CHAPTER 4: CLOUD COMPUTING REPOSITORY
MODEL: CCR MODEL

The present era of cloud computing technology needs the usage of professional models of software development. These models tend to be very costly as the software is developed from the scratch. We know that the testing & maintenance comprises of 40% cost in the project. The concept of reusability occurs which implies reusing the existing software/ component by modifying it as per needs & finally obtain the required software hence reducing cost in every respect. The authors have proposed a model by using cloud component repository where the component reusability occurs from the help of cloud component repository. This model overcomes the drawback of traditional software development where the last cost was involved.

4.1 Overview

Reusability is the mean of taking the existing system & as per the requirements of the new system the existing software is being modified by the involvement of experts in the manner that an existing system needs to provide the functionalities of new system & also work in itself. (Basili V, Rombach et al (1991)). The process of software engineering focuses only on the software development & is now able to recognition so as to achieve the software more quickly with low price budget hence to adopt the design need based on systematic software reuse is very important (Mathur P (2010)). Reverse engineering implies the evaluation of something in order to understand its working so as to duplicate or enhance it. (Hyatt L, Rosenberg L (1996)) ((Olsem, Michael R et al (1995))). The object oriented software developers admit the fact that the software based on this principle is easy to develop but not as easy as it tends to be due to the presence of various complexities. The programs are normally very complex if they are not documented properly hence they are difficult to comprehend & even a simpler upgrade may become very typical (F Chong, G Carraro (2006)) (Asit Kumar Gahalaut (2010)). So the eminent development focuses on advanced documentation of the program as the
reengineering code largely affect the issues related to the program comprehension during the increase in software size. (J Handler, N Shadbolt et al (2008)) (Basili V, Rombach D (1991)).

4.2 Related Works

Re-engineering is the process of examining, analyzing & altering the existing software system so that it can be reconstituted in a new form by doing the alteration & modification in the existing system & finally implementing it into a new form. This process is comprised of combination of various processes including the reverse engineering, documenting again, structuring again, re-translating as well as the engineering forwardly. The re-engineering objective is maintaining the existing functionality & and focusing on the functionality to be added without disturbing the existing functionalities. The re-engineering challenges are taking the existing systems and thereby good software development methods and properties are applied hereon for the generation of a new target system thereby the maintainance of keeping the new functionalities by the use of latest technologies. Re-engineering objectives are discussed below:

- Functional enhancement preparation
- Maintainability improvement
- Migration
- Reliability improvement

The technique of re-engineering required that the various features or principles that need to be followed including the portability, reliability, maintainability, testability & usability (Sneed, Harry M et al (1990)). It allows the program and also helps in construction of a high-level representation to be used in various activities including documenting, maintaining, reusing & the reverse engineering technique begins by analyzing a program structure to obtain the goal by determination of syntax, lexical and semantic rules for the construction of the legal programs which is thereon applied in the re-engineering process by software analysis for applying changes & thereby new features are incorporated with
to provide support for latest environment. (Wilson W, Rosenberg L et al (1996)). The Figure 4.1 shows the typical reverse engineering process which is started by the requirements extraction and gathering the detailed information of design from the source code and the documents which are at present. The recovered design is reviewed for consistency and correctness.

![Diagram of reverse engineering process]

**4.3 Analysis**

Software transforms the personal data including the financial transactions in order to make the data more usable in the local context & also manages the business information so as to enhance the competition. Software Development Life Cycle (SDLC) is a process of building or maintaining software systems (Wilson W, Rosenberg L et al (1996)). The software development is carried out by means of Software Development Life Cycle which consists of various stages such as requirements gathering, planning, designing, coding, testing & integration (Wilson W, Rosenberg L et al (1996)). The whole
development process from conceptualization till the operation is called as the SDLC process in software engineering. There are various SDLC models such as Waterfall, Incremental, Spiral, RAD, XP model & Internet Speed Development model. The traditional software development cycle starts from the requirements phase where the Software Requirement Specification (SRS) is prepared that covers the functional & non-functional requirements. This SRS is the input for the Design phase where the architectural staff designs the corresponding pages as per requirements. The output of design phase i.e.: UML diagrams, Use Case diagrams are the input for the coding phase. As soon as the developer recognizes the requirements of software to be developed they start their coding part in the required platform & finally the testers validate the developed software by means of test case designing. This process is a very costly means as the software to be developed needs that all phases should be followed. We know that all these phases are very costly. Even a small modification needs too much cost. If we talk about the cloud computing so it is not possible that we can develop the software right from the beginning every time as it needs the software developer as well the personnel from the cloud provider in order to be operational at the cloud platform. Now every organization is moving into the era of cloud computing with the sole objective of reducing hardware/infrastructure as well as other costs. But to attain all these things we need to develop or modify the existing software in order to work them at the cloud platform & this is only possible by means of component reusability or reengineering the existing system. As the component reusability reduces at least the requirements cost as we need to find the appropriate component to make it reusable for cloud software & this is a very good strategy to reduce cost in any manner (Mathur P (2010)). During the development of the software components verification & validation becomes a bit more complicated task.

4.4 Proposed Model

In order to remove the drawback of traditional development life cycle software development model of more cost we have proposed a cloud computing model through the component reusability. The component to be reused will be taken up from the cloud component repository. The Cloud Component Reusability Model is shown in Figure 4.2.
Here we have an inappropriate source code that needs to be modified so we apply the necessary reengineering activities such as forward & backward engineering after which we gets the clean code as well as clean documentation which are then analyzed further in order to obtain reusable component identification process where we come to know how the component to be reused can be identified. Also we tend to obtain the reusable component of cloud computing & finally when we identify & reuse the component then the component testing along with the V&V activities are performed. Finally the quality of cloud component is taken care of & if it is good enough then the black box reuse of component integration is performed & if quality is not as per standards then the white box reuse is applied & finally at last the cloud based development process takes place. In this model we have the inappropriate source code available with us which is not relevant to us hence it needs to be made relevant so that we can use it. Then we have the option of re-thinking, re-specifying and re-designing, restructuring of the data, re-coding, clean documentation and the clean code. Re-think specifies that one should think for the changes to be made again so that the software got usable and fulfils all the necessary requirements. Re-specifying implies that the procedure of software development needs to be re-specified and the requirements been specified need to be specified again so that effectiveness takes place. Since the requirements are re-specified so correspondingly the coding needs to be changed a lot as per the changes in the requirements as the earlier coding will fail to provide the complete functionality. The coding is always dependent on the requirements defined in the software requirement specification. As the documentation available with us also need to be changed as the earlier design comprises of different activity diagram, design patterns, data flow diagram as well as the test cases also need to be changed as all these design structures need to be updated timely and correspondingly the requirements get changed and hence the coding also get changed. Since we have the data earlier is of less importance as the figures got changed due to change in the design and requirements phase and in fact the whole sdlc phase needs to be modified hence there is a need of documenting again very specifically and carefully as the wrong SDLC phase following may cause very big danger. The clean code activity involves cleaning the code by the means of removing the excess code which is not relevant to the requirements. Clean code is a very important criteria as this is the base for the other code
To get a proper understanding. This require very clear specification of the code so that
other can understand the code very easily. As soon as this re processes are done so then
the analysis is being done for the software and the requirements need to be properly
understand and correspondingly the design needs to be prepared along with the necessary
design patterns such as the activity diagram, er diagram, data flow diagram of level 1,
level 2, level 3 till level n and after that correspondingly the coding needs to be done in
the LOC by the software engineer and later the coding needs to be tested for the various
test cases for each and every activity and the functionality of the software as it is very
crucial. Now the role of cloud component repository which consists of various codes of
the softwares that are available free of cost or open source software. This repository is
needed as we need to use the existing code for the re-engineering and getting the required
one quickly. Firstly the components needs to be classified according to the re-usability
principle and then correspondingly the components which have the permission to re-use
are then taken for the re-usability as this reduces the cost at a glance for the development
from scratch. So in this manner the re-usable component srae fully identified and hence
we get our ultimate components which later needs modification. As soon as the desirable
component get selected so the next step is to test that component by various test cases so
as to confirm ts actual usage for the modification. The testing seems very crucial phase
as it decides th quality of the software and then the later maintenance. As long as the
components are tested properly they need to be validated for the various test cases
designed to get the exact requirements and all the data values are used for proper
identification and validation. Validation implies checking the test cases. After all the
necessary conditions are applied so we get the searching of the cloud component. As soon
as it gets selected we check the quality of the selected component and if it is good in
quality so it gets adapted with the black box re-use else the component is adapted with
the white box re-use capability. As the component got selected for the black box re-use so
the component composition is required very quickly. Later the cloud based development
process is followed for the development of relative software from the components being
selected and this process gets followed by again selecting the components from the
repository.
Figure 4.2. Cloud Component Reusability Model
4.5 Conclusion

A model for cloud computing reusability called as cloud component reusability model has been proposed. The main objective behind this is to reduce the cost for developing the cloud compatible software. The study shows that reusability of software component can reduce the cost at a very large extent hence it should be applied to make the software compatible for cloud computing. The practical implementation will be shown in further proceedings. The main objective is the incorporation of this new dimension of interaction with the reusability with the leading software process model. The traditional software development only offers product specific solutions to the problems while the reuse & component based development technology helps in providing a generalized solution. A component is designed such that it can be modified any time according to the needs of the client/cloud provider. Later the component is tested for the quality and if the quality is worth satisfactory so it is adapted for the black box re-use else the component if not satisfactory in the quality so the component is adapted but for the white box re-use and then the component gets integrated with the other components so as to make the software functional for use. As soon as the integration of the relevant components are done so there is the time for the cloud based development process to develop the software from the necessary components according to the needs and finally the cycle again goes on.