Object Oriented Analysis of Centralized C# Compiler Using Cloud Computing with UML

Mr. Yogesh Bhanushali
Department of Computer Engineering
Rajiv Gandhi Institute of Technology, Mumbai, Maharashtra, India

Mr. Dwij Mistry
Department of Computer Engineering
Rajiv Gandhi Institute of Technology, Mumbai, Maharashtra, India

Ms. Shraddha Nakil
Department of Computer Engineering
Rajiv Gandhi Institute of Technology, Mumbai, Maharashtra, India

Ms. Sharmila Gaikwad
Assistant Professor, Department of Computer Engineering
Rajiv Gandhi Institute of Technology, Mumbai, Maharashtra, India

***Abstract*— Object-Oriented Analysis and Design(OOAD) is a popular technical approach to analyzing, designing an application, system, or business by applying the OO paradigm and visual modeling throughout the development lifecycle. The Unified Modeling Language (UML) is a popular international standard language used for object-oriented modeling. With the development of parallel computing, distributed computing, grid computing, a new computing model appeared. The basic principles of cloud computing is to make the computing be assigned in a great number of distributed computers, rather than local computer or remoter server. The CodeDom allows program to be dynamically created, compiled, and executed at runtime. To represent source code, CodeDOM elements are linked to each other to form a data structure known as a CodeDOM graph, which models the structure of source code. Using Object Oriented Analysis Technique we have done all the requirements elicitation for above topic and have analyzed using Rational RequisitePRO.**
Keywords— Centralized Compiler, Cloud Computing, OOA, Rational RequisitePRO, UML.

I. Introduction

[1]Compilers are used to run programs and convert them from a text format to executable format. A compiler that is to be installed manually on every system physically requires a lot of space and also configuring of it if not installed using default parameters. Also once a program is compiled it becomes platform dependent. It is also not easy to carry the same program code to multiple systems if situation doesn't permit the usage of a single system. Another drawback is that we would need to install a different complier on each language on which we wish to work. We propose a solution to this in the form of a cloud based compiler. Cloud computing is a model for enabling convenient, on demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort. Our project aims to create an online compiler which helps to reduce the problems of portability of storage and space by making use of the concept of cloud computing. The main reason for

implementing this paper is to provide a centralized compiling scheme. Also, it will act as a centralized repository for all the codes written. A technology that has fastest growing segments in IT and shown its high growth rate in the last few years, is Cloud Computing.

II. Studying cloud services

A. Software-as-a-Service

This was the earliest cloud service and the first to enjoy widespread adoption. In a nutshell, SaaS is the online delivery of software functionality and capability without the need for locally running software. Rather, SaaS runs on a Web browser. Gmail and Salesforce are two popular SaaS products. Enterprise level SaaS providers deliver a wide variety of sophisticated applications such as product lifecycle management, supply chain management, and many other vertical applications. Direct benefits of SaaS include reduced hardware costs, reduced software licensing costs, and more flexible IT resources that can be dialed up or down quickly on demand. Secondary benefits include reduced or outsourced software support overhead and simpler licensing and product lifecycle management requirements. Equally important, SaaS applications allow users to access and manipulate their data anywhere they have a data connection from any device.

B. Platform-as-a-Service

Broadly speaking a Platform-as-a-Service (PaaS) is a cloud-based application development environment. Using a PaaS, companies can produce new applications more quickly and with a greater degree of flexibility than with older development platforms tied directly to hardware resources. Running application development on a PaaS has a number of key benefits. Programmers and development managers especially appreciate that the cloud provider handles all the care and maintenance of the underlying operating system(s), servers, storage, and application containers. PaaS environments can be extremely useful when development teams are widespread geographically or when partner companies or divisions share development efforts

C. Infrastructure-as-a-Service

The most foundational use of cloud computing is Infrastructure-as-a-Service (IaaS). This entails the rental of complete computing resources for running applications, hosting data, or housing a company’s entire computing environment. IaaS is the delivery model which provides computer Infrastructure as a service. It is a single tenant cloud computing layer where the dedicated resources of Cloud Computing vendors are shared only with the contract based clients at a pay-per-use fee. This greatly reduces the need for huge initial investments in computing hardware such as networking devices, computing servers, and processing unit. It is the base layer of the Cloud computing Virtualized stack. Amazon EC2 is an example of an IaaS.

III. Analysis

IBM Rational RequisitePro is a requirements and use case management tool for project teams. Teams and author can share their requirements using familiar document-based methods, while using database capabilities such as traceability and impact analysis. This can improve communication and requirements management, increase quality and speed time to market. Rational RequisitePro is an easy-to-use tool that helps you : avoids rework and duplication, manage complexity with detailed traceability, improve collaboration, capture and analyze requirements, increase productivity, align business goals and objectives with project deliverables. The private cloud use case is different from the public cloud. In this use case, the user performs functions like Login, Registration, Compile code, Execute code, Download file. To run the code user must have a login id and password. If not he/she has to register first. The user can download the executed code as .exe file which user can run on his own machine. Refer Fig 1.

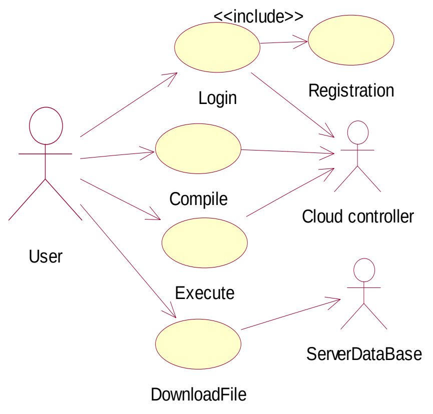
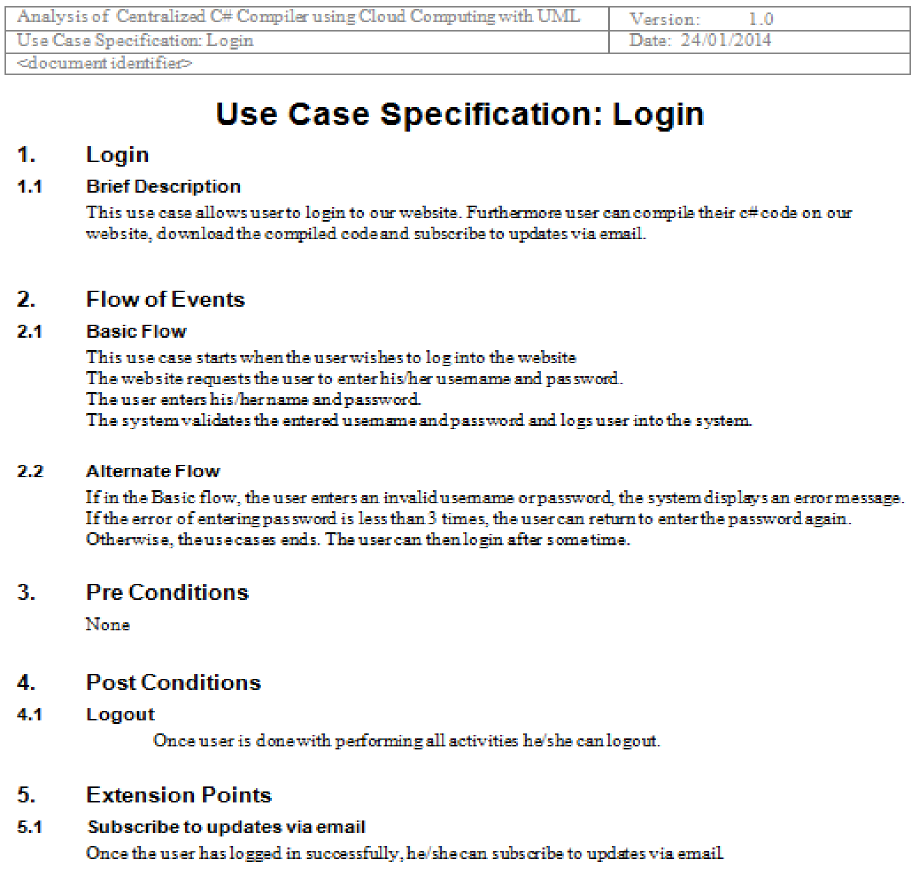


Fig 1: Use case diagram

The figure below shows the use case specification of Login using Rational RequisitePRO.



Analysis of Centralized C# Compiler using Cloud Computing with UML	Version: 1.0
Use Case Specification: Login	Date: 24/01/2014
<document identifier>	

Use Case Specification: Login

1. Login

1.1 Brief Description

This use case allows user to login to our website. Furthermore user can compile their c# code on our website, download the compiled code and subscribe to updates via email.

2. Flow of Events

2.1 Basic Flow

This use case starts when the user wishes to log into the website
The website requests the user to enter his/her username and password.
The user enters his/her name and password.
The system validates the entered username and password and logs user into the system.

2.2 Alternate Flow

If in the Basic flow, the user enters an invalid username or password, the system displays an error message.
If the error of entering password is less than 3 times, the user can return to enter the password again.
Otherwise, the use cases ends. The user can then login after some time.

3. Pre Conditions

None

4. Post Conditions

4.1 Logout

Once user is done with performing all activities he/she can logout.

5. Extension Points

5.1 Subscribe to updates via email

Once the user has logged in successfully, he/she can subscribe to updates via email.

Fig 2: Use case specification using RequisitePRO

The following is the sequence diagram which shows the sequence of actions performed by the users. The user first logs in through user interface on website. After successful login the user can paste the code. The user can then click on compile button. The application logic will compile the code and if there are any errors it will send it to web browser. Refer Fig 3 and Fig 4.

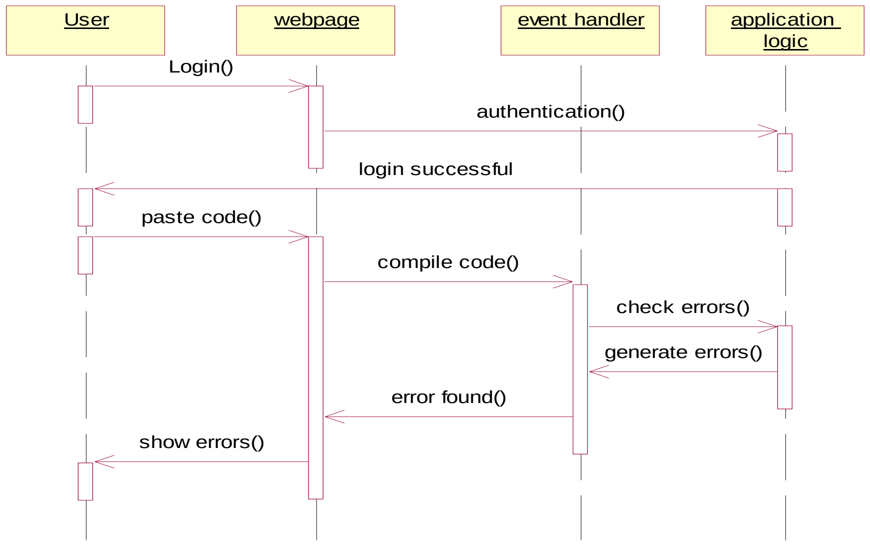


Fig 3: Sequence diagram for Compiled Errors

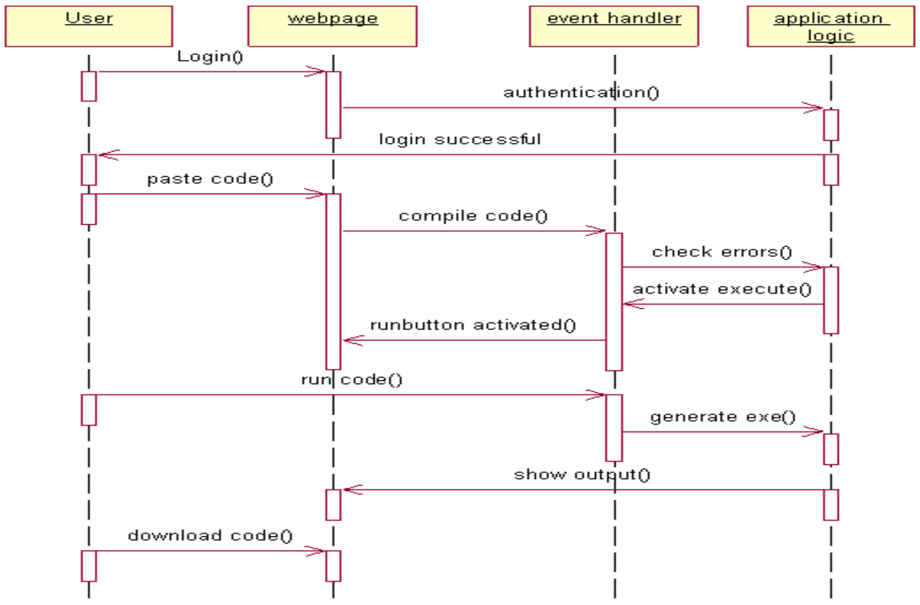


Fig 4: Sequence diagram for Compiled Output

A. Analysis of System using Activity Diagram

Activity diagram is important diagram in UML to describe dynamic aspects of the system. Activity diagram is basically a flow chart to represent the flow form one activity to another activity. The activity can be described as an operation of the system. So the control flow is drawn from one operation to another. Refer Fig 5.

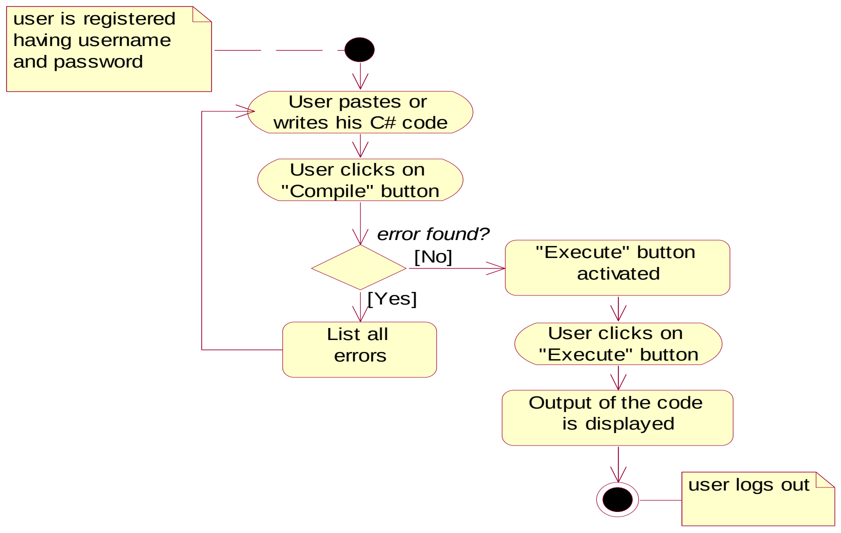

user is registered having username and password
User pastes or writes his C# code
User clicks on "Compile" button
error found?
[No]
[Yes]
"Execute" button activated
List all errors
User clicks on "Execute" button
Output of the code is displayed
user logs out

Fig 5: Activity diagram for the System

IV. IMPLEMENTATION

The CodeDOM provides types that represent many common types of source code elements. You can design a program that builds a source code model using CodeDOM elements to assemble an object graph. This object graph can be rendered as source code using a CodeDOM code generator for a supported programming language. The CodeDOM can also be used to compile source code into a binary assembly. Some common uses for the CodeDOM include:

Templated code generation: generating code for ASP.NET, XML Web services client proxies, code wizards, designers, or other code-emitting mechanisms.

Dynamic compilation: supporting code compilation in single or multiple languages.

System.CodeDomNamespace

The System.CodeDom namespace contains classes that can be used to represent the elements and structure of a source code document. The classes in this namespace can be used to model the structure of a source code document that can be output as source code in a supported language using the functionality provided by the System.CodeDom.Compiler namespace.

System.CodeDom.CompilerNamespace

The System.CodeDom.Compiler namespace contains types for managing the generation and compilation of source code in supported programming languages. Code generators can each produce source code in a particular programming language based on the structure of Code Document Object Model (CodeDOM) source code models consisting of elements provided by the System.CodeDom namespace.

V. *CONCLUSION*

The paper aims at creating & compiling C# code in the cloud. As compared to the current scenario where each machine need to install compilers separately. This would eliminate the need to install compilers separately. So we can check our code at the centralized server. Another advantage of such project is that whenever the compiler package is to be upgraded it can be done easily without again installing it on each and every machine.

VI. REFERENCES

[1] Yogesh Bhanushali, Dwij Mistry, Shraddha Nakil, Sharmila Gaikwad, “Centralized C# Compiler Using Cloud Computing”, Advanced Computing & Communication Technologies(ICACCT-2013). 7th International Conference.

[2] H. Singh and D.Seehan, “Current trends in Cloud Computing: a Survey of Cloud Computing Systems”, BSIOTR (W).

[3] Shuai Zhang Shufen Zhang Xuebin Chen Xiuzhen Huo, “Cloud Computing Research and development Trend”, Future Networks, 2010. ICFN '10. Second International Conference.

[4] Cloud Documentation and Centralized Compiler for Java & Php Namrata Raut Darshana Parab Shephali Sontakke, Sukanya Hanagandi Department of Computer Engineering, JSPMs BSIOTR(W)