subset is evaluated individually.

Boundary value analysis is a technique that targets data values at immediately above and immediately below a specific boundary condition. Historical experience and root cause analysis of recurring problems demonstrate that anomalies are common occurrences at or near the boundary conditions of independent input or output parameters.

With cloud computing technologies emerging, traditional approach to functional testing and software testing life cycle processes are likely to be impacted. Test requirements, test planning and test data management aspects of functional testing are will need to revisited in the context of cloud software testing.

Test Requirements: Applications and systems are expected to be hosted in a private, public or hybrid clouds which may present a dependency on the third party vendors.

Test Planning: An environment which represents similar characteristics to the one in which it will be accessed when it goes live is required for sake of completeness. Testing should scrutinize the application's performance, reliability, speed, security and functionality. Software testing tools that are used for testing of conventional applications have to be re-evaluated when applied to testing application which is hosted in the cloud, as there is a need for tools to allow test engineers to analyze the network, desktop and implications of changes within the cloud.

Test Data Management: Testing should be covering data encryption and doubly scrutinized while testing in cloud. Data obfuscation takes a big precedence and synthetic data has increasing importance while testing cloud based applications.

Due to nature of the cloud computing applications, as an addition to traditional testing methods/techniques the following additional components should be included in the scope.

Support for Multiple browsers: Since the applications residing on cloud are targeted by different types and vendors of browsers, any incompatibility should be tested before going live. Browser types such as Internet Explorer, Firefox, Safari, Google Chrome, Opera and also environments such as mobile/handheld devices; personal computers etc. should be evaluated.

Availability: Availability is the verification of the redundancy of individual modules, including the software that controls these modules (Naik & Tripathy, 2008). Since there is a build in dependency on 3rd party cloud platform vendors, availability plays a critical role on the software. Most vendors have published service level agreements (SLA) for availability. Along with these SLA, high-availability should be main aspect of testing. The concept is also known as fault-tolerance. The goal is to recover quickly and gracefully from failures and continue execution by minimizing the downtime.

Data Security: These types of tests should include verifying that sensitive information which travels through cloud is safe and secure. Any sniffing, man-in-the-middle attacks should only get encrypted data.

2.2.2 Non-Functional Testing

Where functional testing involves positive and negative testing of the functionality in the application under test, non-functional testing include

performance, load, security, reliability and many other areas. It is referred as behavioral tests or quality tests (Page et.al. 2009).

Reliability is a measure of how well software maintains its functionality in mainstream or unexpected situations. It might also include the ability to recover from a fault.

Usability measures how easy it is for users of the software to understand learn and control the application to accomplish whatever they need to do. Usability studies, customer feedback and examination of error messages and other dialogues all support usability.

Maintainability is the effort needed to make changes in software without causing errors. Product code and test code alike must be highly maintainable. Knowledge of the code base by the team members, testability, and complexity all contribute to this attribute.

Portability is the ability for code to be modified easily to be able to run on different platforms, including different architectures, environment, operating system etc.

Unlike the traditional non-functional testing techniques, where scalability is limited to certain number of users within the network, in cloud applications scalability scope is much wider. In order to sustain the attributes mentioned above for cloud applications, the following types of testing types should be carried over.

Performance Testing: Measuring response times and isolating issues related to specific steps or actions while system is subjected to increasing load from different locations and multi user operations is critical for cloud based applications.

Load/Stress Testing: Application / system stability is a major factor as the user count is expected to be many. The application should be able to handle unexpected

loads, and depending on what type of cloud system, should be able to respond by dynamically adding more processing power/bandwidth etc. Due to nature of cloud systems, it is definitively must to identify issues to find out breaking points. Software should be ready to handle 2x, 3x, nx of the maximum expected load.

Failover Testing: Being in cloud environment, a mission critical application must be running in multiple data-centers to accommodate outages for a single datacenter. The software should be able to switch instances quickly with minimal downtime and start running on another instance (datacenter, vm, geo location etc.) without too much manual intervention.

In table 2.1 you can find some characteristics of test environment in Cloud with attributes of cloud-based systems and how testing can benefit from these attributes.

Table 2.3 Test environment in cloud

Attributes of Cloud Solutions	Characteristics	Benefits
Advanced virtualization	Test resources are pooled and virtualized	Providing efficient implementation of independent infrastructure
Rapid provisioning	Test resources are provisioned on demand	Reducing test setup and execution time, and eliminating errors
Service catalog ordering	Test environment are readily available	Enabling visibility, control and automation
Elastic scaling	Test environment can be scaled up or down by large factor as the need emerges	Optimizes infra and software license usage

2.3 Testing In Production (TiP)

With the current nature of cloud applications, the current approach for testing in the classical sense, which is the big up-front testing in a test lab, can only be an attempt to approximate the true complexities of the operating environment. Testers try best to anticipate the edge cases and understand the environments, but in cloud driven world they cannot anticipate everything users do and data centers are hugely complex systems unto themselves with interactions between servers, networks, power supplies and cooling systems.

TiP is not about throwing an untested code directly to all users. The risk needs to be controlled while improving quality. As an emerging trend, TiP is still new and the taxonomy is far from finalized. However, some methodologies can still be identified reviewing the publicly available documentation and practices, as shown in Table 2.4.

Table 2.4 TiP methodologies

Methodology	Description
Ramped Deployment	Launching new software by first exposing it to subset of users then steadily increasing user exposure. Purpose is to deploy, may include assessment. Users may be hand-picked or aware they are testing a new system
Controlled Test Flight	Parallel deployment of new code and old with random unbiased assignment of unaware users to each. Purpose is to assess quality of new code, then may deploy. May be part of ramped deployment.
Experimentation for Design	Parallel deployment of new user experience with old one. Former is usually well tested prior to experiment. Random unbiased assignment of unaware users to each. Purpose is to assess business impact of new experience

Table 2.4 Continued.

Dogfood/Beta	User-aware participation in using new code. Often by invitation. Feedback may include telemetry, but it is often manual/asynchronous.
Synthetic Test in Production	Functional test cases using synthetic data and usually at API level, executing against in-production systems. “Write once, test anywhere” is preferred; same test can run in test environment and production. Synthetic tests in production may make use of production monitors/diagnostics to assess pass or fail.
Load/capacity Test in Production	Injecting synthetic load onto production systems, usually on top of existing real-user load, to assess systems capacity. Requires careful (often automated) monitoring of back-off mechanisms.
User Scenario Execution	End-to-end user scenarios executed against live production system from (or close to) same point of origin as user-originated scenarios. Results then assessed for pass/fail. May also include manual testing.
Data Mining	Test cases search through real user data looking for specific scenarios. Those that fail their specified oracle are filed as bugs.
Destructive Testing	Injecting faults into production systems (services, servers, and network) to validate service continuity in the event of a real fault.
Production Validation	Monitors in production check continuously (or on deployment) for file compatibility, connection health, certificate installation and validity, content freshness etc.

With these methodologies cloud applications can be verified in place in a controlled fashion, or totally invisible to users. Therefore it will be possible to accurately measure quality and impact of new code before making it widely available to public.

2.4 Challenges of Testing Cloud Applications

Cloud applications will be characterized by high level of dynamism. Most decisions, normally made at design time, are deferred to execution time, when the application can take advantage of monitoring (self-observation as well as data collection from the environment and logging of the interactions) to adapt itself to a changed usage context. The realization of this vision involves a number of technologies, including observational reflection and monitoring, dynamic discovery and composition of services and asynchronous communication.

Such features pose several challenges to testing, summarized in Table 2.5.

Table 2.5 Main Testing challenges for Cloud applications

Challenge	Description
Self modification	Rich clients have increased capability to dynamically adapt the structure of the Web pages; server-side services are replaced and recomposed dynamically based on Service Level Agreements (SLA), taking advantage of services newly discovered in the cloud; components are dynamically loaded
Low observability	Cloud applications are composed of an increasing number of 3 rd -party components and services running in the cloud; accessed as a black-box, which are hard to test

Table 2.5 Continued

Asynchronous Interactions	Cloud applications are highly asynchronous and therefore hard to test. Each client submits multiple requests asynchronously; multiple clients run in parallel; server-side computation is distributed over the cloud and concurrent
Ultra-large scale	Since these applications are running in the cloud; traditional testing adequacy criteria cannot be applied, since even in good testing situations low coverage will be achieved.

2.4 Conclusion

Test Lifecycle Management is an important aspect of traditional Software Development Lifecycle practice. It involves many topics and areas of software quality management, which needs a well-structured and carefully designed with a comprehensive execution effort in order to stamp a software product to be ship ready. Some of these areas include functional testing, performance/load/stress testing, security testing, usability testing, reliability/failover testing.

Challenges involved in testing Cloud applications can be addressed by resorting to a combination of advanced testing technologies, including model-based testing, combinatorial testing, concurrent testing, regression testing etc.

There are various tools available to make cloud testing easier. There will be a couple of these tools explained in Chapter 3, along with a simple cloud based application, a web service which resides on the sample cloud platform and sample tests will be developed in order to test this web service application.

CHAPTER THREE

TOOLS AND METHODS FOR CLOUD TESTING

3.1 A Sample Cloud Application to Be Tested

In order to compare different tools for testing cloud applications, a sample application will be designed and developed on a selected cloud platform. The sample application will be an e-mail sending application and will be developed on Windows Azure. In order to get ready for development, a couple of steps required in order to prepare development environment. Visual Studio 2012 with Windows Azure SDK will be used as the development environment.

3.1.1 Preparing Development Environment

In order to start developing Windows Azure application, the following steps should be carried on.

- Install Windows Azure SDK for .NET
- Set up a free Windows Azure account
- Create Storage account in Windows Azure
- Create a Cloud Service in Windows Azure
- Creating a Web Site in Windows Azure

3.1.1.1 Install Windows Azure SDK for .NET

First of all Windows Azure SDK for .NET should be installed on the machine. If there is no Visual Studio 2012 installed, the SDK installs the free Visual Studio 2012 for Web Express version. The SDK can be installed from

<http://go.microsoft.com/fwlink/?LinkId=254364&clid=0x409>

When prompted, click “Run” to install vwdorvs11azurepack.exe. In the Web Platform installer click “Install” to proceed with installation



Figure 3.1 Web Platform Installer to install Windows Azure SDK for Visual Studio

When the installation is complete everything is ready to start development.

3.1.1.2 Set up a Windows Azure Account

In order to be able to deploy the application a free Windows Azure account is required. This can be obtained from www.windowsazure.com by clicking on the “Free trial” link and following the instructions.

3.1.1.3 Create a Windows Azure Storage Account

When running the sample application in Visual Studio, tables, queues and blobs are accessed in Windows Azure development storage or in a Windows Azure Storage

account in the cloud. Development storage uses a SQL Server Express LocalDB database to emulate the way Windows Azure Storage works in the cloud. Following are the steps on how to create the Windows Azure Storage account.

From Windows Azure Management Portal (<http://manage.windowsazure.com/>) after logging in, clicking on the “Storage” and then “New” will display as in the Figure 3.2.

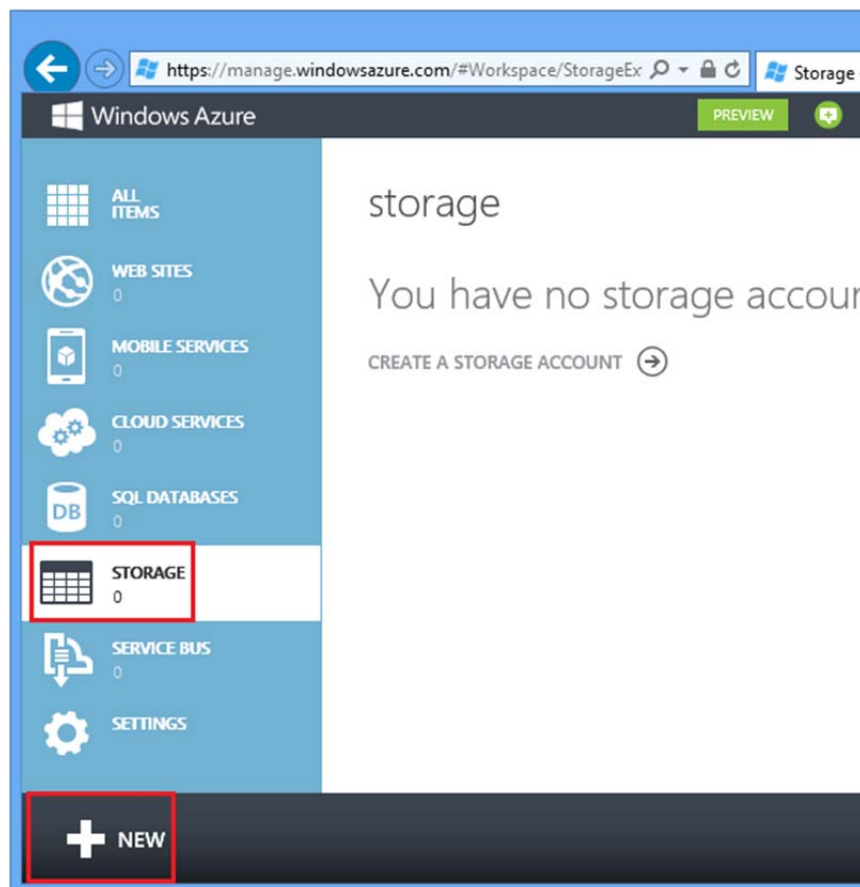


Figure 3.2 Storage screen on Windows Azure

Click “Quick Create” as shown in Figure 3.3.

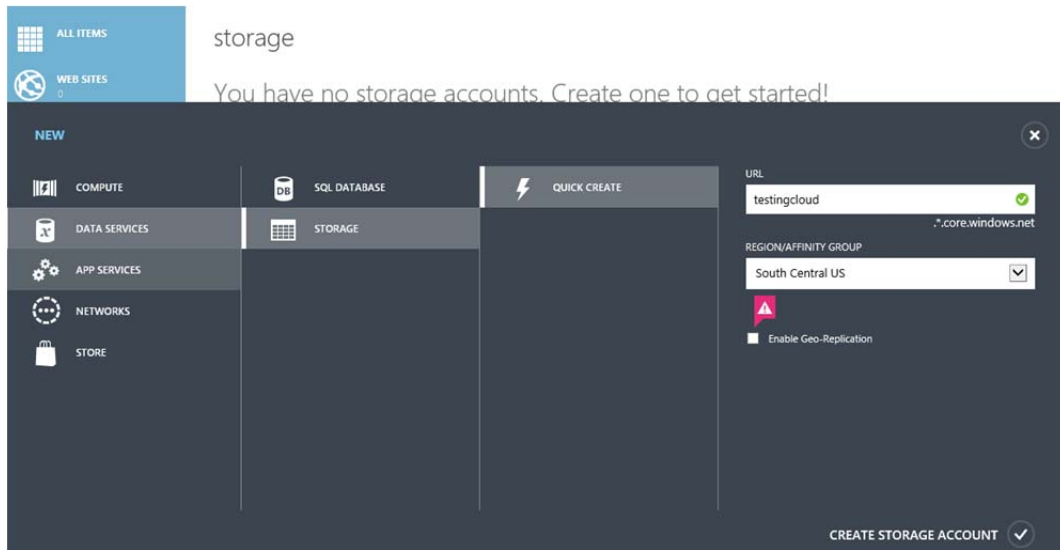


Figure 3.3 Creating storage account in Windows Azure

After this, selecting a URL prefix, which uniquely identifies a storage account and the region to the area where the application will be deployed, clicking on “Create Storage Account” will be completing this step. A storage account with the URL testingcloud.core.windows.net will be created. Once this is created, a primary or secondary key will be required in Azure connection string to be able to access this storage. This can be obtained by clicking on “Manage Keys” in the storage screen as shown in Figure 3.4 and Figure 3.5.

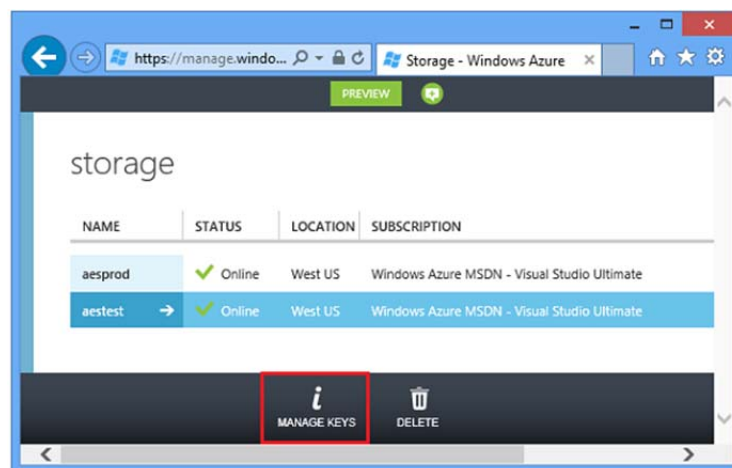


Figure 3.4 Managing keys for storage account in Windows Azure.

There are two keys “Primary Access Key” and “Secondary Access Key” as shown in Figure 3.5 in order to obtain an uninterrupted service in live application when

there is a need to change the keys, or if periodic change of keys is followed for security purposes. One of these two keys will be needed during the application development and will be used within the connection string to connect to the tables, queues and blobs on Windows Azure.

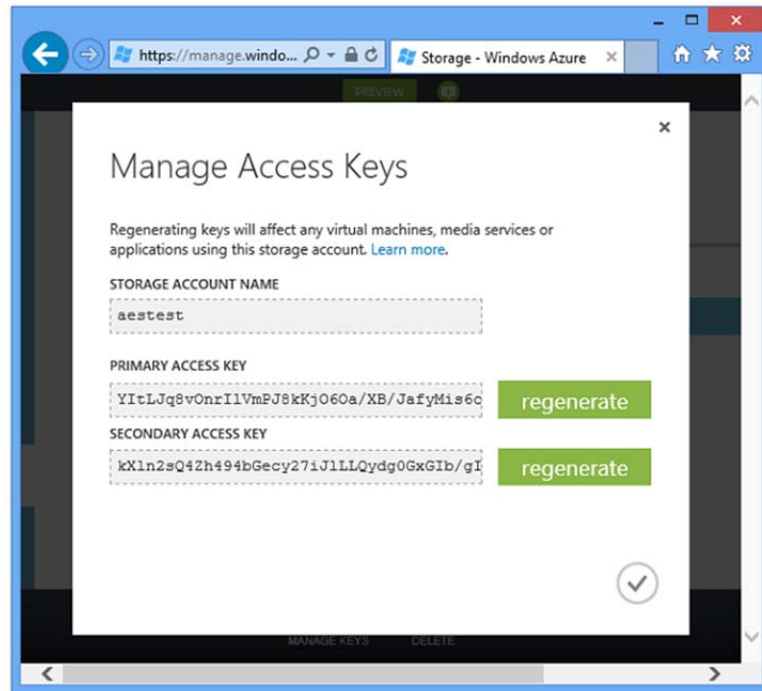


Figure 3.5 “Primary/Secondary Storage Keys” in Windows Azure.

3.1.1.4 Creating Cloud Service in Windows Azure

While on Windows Azure Management Portal, clicking on “Cloud Services” and then “New” will display the “Quick Create” window as shown in Figure 3.6.

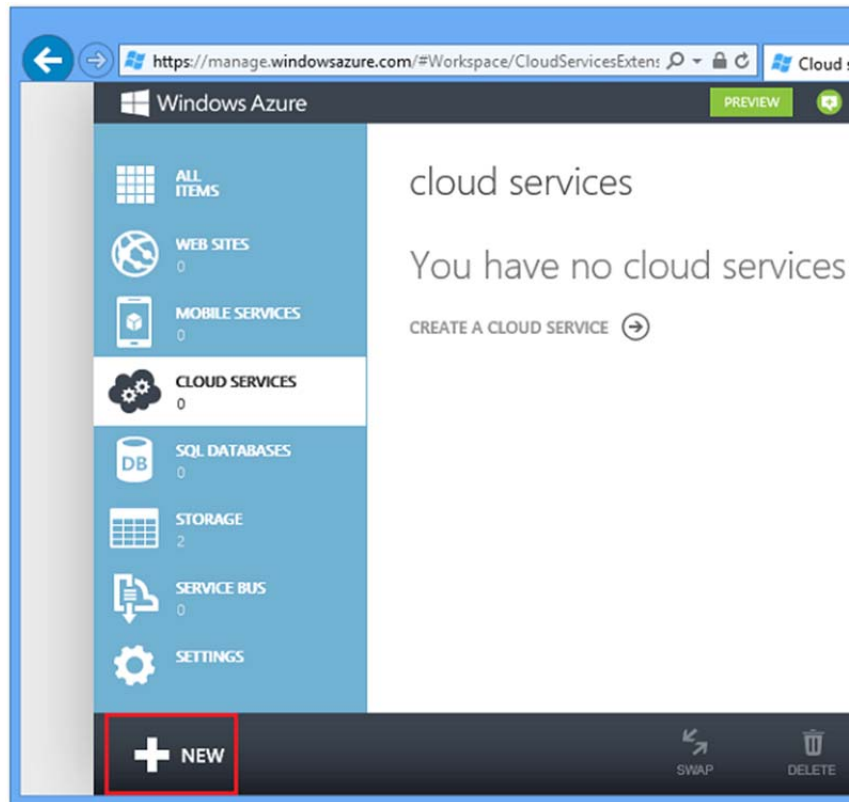


Figure 3.6 Creating a cloud service in Windows Azure.

A unique URL prefix and region for the cloud service is required. The region has to be same as the storage account, otherwise latency will increase and an extra charge will incur for bandwidth outside the data center. Bandwidth within the same data center is free. In Figure 3.7 a cloud service named aescloud.cloudapp.net is created for this purpose.

The image shows a dark-themed dialog box for creating a cloud service. It has a close button (X) in the top right corner. The form contains three main sections: 1. URL: A text input field containing 'aescloud' with a green checkmark on the right. Below the input, the text '.cloudapp.net' is displayed. 2. REGION/AFFINITY GROUP: A dropdown menu showing 'West US' with a downward arrow. 3. SUBSCRIPTION: A dropdown menu showing 'Windows Azure MSDN - Visual Studio Ultimate' with a downward arrow. At the bottom center, there is a button labeled 'CREATE CLOUD SERVICE' with a white checkmark icon.

Figure 3.7 Options for creating cloud service in Windows Azure.

3.1.1.5 Creating a Web Site in Windows Azure

On the Windows Azure Management Portal clicking on “Web Sites” and then “New” initiates the process of creating a new Windows Azure web site as shown in Figure 3.8.

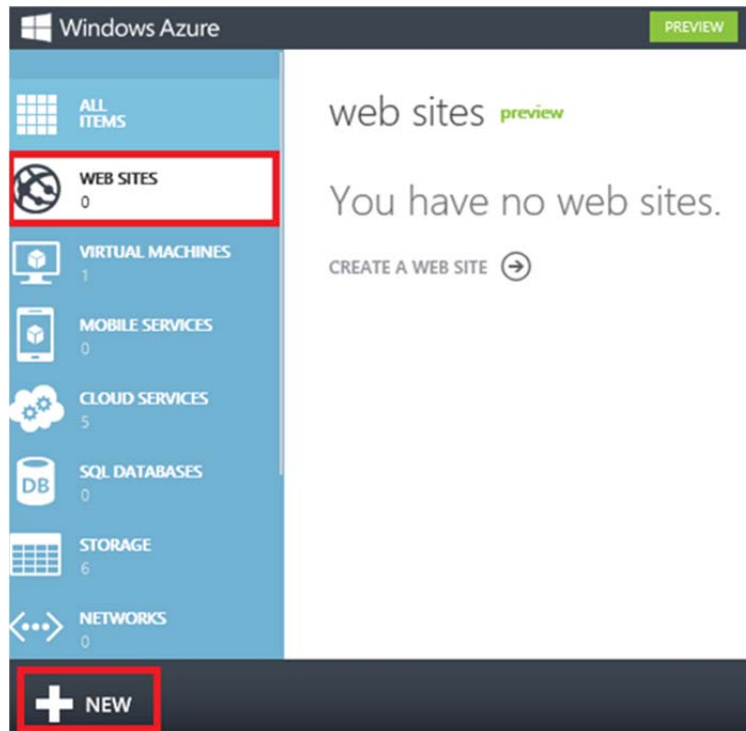


Figure 3.8 Creating a new web site in Windows Azure

After this step, in “Quick Create” screen, as shown in Figure 3.9, a unique URL will be required to identify the Windows Azure web site. After selecting the data center region and clicking “Create Web Site”, a web site will be created which will be used for the sample application.

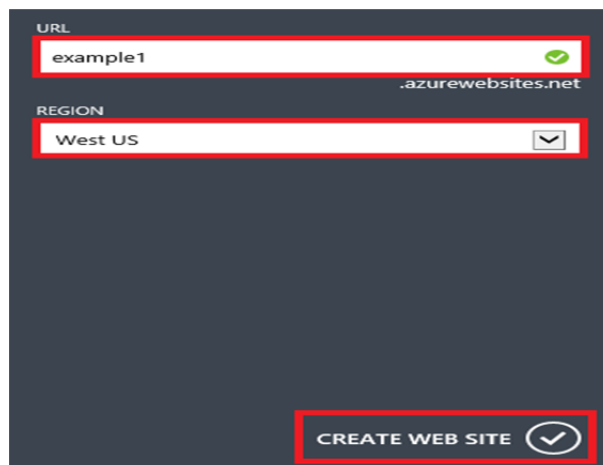


Figure 3.9 Create web site options in Windows Azure

After this step, all the prerequisites for developing a Windows Azure application are completed. The next step is to create a Visual Studio 2012 solution for the sample Windows Azure Application.

3.1.2 Developing Windows Azure Application

In order to upload content to Windows Azure web site created in section 3.1.1 a Visual Studio Web Application will be developed which can be published to Windows Azure.

Within Visual Studio 2012, from the “File” menu, clicking on “New”, and then “Project”, the project type selection screen is shown as in Figure 3.10.

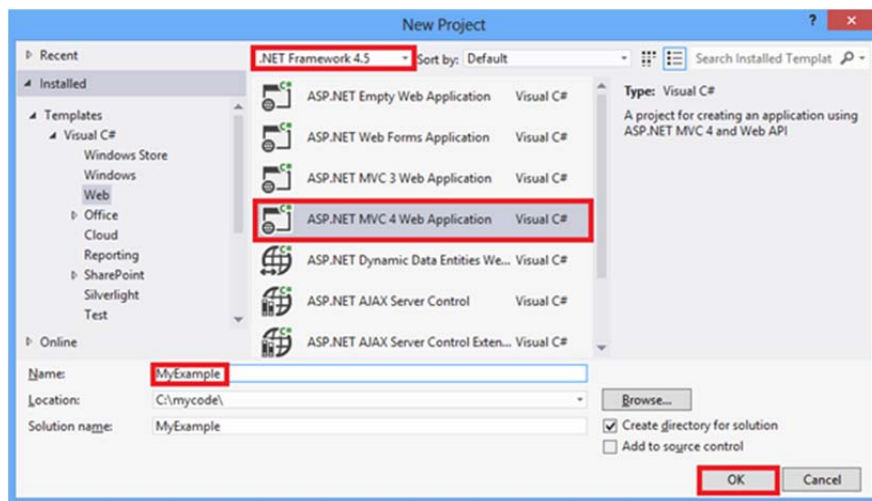


Figure 3.10 Creating a Web Application in Visual Studio 2012

Inputting name “MyExample” as project name and clicking “OK” will display the project template selection screen as in Figure 3.11.

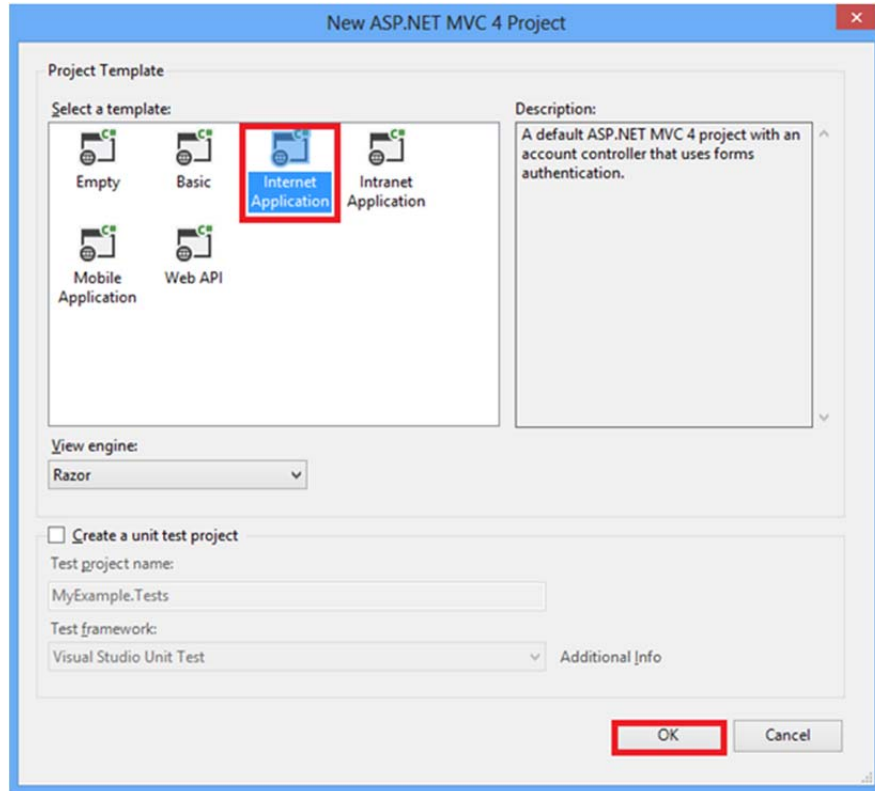


Figure 3.11 Internet project template selections in Visual Studio 2012

After the project is created, pressing Ctrl+F5 runs the application locally which shows the application home page in the default browser as in Figure 3.12.

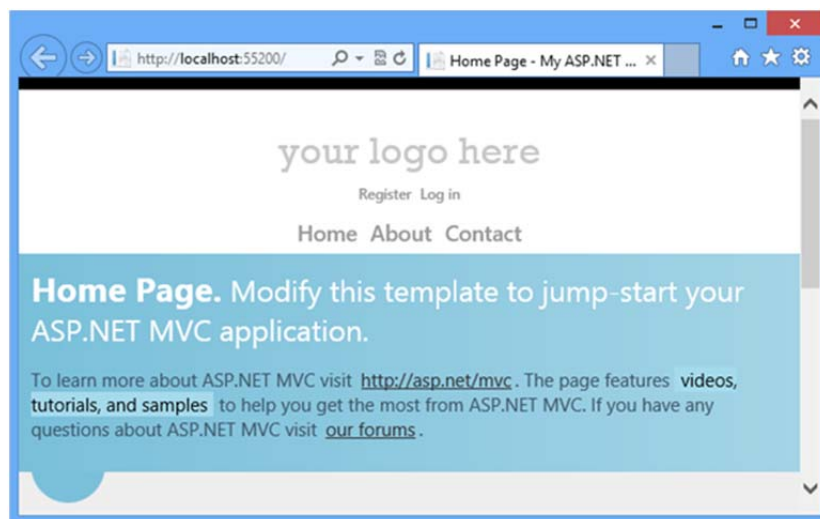


Figure 3.12 Sample application home screen

3.1.3 Deploying Windows Azure Application

After the sample application is created, it needs to be deployed to Windows Azure. This can be done using Windows Azure Management Portal. After logging in to the portal, on the “Web Sites” tab, there will be the site previously created as in Figure 3.13.

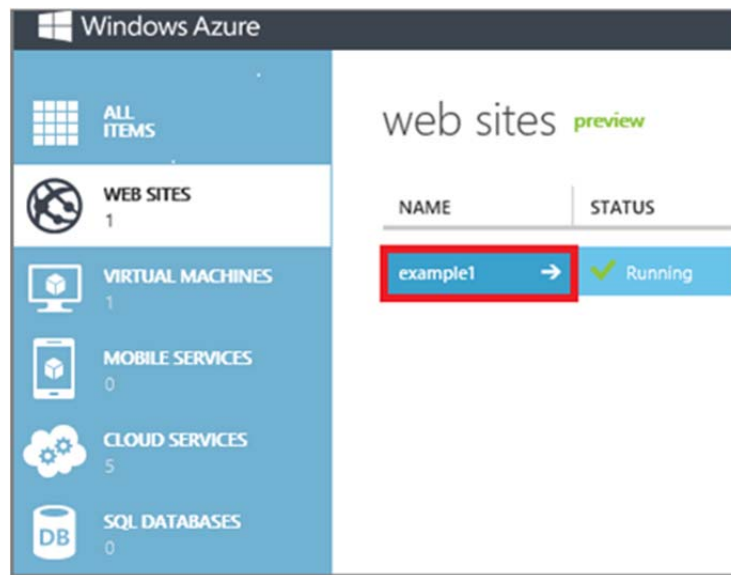


Figure 3.13 Screenshot of the “Web Sites” tab in Windows Azure

Clicking on the web site displays “Dashboard” tab, where under “Quick Glance” section there is a link for downloading publish profile as shown in Figure 3.14.



Figure 3.14 Screenshot of the “Web Site Dashboard” in Windows Azure

After downloading .publishsettings file on the local computer, the Visual Studio project can be configured to use this file to publish the web site application. This can be done via “Publish Web” wizard in Visual Studio as shown in Figure 3.15.

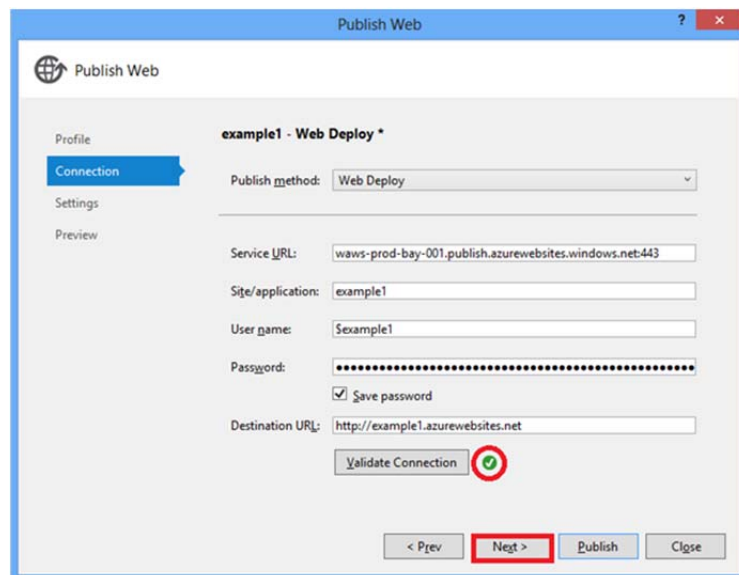


Figure 3.15 Screenshot for Visual Studio publish web wizard

After configuring Visual Studio for publishing to Windows Azure, clicking on “Publish” will start copying all required application files to Windows Azure and browsing to the Windows Azure Web site URL will show the cloud application.

This will be the application that will be used by testing tools including Visual Studio and Selenium to be tested. Next section will be describing step by step how to use these tools and what type of testing can be accomplished using these tools.

3.2 Testing the Sample App with Visual Studio

Visual Studio can be used to simulate user activity on the cloud application. Web performance tests can be used to build a suite of repeatable tests that can help analyze the performance of the cloud applications and identify potential bottlenecks. Visual Studio enables to easily create Web performance tests by recording actions as web application is being used.

Verifying that a cloud application is ready to be published involves additional analysis. How will the application behave when many people begin using it concurrently? The load testing features of Visual Studio enables to execute one or more tests repeatedly, tracking the performance of the target system. It can also be used to configure the environment to run distributed load tests. A distributed load test enables to spread the work of creating user load across multiple machines, called agents. Details from each agent are collected by a controller machine, which can be used to see the overall performance of the application under stress.

3.2.1 Web Performance Tests

Web performance tests enable verification that a Web application’s behavior is correct. They issue an ordered series of HTTP/HTTPS requests against a target cloud application, and analyze each response for expected behaviors. Integrated “Web Test Recorder” can be used to create a test by observing the interaction with a target cloud

site through a browser window. Once the test is created, the recorded actions can be consistently replayed against the target cloud application.

Web performance tests offer automatic processing of redirects, dependent requests, and hidden fields, including “ViewState”. In addition, coded Web performance tests can be written in Visual Basic or C#, enabling to take the full advantage of the power and flexibility of these languages.

Web performance tests can be used primarily for performance testing, and can be used as the basis for generating load tests.

3.2.2 Creating and Configuring Web Tests

There are three ways to create Web Tests in Visual Studio. The most common way is using the “Web Test Recorder”. The second way is to create a test manually, using the “Web Test Editor” to add each step. This is time consuming and error-prone, but allows more fine-tuning Web performance tests. Finally a coded Web performance test can be created, that specifies each action via code, and offers a great deal of customization.

To create a new Web performance test, a new test project needs to be created as shown in Figure 3.16.

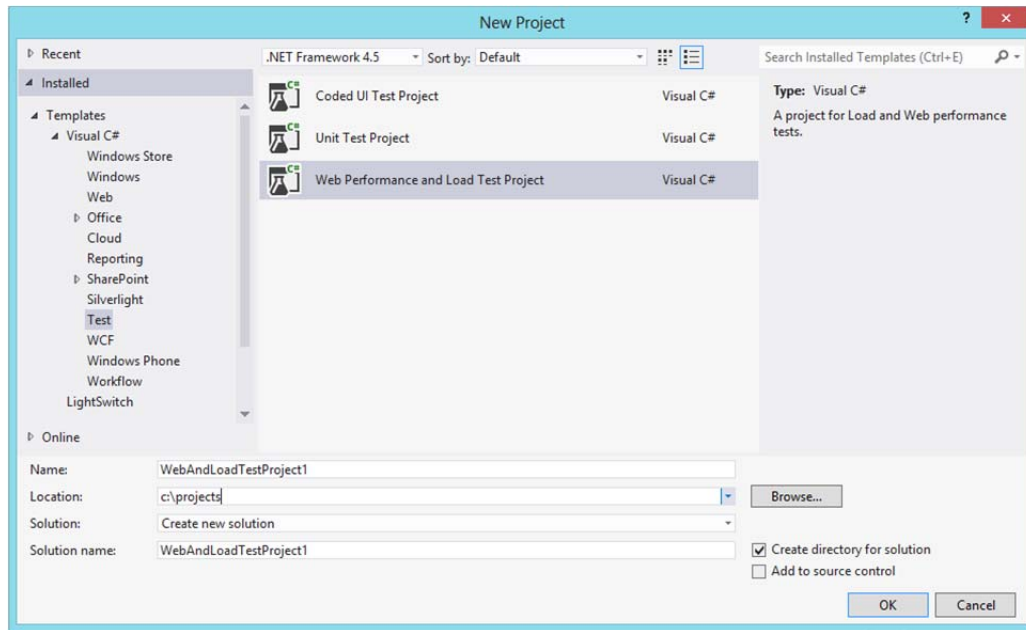


Figure 3.16 Creating new Test Project in Visual Studio

Web performance tests are stored as XML files with a “.webtest” extension.

3.2.2.1 Recording a Web Performance Test

In Visual Studio, after creating test project right-clicking on test project node in “Solution Explorer” window and selecting “Add-Web Performance Test” menu, the Web Test will be added to the project and the default browser will be loaded to start the web recording. Double-clicking on the existing Web Performance Test and then clicking on the “Add Recording” button will also load the default web browser.

After this navigating to the cloud application created in section 3.1.3 will display the application on the browser window. Using the web browser, running normal usage scenarios will let Visual Studio to record the actions and save them to the web performance test as in Figure 3.17. The Web Test Recorder provides several options that may be useful while recording. The “Pause Button” in the upper-left corner temporarily suspends recording and timing of the interaction with the browser. “Add a comment” button enables to add documentation to the Web Performance test which is very useful when the recorded test is converted to coded Web Performance test.

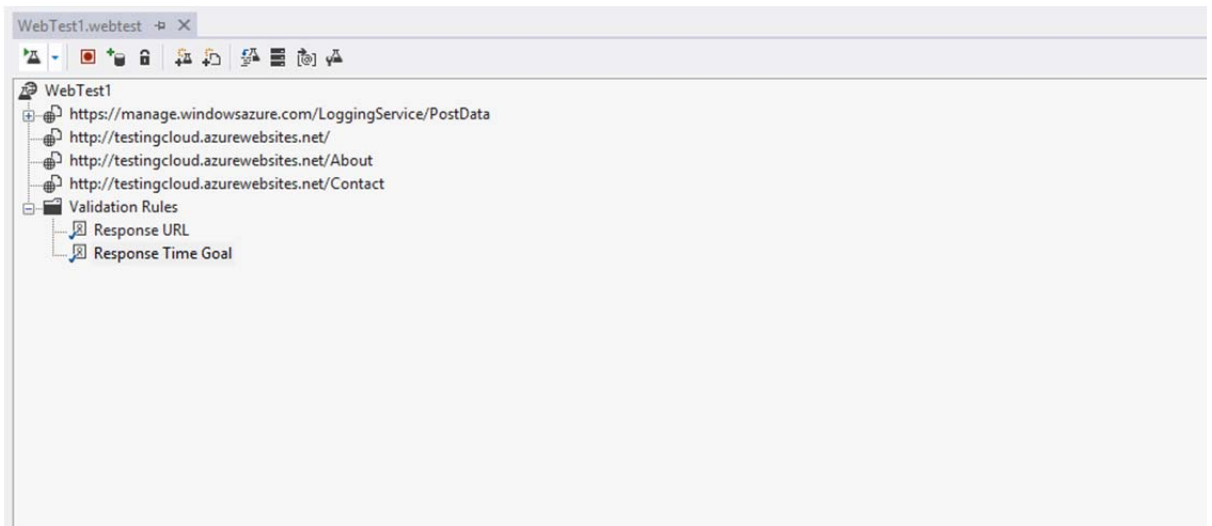


Figure 3.17 Result of Web Performance Test recording in Visual Studio

It is possible to configure the test for advanced settings. “Fix run count” setting in the “Test Settings” window enables to specify number of times the Web performance tests will be executed when included in a test run. Running the test a few times can help eliminate errant performance timings caused by system issues on the client or server, and can help derive a better estimate for how the cloud application is actually performing. However, this should not be set to a large number to simulate load. Instead a load test should be created referencing this web performance test.

The browser type setting enables to simulate using a number of browsers as Web performance test’s client. This will automatically set the user agent field for requests sent to the Web performance test to simulate the selected browser. However this will not help to determine if the cloud application will render as desired in a given browser type, since Web performance tests only examine HTTP/HTTPS responses and not the actual renderings of pages. Changing the browser type is only important if the cloud application responds differently based on the user agent sent by the client. For example, a cloud application may send a more lightweight user interface to a mobile device than it would to a desktop computer.

The final option, “Simulate think times”, enables the use of delays in the Web performance test to simulate the normal time taken by users to read content, modify values, and decide on actions. When the web test is recorded, the time it took to

submit each delay will occur between the requests sent by the Web performance test to the cloud application. Think times are disabled by default, causing all requests to be sent as quickly as possible to the cloud application, resulting in a faster test. Think times are important for load tests.

Visual Studio also allows emulating different network speeds for the tests. From within “Test Settings” the option “Data and Diagnostics” on the left will bring this window. Enabling the network emulation adapter and clicking configure, a variety of network speeds (such as dial-up 56K connection) can be selected to examine the effect that slower connection speeds have on the cloud application.

The XML output of the generated Web performance test is shown in Table 3.1. Closely observing the Web performance test XML code there will be <request> tags with the methods and the URL value for each tag.

Table 3.1 XML code behind for web performance test

```
<?xml version="1.0" encoding="utf-8"?>
<WebTest Name="WebTest1" Id="6e65ba0d-2b50-4ac0-9fe3-2b90db7adcd8" Owner="" Priority="2147483647"
Enabled="True" CssProjectStructure="" CssIteration="" Timeout="0" WorkItemIds=""
xmlns="http://microsoft.com/schemas/VisualStudio/TeamTest/2010" Description="" CredentialUserName=""
CredentialPassword="" PreAuthenticate="True" Proxy="" StopOnError="False"
RecordedResultFile="WebTest1.6ea15a9e-cdea-4605-9515-82989a55b442.rec.webtestresult">
<Items>
<Request Method="GET" Guid="54b611f3-442c-4895-8574-bb23106fe2a5" Version="1.1"
Url="http://testingcloud.azurewebsites.net/" ThinkTime="4" Timeout="300" ParseDependentRequests="True"
FollowRedirects="True" RecordResult="True" Cache="False" ResponseTimeGoal="0" Encoding="utf-8"
ExpectedHttpStatusCode="0" ExpectedResponseUrl="" ReportingName="" />
<Request Method="GET" Guid="08d0962b-e483-4e3a-8fb1-a8b54d4b7b66" Version="1.1"
Url="http://testingcloud.azurewebsites.net/About" ThinkTime="4" Timeout="300" ParseDependentRequests="True"
FollowRedirects="True" RecordResult="True" Cache="False" ResponseTimeGoal="0" Encoding="utf-8"
ExpectedHttpStatusCode="0" ExpectedResponseUrl="" ReportingName="" />
<Request Method="GET" Guid="377ebb5a-39b3-4422-bbbc-af6a6378528a" Version="1.1"
Url="http://testingcloud.azurewebsites.net/Contact" ThinkTime="0" Timeout="300"
ParseDependentRequests="True" FollowRedirects="True" RecordResult="True" Cache="False"
ResponseTimeGoal="0" Encoding="utf-8" ExpectedHttpStatusCode="0" ExpectedResponseUrl=""
ReportingName="" />
</Items>
<ValidationRules>
<ValidationRule Classname="Microsoft.VisualStudio.TestTools.WebTesting.Rules.ValidateResponseUrl,
Microsoft.VisualStudio.TestTools.WebTestFramework, Version=10.0.0.0, Culture=neutral,
PublicKeyToken=b03f5f7f11d50a3a" DisplayName="Response URL" Description="Validates that the response URL
after redirects are followed is the same as the recorded response URL. QueryString parameters are ignored."
Level="Low" ExecutionOrder="BeforeDependents" />
<ValidationRule
Classname="Microsoft.VisualStudio.TestTools.WebTesting.Rules.ValidationRuleResponseTimeGoal,
Microsoft.VisualStudio.TestTools.WebTestFramework, Version=10.0.0.0, Culture=neutral,
PublicKeyToken=b03f5f7f11d50a3a" DisplayName="Response Time Goal" Description="Validates that the response
time for the request is less than or equal to the response time goal as specified on the request. Response time goals of
zero will be ignored." Level="Low" ExecutionOrder="AfterDependents">
```

3.2.3 Running and Observing Results for Web Performance Tests

Selecting Web Performance Test and clicking “Run” on the toolbar will simply execute the Web performance test, and re-run all the steps recorded in the Web performance test. When the test run is started, a window specific to that Web performance test execution will appear. The results will automatically be displayed as shown in Figure 3.18. Web performance test can also be debugged, running it one request at a time. The results will be summarized in the “Test Results” window docked at the bottom of the screen. Double-clicking on the Web performance test’s entry in the “Test Results” window will display the test’s execution details.

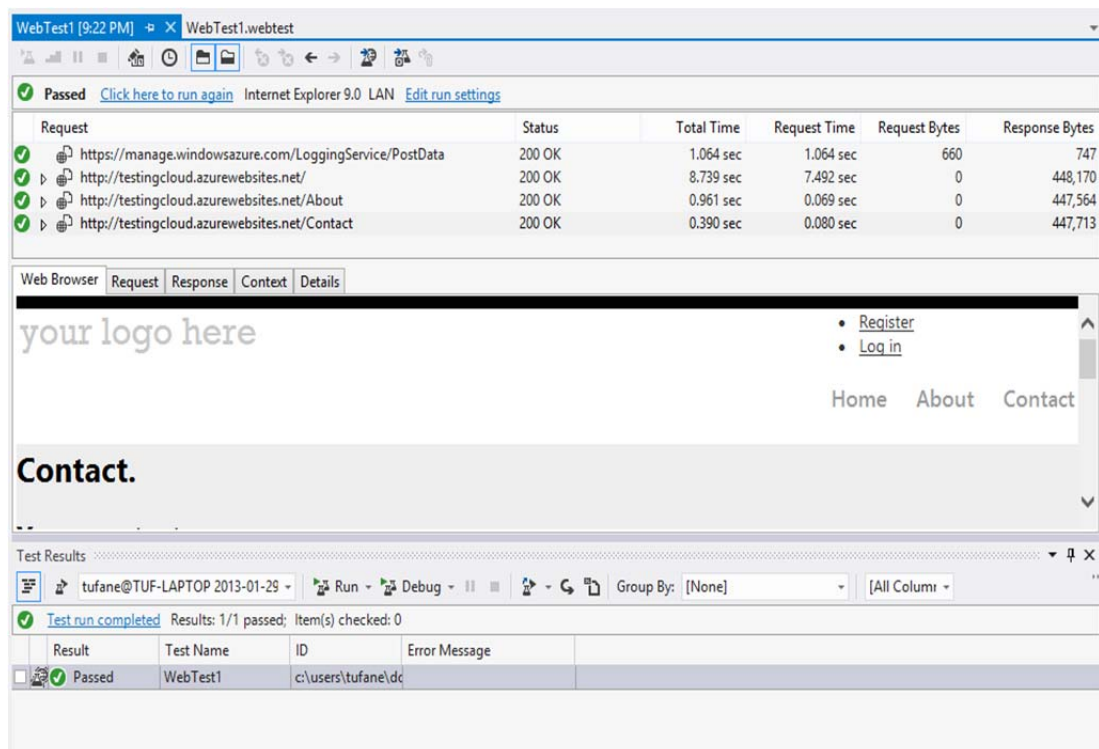


Figure 3.18 Results of the execution of Web Performance Test

3.2.4 Coded Web Performance Tests

In order to control over the actions that are taken, coded web performance tests can be developed. Recorded Web performance tests are stored as XML files with .webtest extension as shown in Table 3.1. Visual Studio uses this XML to generate the code that is executed when the Web performance test is run.

Coded Web performance test enables to perform actions not possible with a standard Web performance test. For example, branching can be performed based on the responses received during a Web performance test or based on values of a data-bound test. A coded Web performance test inherits from either a base with “WebTest” class for C# tests, or from a “ThreadedWebTest” base for Visual Basic tests. These classes are in the “Microsoft.VisualStudio.TestTools.WebTesting” namespace. A coded Web performance test can be developed from the scratch, or it can be generated using an existing recorded Web performance test. In Table 3.2 there is the code generated for the Web performance test which was created in the previous section.

Table 3.2 Coded Web performance test in Visual Studio

```

public override IEnumerator<WebTestRequest> GetRequestEnumerator()
{
    // Initialize validation rules that apply to all requests in the
    WebTest
    if ((this.Context.ValidationLevel >=
Microsoft.VisualStudio.TestTools.WebTesting.ValidationLevel.Low))
    {
        ValidateResponseUrl validationRule1 = new
ValidateResponseUrl();
        this.ValidateResponse += new
EventHandler<ValidationEventArgs>(validationRule1.Validate);
    }
    if ((this.Context.ValidationLevel >=
Microsoft.VisualStudio.TestTools.WebTesting.ValidationLevel.Low))
    {
        ValidationRuleResponseTimeGoal validationRule2 = new
ValidationRuleResponseTimeGoal();
        validationRule2.Tolerance = 0D;
        this.ValidateResponseOnPageComplete += new
EventHandler<ValidationEventArgs>(validationRule2.Validate);
    }

    WebTestRequest request1 = new
WebTestRequest("http://testingcloud.azurewebsites.net/");
    request1.ThinkTime = 4;
    yield return request1;
    request1 = null;

    WebTestRequest request2 = new
WebTestRequest("http://testingcloud.azurewebsites.net/About");
    request2.ThinkTime = 4;
    yield return request2;
    request2 = null;

    WebTestRequest request3 = new
WebTestRequest("http://testingcloud.azurewebsites.net/Contact");
    yield return request3;
    request3 = null;
}

```

The “GetRequestEnumerator” method uses “yield” statement to provide “WebTestRequest” instances, one per HTTP request, back to the Web test system. Visual Basic test projects generate slightly different code than C# tests because Visual Basic does not currently support iterators and the “yield” statement. Instead of having a “GetRequestEnumerator” method that yields “WebTestRequest” instances one at a time, there is a “Run” subroutine that uses the base “ThreadedWebTest.Send” method to execute each request.

3.2.4 Load Tests

Load tests are used to verify that the application will perform as expected while under the stress of multiple concurrent users. Configuring the levels and types of load that is targeted to simulate and then the test is executed. A series of requests will be generated against the target cloud application, and Visual Studio will monitor the system under test to determine how well it performs.

Load testing is used with Web performance tests to conduct smoke, load and stress testing of cloud applications.

3.2.4.1 Creating and Configuring Load Tests

In order to create a load test, “New Load Test Wizard” can be used. When a new load test is added, the “New Load Test Wizard” is started. This wizard guides through the many configuration options available for a load test.

3.2.4.1.1 *Scenarios and Think Times*. A load test is composed of one or more scenarios. A scenario is a grouping of Web performance and/or unit tests, along with a variety of preferences for user, browser, network, and other settings. Scenarios are used to group similar tests or usage environments. When the “New Load Test Wizard” is launched, the first screen describes the load test creation process. Clicking “Next” will prompt to assign a name for the load test’s first scenario as shown in Figure 3.19.

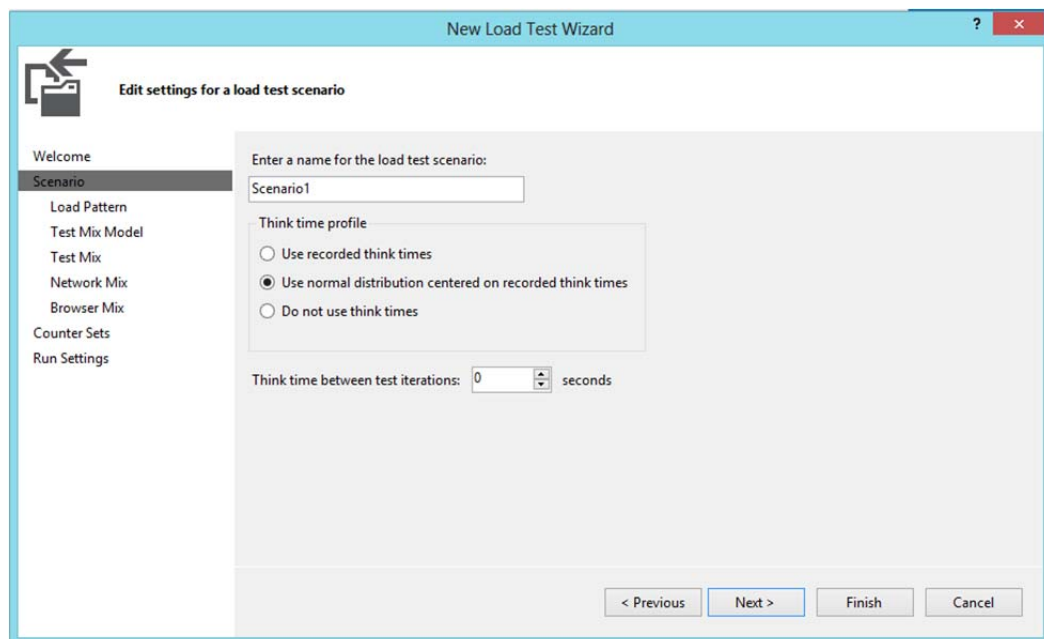


Figure 3.19 First page of “New Load Test Wizard”

The second option on this page is to configure think times. Think time is a delay between each request, which can be used to approximate how long a user will pause to read, consider options, and enter data on a particular page. These times are stored with each of a Web performance test’s requests. The think time profile panel enables to turn these on or off.

If it is enabled, think times can be used as is, or apply a normal distribution that is centered around recorded think times as a mean. The normal distribution is generally recommended if simulation of the most realistic user load is wanted, based on what to expect the average user does. Think times can also be configured between test

iterations to model a user who pauses after completing a task before moving to the next task.

3.2.4.1.2 Load Patterns. The load pattern enables simulation of different types of user load. There are two load pattern options in the wizard: Constant and Step, as shown in Figure 3.20.

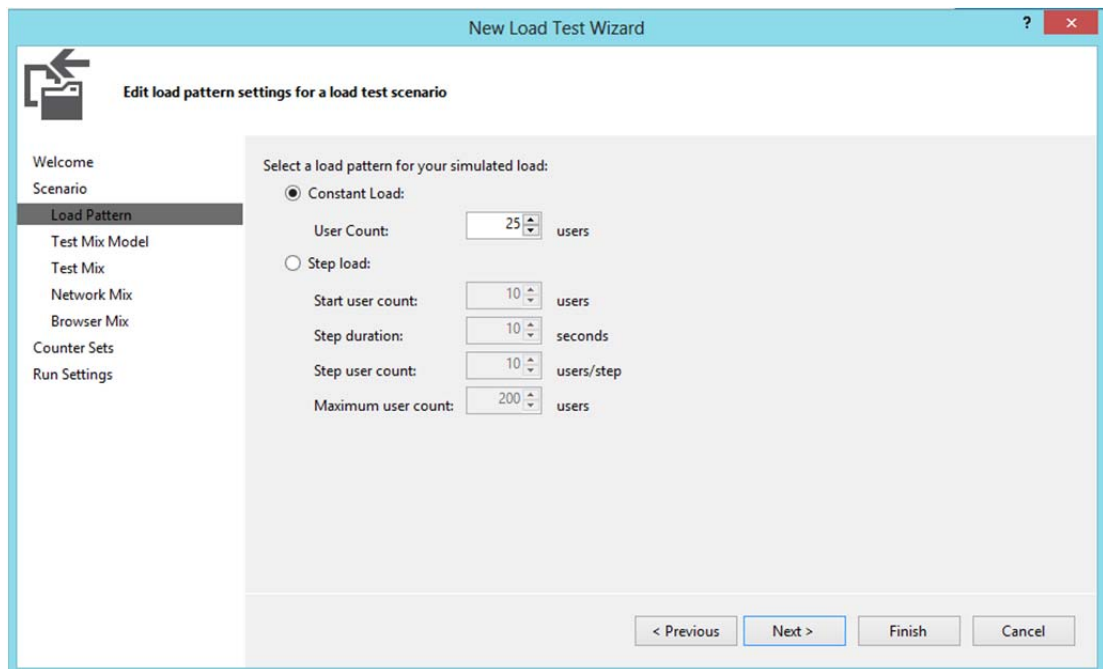


Figure 3.20 Load Pattern options in “New Load Test Wizard”

A constant load enables to define a number of users that will remain unchanged throughout the duration of the test. In order to analyze the performance of the cloud application under a steady load of users this is the option to use.

A step load defines a starting and maximum user count. Step duration and step user count can also be assigned. Every time the number of seconds specified in the step duration elapse, the number of users is incremented by the step count, unless the maximum number of users has been reached. Step loads are very useful for stress-testing the cloud application, finding the maximum number of users the cloud application will support before serious issues arise.

A third type of load profile pattern called “Goal Based” is available through the “Load Test Editor” after the load test is created. The goal-based pattern is used to raise or lower the user load over time until a specific performance counter range has been reached. This is useful when the peak loads of the cloud application can withstand is needed. It has similar options as step pattern, where there is initial and maximum user count. But there is also a maximum user count increment and decrement and minimum user count. The load test will dynamically adjust the current user count according to these settings in order to reach the goal performance counter threshold.

By default, the pattern will be configured against the % Processor Time performance counter. To change this, enter the category (for example, Memory, System, and so on), the computer from which it will be collected (leave this blank for the current machine), and the counter name and instance which is applicable if there are multiple processors.

Then the test must be informed about the performance counter selected. First, identify the range to reach using the High-End and Low-End properties. Set the Lower Values Imply Higher Resource Utilization option if a lower counter value indicates system stress. For example, this needs to be set to “True” when using the system group's Available M Bytes counter. Finally, the load test can be configured to remain at the current user load level when the goal is reached with the Stop Adjusting User Count When Goal Achieved option.

3.2.4.1.3 Test Mix Model. The test mix model determines the frequency at which tests within the load test will be selected from among other tests within the load test as shown in Figure 3.21. The test mix model allows several options for realistically modeling user load.

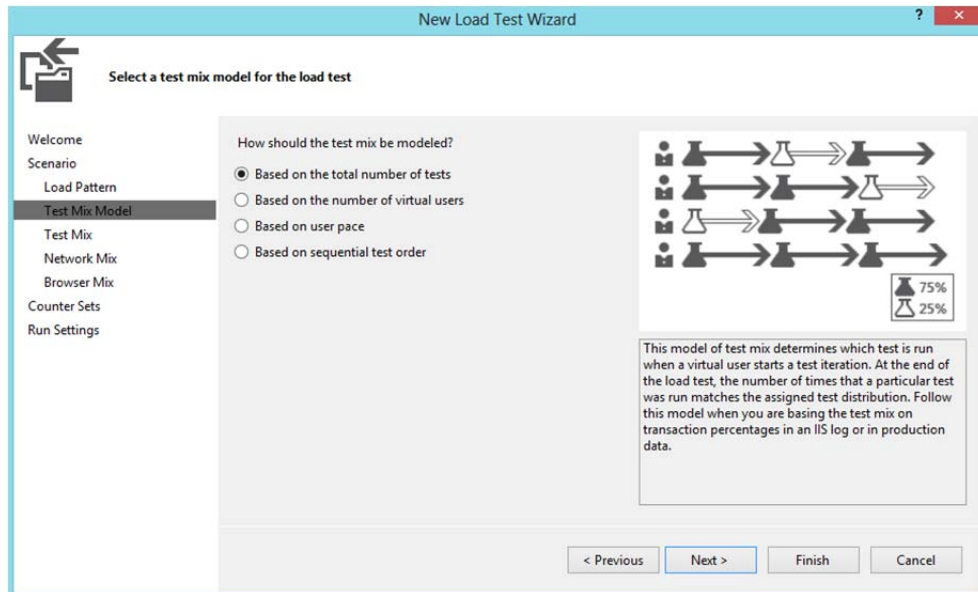


Figure 3.21 Test mix model in “New Load Test Wizard”

The options for test mix model are as follows.

- Based on the total number of tests: This model assigns a percentage to each test that dictates how many times it should be run. Each virtual user will run each test corresponding to the percentage assigned to that test.
- Based on the number of virtual users: This model assigns a percentage of virtual users who should run each test.
- Based on user pace: This model executes each test a specified number of times per virtual user per hour. When using this model, the think time between iterations value from the Scenario page of the wizard is ignored.
- Based on sequential test order: If users are performing steps in a specific order (for example, logging in, then finding an item to purchase, then checking out) you can use this test mix model to simulate a sequential test behavior for all virtual users. This option is functionally equivalent to structuring tests as ordered tests.

3.2.4.1.4 Test Mix. This page of the wizard enables to select the tests to include in the scenario, along with the relative frequency with which they should run.

3.2.4.1.5 *Network Mix*. Network mix specifies kinds of network connectivity the users expected to have, such as LAN, DSL, Cable and Dial-up as shown in Figure 3.22.

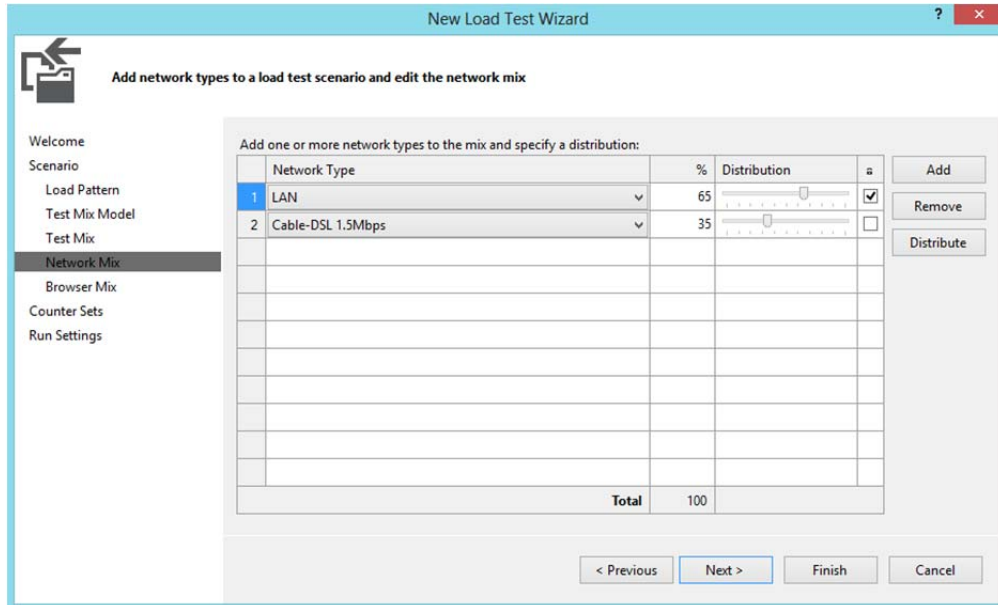


Figure 3.22 Network mix selection in “New Load Test Wizard”

Sliders can be used to adjust the percentages, lock a particular percent, or clicking on “Distribute” button resets to an even distribution. As with the test mix settings, each virtual user will select a browser type at random according to the percentages you set. A new browser type is selected each time a test is chosen for execution. This also applies to the browser mix.

3.2.4.1.6 *Browser Mix*. The next step in the wizard defines the distribution of browser types that will be simulated. Visual Studio then adjusts the headers sent to the target application according to the selected browser for that user. As shown in Figure 3.23, different types of browsers can be selected with a percent distribution for their use.

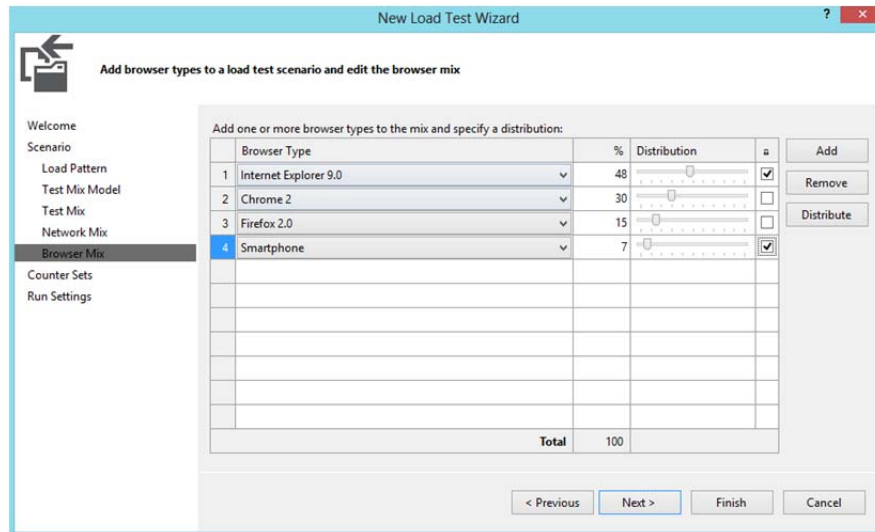


Figure 3.23 Browser distribution page for New Load Test Wizard

3.2.4.1.6 Performance Counter Sets. A vital part of load testing is the tracking of performance counters. Load test can be configured to observe and record the values of performance counters, even on remote machines. If target application is hosted on a different machine, which is the case for cloud applications, counters from these machines can be collected and stored by Visual Studio. However this requires that the cloud-application be deployed on IAAS (Infrastructure as a Service) type cloud system, not PAAS (Platform as a Service). With IAAS, a virtual machine is hosted on cloud system, along with the application, for example IIS Web Site, where Visual Studio can collect counters.

A counter set is a group of related performance counters. All of the contained performance counters will be collected and recorded on the target machine when the load test is executed.

3.2.4.1.7 Run Settings. As the final step in the “New Load Test Wizard”, test run settings can be specified as shown in Figure 3.24.

First the timing detail for the test is selected. "Warm-up duration" specifies a window of time during which (although the test is running) no information from the test is tracked. This gives the target application a chance to complete actions such as just-in-time (JIT) compilation or caching of resources. Once the warm-up period

ends, data collection begins and will continue until the "Run duration" value has been reached.

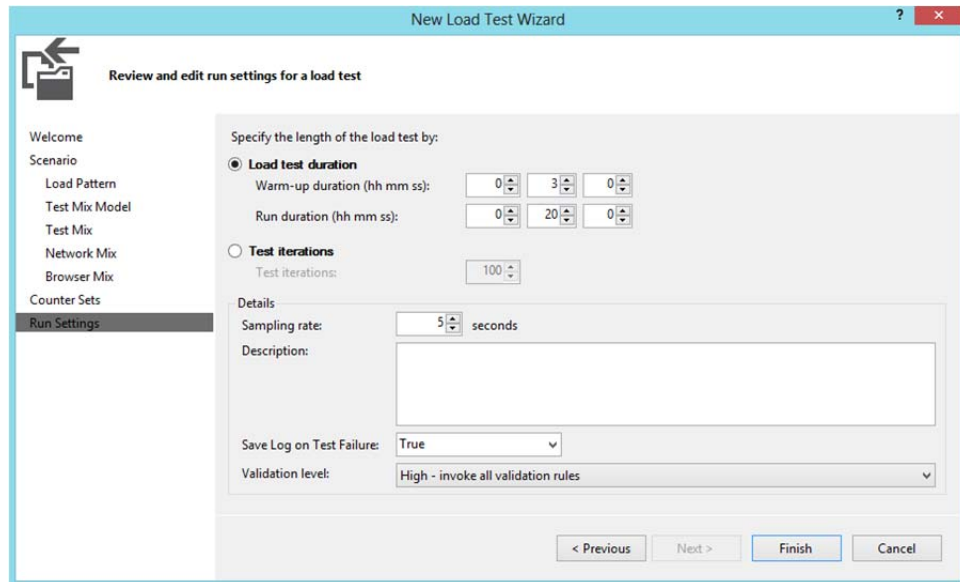


Figure 3.24 Run settings page for “New Load Test Wizard”

The "Sampling rate" determines how often performance counters will be collected and recorded. A higher frequency (lower number) will produce more detail, but at the cost of a larger test result set and slightly higher strain on the target machines.

Any description entered will be stored for the current run settings. “Save Log on Test Failure” specifies whether or not a load test log should be saved in the event that tests fail.

Finally, the "Validation level" setting indicates which Web performance test validation rules should be executed. This is important, because the execution of validation rules is achieved at the expense of performance. In a stress test, raw performance might be more interesting than a set of validation rules pass. There are three options for validation level:

- Low – Only validation rules marked with “Low” level will be executed.
- Medium – Validation rules marked “Low” or “Medium” level will be executed.

- High – All validation rules will be executed

3.2.4.2 Executing Load Tests and Viewing Results

In order to execute load tests, various windows in Visual Studio can be used, “Load Test Editor”, “Test Manager” and “Test View”, or command-line tools can be used.

If the test is executed from either “Test Manager” or “Test View”, the status of the test can be seen in the “Test Results” window as shown in Figure 3.24.

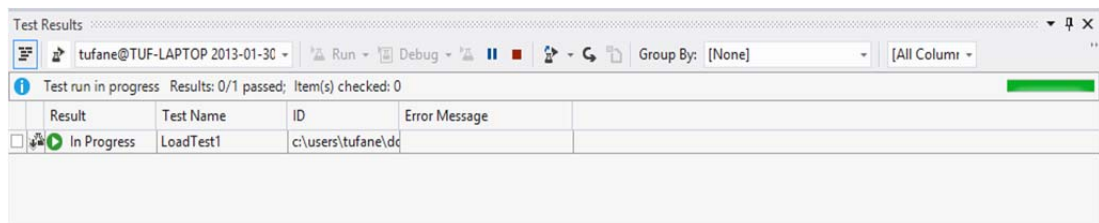


Figure 3.25 Test Results for Load Test

Once the status of the test is in Progress or Complete, the “Load Test Monitor” is shown as in Figure 3.25.

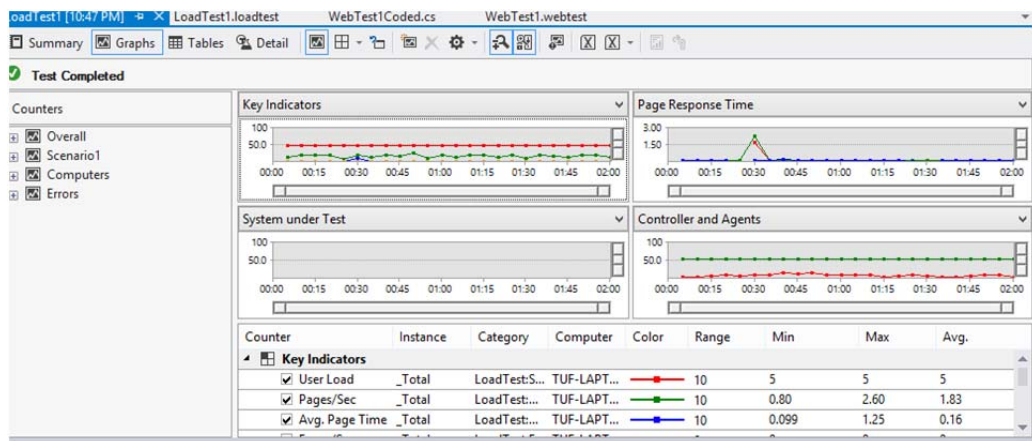


Figure 3.26 Load Test monitor in Visual Studio

The progress of the test can be observed and then same window can be used to review results after the test has completed. There are four graphs which are selected

by default. These graphs plot a number of selected performance counters over the duration of the test.

The tree in the left-hand (“Counters”) pane shows a list of all available performance counters, grouped into a variety of sets – for example by machine. As the load test runs, the graph is updated at each snapshot interval. If the value of a performance counter exceeds a predefined threshold then the corresponding node in the “Counters” pane is marked with a red error or yellow warning icon. The list at the bottom of the screen is a legend that shows details of the selected counters. Those that are checked appear in the graph with the indicated color. If a counter is selected, it will be displayed with a bold line.

3.3 Summary for Visual Studio Tools

Visual Studio delivers strong and integrated tooling support for testing cloud based applications. As discussed in the sections above, below are the testing highlights that are performed by Visual Studio test tools suite on cloud applications for better performance and availability:

- Validation and verification test: Visual Studio tools can be used to help verifying the inputs or the expected entries that satisfy the requirements.
- Web page usability test: Visual Studio test tools can be used to simulate the user’s way of experiencing the application in production, and testing the same as per requirement.
- Security testing: Visual Studio test tools can help verifying the application response for different end users based on the credentials and different other resources required from the local system or a server in the network.
- Performance testing: Visual Studio test tools can help verifying web page responses as per expectations based on the environment. This also includes stress testing and load testing of the application with multiple user scenarios.

- Application compatibility testing: Visual Studio test tools can help testing the application with multiple browsers based on user requirements.
- Application testing using different networks: Visual Studio test tools can help verifying the applications behavior depending on the network involved in providing the cloud application to the user.

However there are also limitations where Visual Studio testing tools doesn't support out of the box when considering data security testing as described in Section 2.2. Verifying the sensitive data travelling through cloud is safe and secure requires additional tools or framework.

Failover testing is another type of testing which is not supported out of the box by Visual Studio.

3.4 Testing the Sample App with Selenium

Selenium is a set of different software tools each with a different approach to supporting test automation. The entire suite of tools results in a rich set of testing functions specifically geared to the needs of testing of web applications of all types. These operations are highly flexible, allowing many options for locating UI elements and comparing expected results against actual application behavior.

3.4.1 Brief history of the Selenium Project

Selenium first came to life in 2004 when Jason Huggins developed a JavaScript library that could drive interactions with a web page, allowing to automatically rerunning tests against multiple browsers. That library eventually became Selenium Core, which underlies all the functionality of Selenium Remote Control (RC) and Selenium IDE. Selenium RC was ground breaking because no other product allowed a developer to control a browser from a language of choice.

However, because of the security limitations browsers apply to JavaScript different things became impossible to do. To make things worse, web apps became more and more powerful over time, using all sorts of special features new browsers provide and making these restrictions more and more painful.

In 2006, an engineer from Google, Simon Stewart, started working on a project called WebDriver. Google had long been a heavy user of Selenium, but testers had to work around the limitations of the product. Simon wanted a testing tool that spoke directly to the browser using the native method for the browser and the operating system, thus avoiding the restrictions of a sandboxed Javascript environment. The WebDriver project began with the aim to solve the Selenium pain points.

In 2008, Selenium and WebDriver projects merged which provided a common set of features for all users.

With this Selenium 2, also known as Selenium WebDriver, provides features like including more cohesive and object oriented API as well as an answer to the limitations of the old implementation.

Selenium 1, also known as Selenium RC or Remote control, provides some features that may not be available in Selenium 2, including support for several languages (Java, Javascript, Ruby, PHP, Python, Perl, and C#) and support for almost all types of browsers.

3.4.2 Selenium RC Architecture

The main restriction that Selenium faces is the “Same Origin Policy”. This security restriction applied by every browser in the market and its objective is to ensure that a site’s content will never be accessible by a script from another site. It cannot perform actions on another website. This is to prevent Cross-site scripting (XSS) so that a malicious site opened on one tab of the browser cannot read the information of a bank account opened on another tab.

To work within this policy, Selenium-Core and its JavaScript commands must be placed in the same origin (URL) as the application under test. Selenium-Core was limited by this problem since it was implemented in JavaScript. Selenium RC is not, however, restricted by the “Same Origin Policy”. Its use of the Selenium Server as a proxy avoids this problem. It, essentially, tells the browser that the browser is working on a single “spoofed” website that the Server provides.

3.4.2.1 Proxy Injection

The first method Selenium used to avoid the “Same Origin Policy” was Proxy injection. In Proxy Injection Mode, the Selenium Server acts as a client-configured “HTTP Proxy” that sits between the browser and the application under test. It then masks the AUT under a fictional URL, embedding Selenium-Core and the set of tests and delivering them as if they were coming from the same origin. Figure 3.27 shows an architecture diagram for Selenium RC

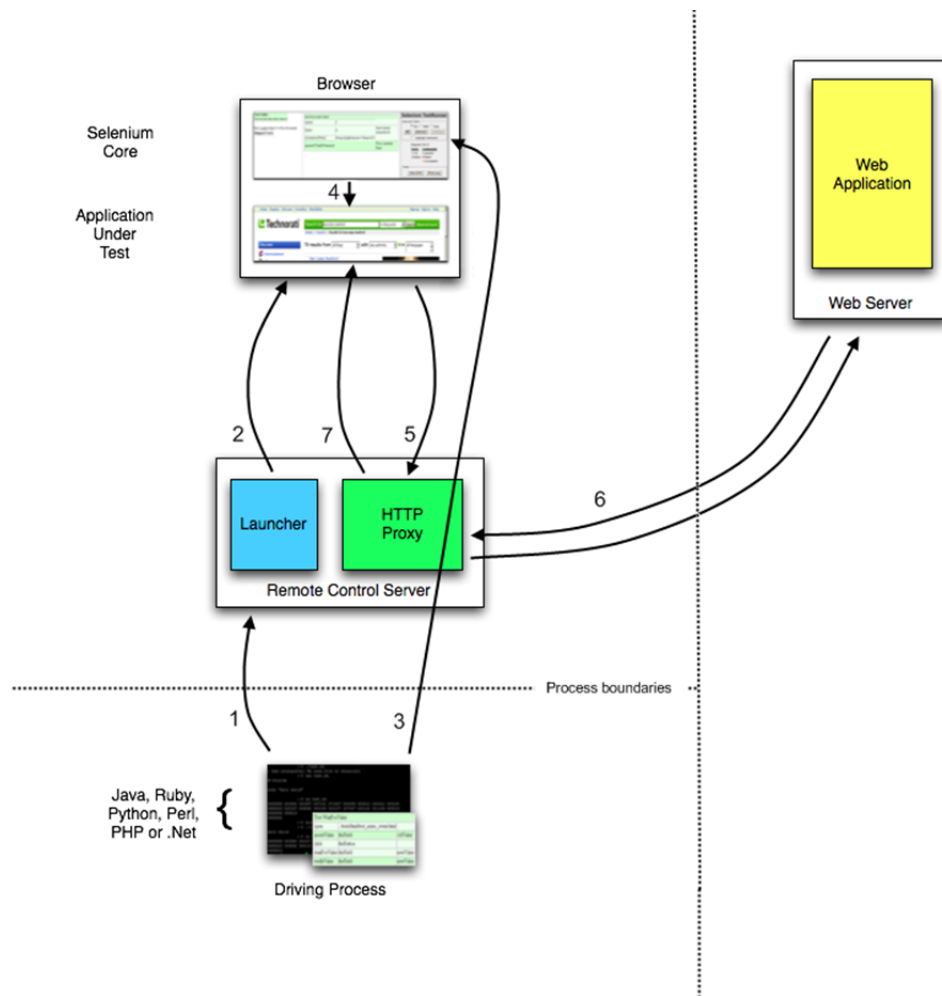


Figure 3.28 Selenium RC Architecture

As a test suite starts, the following happens:

1. The client/driver establishes a connection with the Selenium RC server.
2. Selenium RC server launches a browser with a URL that injects Selenium Core JavaScript library into the browser-loaded page.
3. The client driver passes a Selenium command (Selenese) command to the server.
4. The server interprets the command and then triggers the corresponding JavaScript execution to execute that command within the browser.
5. Selenium Core instructs the browser to act on that first instruction, typically opening a page of the AUT.

6. The browser receives the open request and asks for the website's content from the Selenium RC server (set as the HTTP proxy for the browser to use)
7. Selenium RC server communicates with the Web server asking for the page and once it receives it, it sends the page to the browser masking the origin to look like the page comes from the same server as Selenium Core.
8. The browser receives the web page and renders it in the frame/window reserved for it.

3.4.3 Building a Web Test with Selenium IDE

The Selenium IDE (Integrated Development Environment) is the tool used to develop Selenium test cases. It is an easy to use Firefox plug-in and is generally the most efficient way to develop test cases. It also contains a context menu that allows selecting a UI element from the browser's currently displayed page and then selecting from a list of Selenium commands with parameters pre-defined according to the context of the selected UI element.

Selenium IDE can be installed from Selenium Project download page (<http://seleniumhq.org/download/>) using FireFox browser. The download page is shown in Figure 3.29

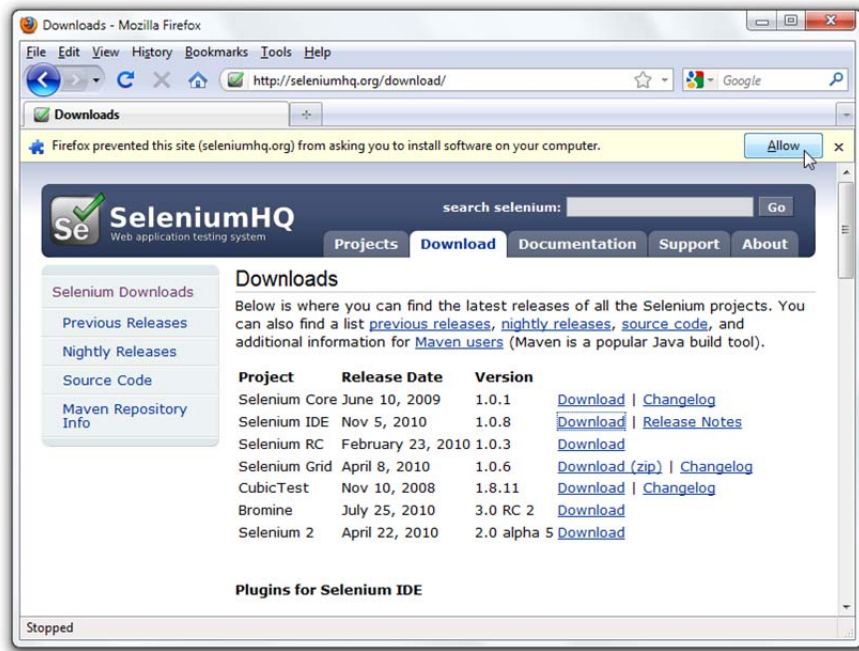


Figure 3.29 Selenium Project download page

After installing Selenium IDE, there will be a menu item under “Tools” to start the IDE. It loads with an empty script-editing window and a menu for loading or creating new test cases.

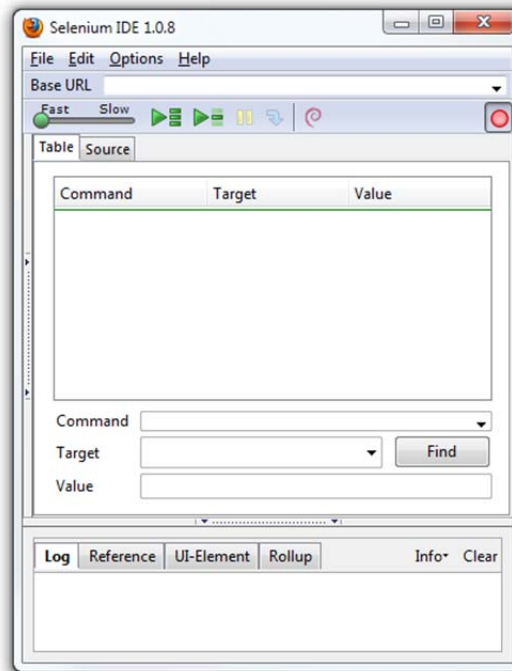


Figure 3.30 Snapshot of Selenium IDE

There are three primary methods for developing test cases. Frequently, a test developer will require all three techniques.

Many first time users begin by recording a test case from their interactions with a web application. When Selenium-IDE is first opened, the record button is on by default. During recording, Selenium IDE will insert commands into the test case based on the actions. Typically, this will include:

- Clicking on a link – “click” and “clickAndWait” commands.
- Entering values – “type” command.
- Selecting options from a drop-down list – “select” command.
- Clicking checkboxes or radio buttons – “click” command.

Test case also needs to check properties of a web page. This requires “*assert*” and “*verify*” commands. This can be done via Selenium context menu items. When a block of text is selected on a Web page, the context menu displays “*verifyTextPresent*” command and suggested parameter should be the text itself. Right-clicking an image, or a user control like a button or a checkbox, there will be extra menu options displayed on the context menu, such as “*verifyElementPresent*”.

Verifying UI elements on a web page is probably the most common feature of automated tests. Selenium commands, “Selenese”, allow multiple ways of checking for UI elements. Choosing between “assert” and “verify” comes down to convenience and management of failures. There is no point checking that the first paragraph on the page is the correct one if the test has already failed when checking that the browser is displaying the expected page. If it is not the correct page, aborting the test case is probably the best option, so that the cause can be investigated. On the other hand, many attributes of the web page might be checked without aborting the test case on the first failure, as this will allow reviewing all failures on the page and taking the appropriate action. An “*assert*” will fail the test and abort the current test case, whereas “*verify*” will fail the test and continue to run the test case.

3.4.4 Example of Selenium Web Driver API

WebDriver is a tool for automating web application testing, and to verify that they work as expected. It aims to provide a friendly API that's easy to explore and understand, which will help to make tests easier to read and maintain. It is not tied to a particular test framework, so it can be used equally well in a unit testing or any other type of testing.

For example, in Table 3.3 there is code to search for a term on Google and then outputs the result page's title to the console.

The first thing done is to navigate to a page by calling “*driver.Navigate().GoToUrl()*”. After navigation the test tries to find an element by name “*q*”. Then a search query “*Cheese*” is sent to this text box. After calling “*submit*” method on the text box, the test waits for 10 seconds or until the page loads for the results to load and checks for the title of the page starts with the keyword. And then it quits by writing an informational message to the console with the result page title.

Table 3.3 Sample Selenium WebDriver code

```
using OpenQA.Selenium;
using OpenQA.Selenium.Firefox;

// Requires reference to WebDriver.Support.dll
using OpenQA.Selenium.Support.UI;

class GoogleSuggest
{
    static void Main(string[] args)
    {
        // Create a new instance of the Firefox driver.

        // Notice that the remainder of the code relies on the interface,
        // not the implementation.

        // Further note that other drivers (InternetExplorerDriver,
        // ChromeDriver, etc.) will require further configuration
        // before this example will work. See the wiki pages for the
        // individual drivers at http://code.google.com/p/selenium/wiki
        // for further information.
        IWebDriver driver = new FirefoxDriver();

        //Notice navigation is slightly different than the Java version
        //This is because 'get' is a keyword in C#
```

Table 3.3 Continued

```
driver.Navigate().GoToUrl("http://www.google.com/");

// Find the text input element by its name
IWebElement query = driver.FindElement(By.Name("q"))
// Enter something to search for
query.SendKeys("Cheese");

// Now submit the form. WebDriver will find the form for us from the element
query.Submit();

// Google's search is rendered dynamically with JavaScript.
// Wait for the page to load, timeout after 10 seconds
WebDriverWait wait = new WebDriverWait(driver, TimeSpan.FromSeconds(10));
wait.Until((d) => { return d.Title.ToLower().StartsWith("cheese"); });

// Should see: "Cheese - Google Search"
System.Console.WriteLine("Page title is: " + driver.Title);

//Close the browser
driver.Quit();
}
```

3.4.5 Creating and Running Selenium Test

After reviewing Selenium components and tools, and looking at sample code, Selenium IDE can be used to create a test for the cloud application created in section 3.1.

As soon as Selenium IDE is started Firefox browser is started with recording mode. Navigating to the sample cloud application (<http://testingcloud.windowsazuresites.net>) the home page is loaded. Right-clicking on the title “Home Page” and selecting “*verifyTitle Home Page – My ASP.NET Web Page*” as shown in Figure 3.31 will create a verification step in Selenium IDE.

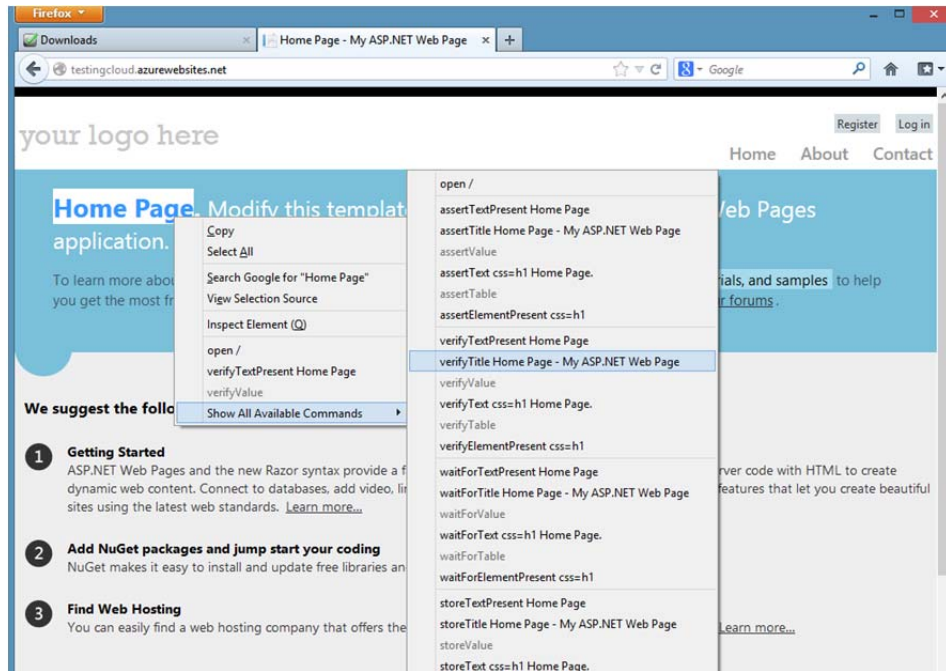


Figure 3.31 Available commands for Selenium context-menu

After that clicking on the “About” link will create two more steps in the test case, “*clickAndWait*” and “*open*” which will navigate the test to the specified page. The test can be concluded by right-clicking on the browser and selecting “*assertTitle About – My ASP.NET Web Page*” which will also add another command in the Selenium IDE as shown in Figure 3.32

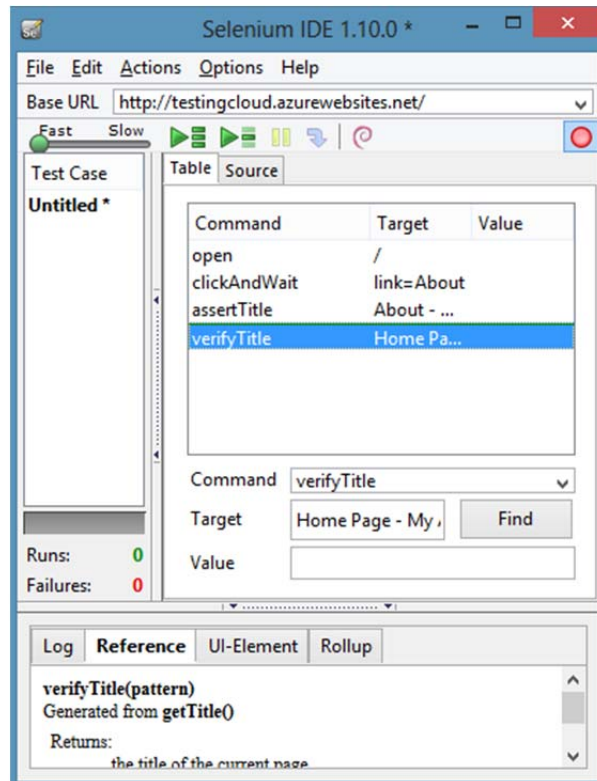


Figure 3.32 Selenium IDE after the recording stopped

Clicking on “Stop Recording” button will end the recording and the test can be saved. After saving the test, it can be re-run as many as time requested which will result in executing the steps and running through verification steps as shown in Figure 3.33

The results will be displayed in the “log” pane below the IDE window. The summary will be shown under “Test case” pane. If there are more than one test case in the test suite, the overall report can also be observed here.

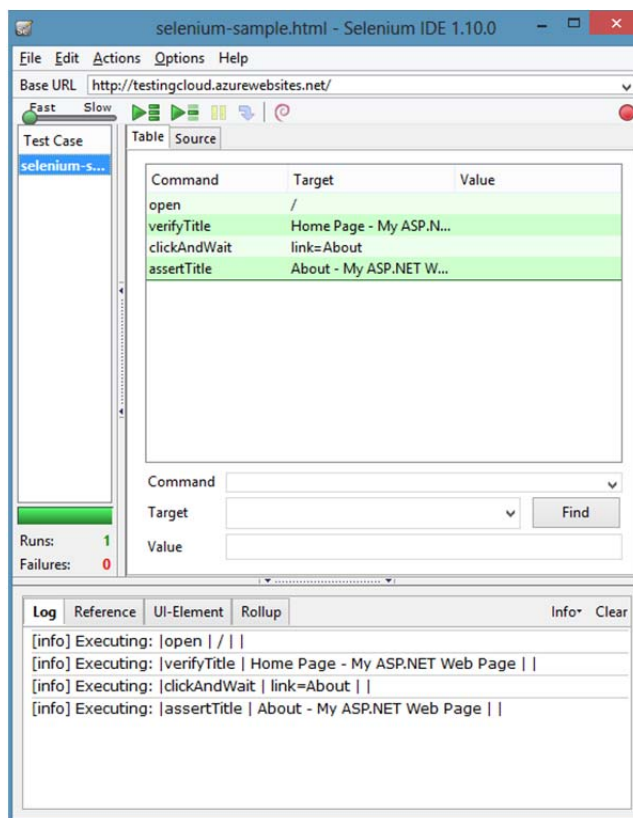


Figure 3.33 Running tests in Selenium IDE

It is possible to add more commands and extend the test case using Selenium IDE. The other option is also to export this test case as a WebDriver or RC test case with whatever language is preferred. This way the test case can be added to source control systems and re-run automated using WebDriver libraries. The sample test case is exported as C# WebDriver test which is shown in Table 3.4

Table 3.4 Selenium WebDriver code to test cloud application

```

using NUnit.Framework;
using OpenQA.Selenium;
using OpenQA.Selenium.Firefox;
using OpenQA.Selenium.Support.UI;
namespace SeleniumTests
{
    [TestFixture]
    public class SeleniumCsharp
    {
        private IWebDriver driver;
        private StringBuilder verificationErrors;
        private string baseURL;
        private bool acceptNextAlert = true;
        [SetUp]
        public void SetupTest()
        {
            driver = new FirefoxDriver();
            baseURL = "http://testingcloud.azurewebsites.net/";
        }
    }
}

```

Table 3.4 Continued

```

verificationErrors = new StringBuilder();
}
[TearDown]
public void TeardownTest()
{
    try
    {
        driver.Quit();
    }
    catch (Exception)
    {
        // Ignore errors if unable to close the browser
    }
    Assert.AreEqual("", verificationErrors.ToString());
}
[Test]
public void TheSeleniumCsharpTest()
{
    driver.Navigate().GoToUrl(baseURL + "/");
    try
    {
        Assert.AreEqual("Home Page - My ASP.NET Web Page", driver.Title);
    }
    catch (AssertionException e)
    {
        verificationErrors.Append(e.Message);
    }
    driver.FindElement(By.LinkText("About")).Click();
    Assert.AreEqual("About - My ASP.NET Web Page", driver.Title);
}
private bool IsElementPresent(By by)
{
    try
    {
        driver.FindElement(by);
        return true;
    }
    catch (NoSuchElementException)
    {
        return false;
    }
}
private string CloseAlertAndGetItsText() {
    try {
        IAlert alert = driver.SwitchTo().Alert();
        if (acceptNextAlert) {
            alert.Accept();
        } else {
            alert.Dismiss();
        }
        return alert.Text;
    } finally {
        acceptNextAlert = true;
    }
}
}
}

```

The code starts by running method “*SetupTest*” which loads a “*FirefoxDriver*”. It is also possible to run tests with other drivers, such as “*InternetExplorerDriver*”, “*ChromeDriver*” etc., which can be downloaded from Selenium Project’s web site. The “*baseURL*” is set to home page of the cloud application.

The main method “*TheSeleniumCsharpTest*” method is called next which navigates to the base URL. Then verification steps are run followed by finding elements on the web page and executing actions on them.

3.5 Summary for Selenium Suite

Selenium tools suite is an open-source project which is based on JavaScript and browser API. It presents many different ways of testing browser based applications including cloud applications.

However, it is not as integrated as Visual Studio. It requires combining different tools, drivers and scripts to be able to develop test cases. But this also presents a higher level of extensibility alongside strong customization support to tooling.

During the tests in this project, the lack of support for load/performance testing for Selenium test suite is observed. In order to load test the cloud application, other tools such as JMeter, Grinder or httpperf should be used.

Also for security testing, Selenium test suite can be used but not in full-fledged fashion. Security testing in Selenium requires embedding security testing concepts within Selenium tests and executing those tests. Selenium doesn't come with a ready to use security testing framework which can fill in the gaps and make this a more constructive, well-thought approach, rather than leaving it to the tester to come up with all scenarios.

Selenium is a very powerful framework for functional and UI based testing, however it lacks a couple of important features for testing cloud applications, such as load/performance, security etc.

CHAPTER FOUR

CONCLUSION

Software development especially in enterprise organizations is facing challenges of developing software more agile and with higher quality and less cost. Businesses are required to respond changing environment quicker than ever before, otherwise they risk losing money and customers if they don't.

As described in previous chapters, cloud computing can help companies deal with changes quickly and cost effectively.

However there are challenges designing and testing these applications which will be running in the cloud. Designing and developing applications which will be running on cloud has still some distance to go, the software lifecycle management and design patterns for developing cloud applications are still not an agreed upon topic. There are many methodologies and frameworks existing for managing such development activities. They are far from where the traditional software development methodologies are.

Having this scattered approach for designing and developing cloud applications also has adverse effects on agreeing upon testing framework and methodologies for testing cloud applications. Challenges involved in testing cloud applications can be addressed by resorting to a combination of advanced testing technologies and tools. Thus, it will result a level of automation that will enable testing in a continuous mode.

We expect the traditional software testing tools, especially targeting Web application testing can be adjusted to run against these cloud applications. Also new approaches, such as "Testing in Production" (TiP) will become more and more important, since the tests will be running as part of the deployed code, maybe in an isolated context, helping to reduce the cost of testing efforts for cloud platforms.

After all, everything that targets the cloud application results in bandwidth/cpu/storage charges.

However there are still challenges to be resolved. The cloud applications will have the ability to adapt to continually changing environmental conditions, but they will also have components which test themselves to ensure that software adaptations do not introduce faults into the system at runtime.

There will be tools, probably offered by the cloud platform vendors, integrated into their cloud management tools, which will present automated test generation of all types including unit, functional, stress, performance and load, and security test cases. And the existing test tools will have tight integrations with cloud platforms, so that the tests can be published to cloud to run in cloud.

New concepts such as “Software Testing as a Service”, “Testing in Production”, “Automatic Software Testing”, and “Improving testability of Cloud Applications” should be researched further within this topic.

REFERENCES

- Chappell, D. (December, 2009). *Introducing the Windows Azure Platform*. Retrieved December 2012, from http://www.davidchappell.com/writing/white_papers/Windows_Azure_Platform_v1.3--Chappell.pdf
- Eliot, S. (Jun 7, 2011). *Testing in Production (TiP) - It Really Happens—Examples from Facebook, Amazon, Google, and Microsoft*. Retrieved January, 2013 from <http://blogs.msdn.com/b/seliot/archive/2011/06/07/testing-in-production-tip-it-really-happens-and-that-s-a-good-thing.aspx>
- Eliot, S (November 15, 2011). *The Future of Software Testing Part One – Testing In Production*. Retrieved January, 2013 from <http://www.thetestingplanet.com/2011/11/THE-FUTURE-OF-SOFTWARE-TESTING-PART-ONE-TESTING-IN-PRODUCTION>
- Furht, B. & Escalante A. (Eds.). (2010). *Handbook of Cloud Computing*. New York, NY: Springer
- Kumar N.S. & Subashni S. (2010). *Software Testing using Visual Studio 2010: A Step-by-Step Guide to Understanding the Features and Concepts of Testing Applications Using Visual Studio*. Birmingham: Packt Publishing
- Naik S. & Tripathy P. (2008). *Software Testing and Quality Assurance: Theory and Practice*. Hoboken, NJ: John Wiley & Sons
- Page A., Johnston K., Rollison B. (2009). *How We Test Software at Microsoft*. Redmond, WA: Microsoft Press
- Sarna, D.E.Y. (2011). *Implementing and Developing Cloud Computing Applications*. Boca Raton, FL: Auerbach Publications

- Sitaram D. & Manjunath G. (2012). *Moving to the Cloud: Developing Apps in the New World of Cloud Computing*. Waltham, MA: Syngress
- Sosinsky, B. (2011). *Cloud Computing Bible*. Indianapolis, IN: John Wiley&Sons
- Subashni S. & Kumar N.S. (2008). *Software Testing with Visual Studio Team System 2008*: Birmingham: Packt Publishing
- Tiller S. & Parveen T. (Eds.) (2013). *Software Testing in the Cloud: Perspectives on an Emerging Discipline*. Hershey, PA: IGI Global
- Selenium Documentation*. (n.d.). Retrieved January, 2013, from <http://seleniumhq.org/docs/>
- Windows Azure .NET Developer Resources and Tutorial*, (n.d.). Retrieved January, 2013, from <http://www.microsoft.com/windowsazure/windowsazure>