International Journal of u- and e- Service, Science and Technology
Vol.6, No.5 (2013), pp.191-202
http://dx.doi.org/10.14257/ijunesst.2013.6.5.17

Introducing Agility in Cloud Based Software Development through ASD

Sukhpal Singh[1] and Inderveer Chana[2]

[1, 2]*Thapar University, Patiala*
ssgill@thapar.edu, inderveer@thapar.edu

Abstract

Cloud based development is a challenging task for several software engineering projects, especially for those which need change along with reusability. The present scenario of cloud computing is allowing new professional models to use agile software development. The expected upcoming trend of computing is assumed to be cloud computing as it ensures a lot of payback like no principal outflow, speed of application deployment, shorter time to market, lower cost of operation, and change at any level, reusability and easier maintenance for the tenants. Until Adaptive Cloud Development Model is considered a fundamental capability, predictable demand cannot be delivered to cloud users. This paper extends the traditional agile process model named Adaptive Software Development (ASD) and integrates interaction with the cloud provider to facilitate acceptance of cloud computing. In this paper, Adaptive Cloud Development Model has been proposed. Using the agile based cloud computing proposed approach, development cost can be minimized and customer satisfaction and reusability can be improved.

Keywords: *Adaptive, Agile software development, Agility, Cloud based software development, Cloud computing, Reusability, Software engineering, Software reuse*

1. Introduction

Agile software development is a group of software development methods based on iterative and incremental development, where requirements and solutions evolve through collaboration between self-organizing, cross-functional teams [1]. It promotes adaptive planning, evolutionary development and delivery, a time-boxed iterative approach, and encourages rapid and flexible response to change [4]. It is a conceptual framework that promotes foreseen interactions throughout the development cycle. Adaptive Software Development is a software development process that grew out of rapid application development work by Jim High smith and Sam Bayer [7]. ASD embodies the principle that continuous adaptation of the process to the work at hand is the normal state of affairs [2]. ASD replaces the traditional waterfall cycle with a repeating series of speculating, collaborate, and learn cycles. This dynamic cycle provides for continuous learning and adaptation to the emerging state of the project. The characteristics of an ASD life cycle are mission focused, feature based, iterative, time boxed, risk driven, and change tolerance [5]. In Section 2 the related work has been described. The challenges of cloud computing platform for software are analyzed in Section 3. The Section 4 the Adaptive Cloud Development Model has been discussed. The experimental results explained in Section 5. The Section 6 concludes the whole work and provides future work.

ISSN: 2005-4246 IJUNESST

2. Related Work

The word speculates refers to the paradox of planning – it is more likely to assume that all stakeholders are comparably wrong for certain aspects of the project's mission, while trying to define it [8, 9]. Collaboration refers to the efforts for balancing the work based on predictable parts of the environment (planning and guiding them) and adapting to the uncertain surrounding mix of changes caused by various factors – technology, requirements, stakeholders, software vendors, *etc.*, [3, 19]. The learning cycles, challenging all stakeholders, are based on the short iterations with design, build and testing. During these iterations the knowledge is gathered by making small mistakes based on false assumptions and correcting those mistakes, thus leading to greater experience and eventually mastery in the problem domain [6, 10]. Reusable architectures can be developed from reusable architectural patterns [11] as in FIM architecture [12] which operates at three different levels of reuse: Federation, domain and application. Paulisch F. and Siemens AG focuses on how non-functional property reusability relates to the software architecture of a system. K.S. J. and Dr. Vasantha R. presented a suggested software process model for reuse based software development approach [17, 18]. A common ground for agile software development was defined in 2001, when 17 experienced and recognized software development "gurus", inventors and practitioners of different agile software development methods gathered together. Participants agreed and signed The Manifesto for Agile Software Development [3].

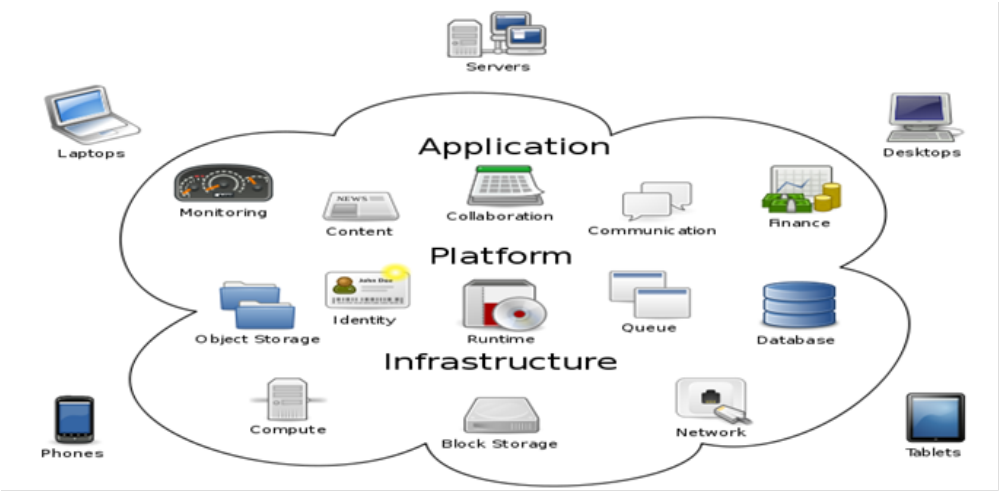

Figure 1. Cloud Computing and its Services [13]

Cloud computing is the use of computing resources (hardware and software) that are delivered as a service over a network (typically the Internet). The name comes from the use of a cloud-shaped symbol as an abstraction for the complex infrastructure it contains in system diagrams as shown in Figure 1. Cloud computing entrusts remote services with a user's data, software and computation [13]. We had conducted a survey on the number of approaches existing for Agile Software Development [14, 9], Cloud Based Development and Reusability [11] individually, but the proposed model combines both Agile Software Development and Cloud computing along with Reusability into a single approach for achieving efficient classification, storage and retrieval of software components and improve cloud customer satisfaction . Presently there is no such approach as presented in proposed model which combines Agile Software and Component based Development [15].

3. Analysis

In the rapidly changing computing environment with cloud platform, software development is going to be very challenging. The software development process will involve heterogeneous platforms, distributed web services, multiple enterprises geographically dispersed all over the world.

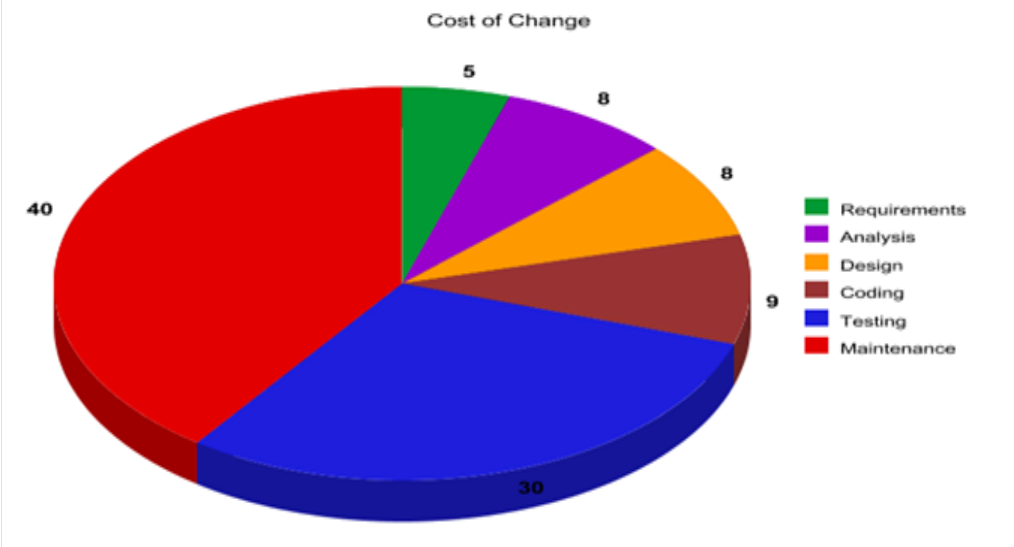

Figure 2. Evolution of Software Engineering

Existing software process models and framework activities are not going to be adequate unless interaction with cloud providers is included [16]. Requirement changes of a SW are the major cause of increased complexity, schedule and budget slippage.

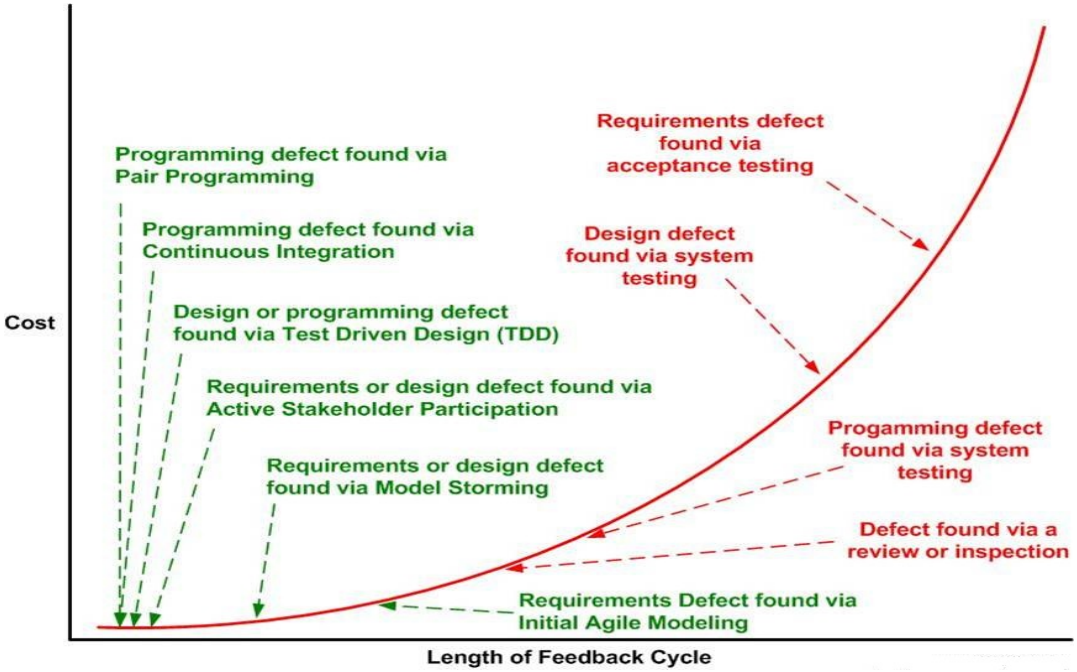

Figure 3. Variation in Cost [21]

Table 1. 12 Principles of Agile Software Development [1]

Agile Software Development Principles	
1. Satisfy the customer	7. Measure by working software
2. Welcome changing requirements	8. Maintain constant pace
3. Deliver working software frequently	9. Sustain technical excellence and good design
4. Motivate individuals	10. Keep it simple
5. Interact frequently with stakeholders	11. Empower self-organizing teams
6. Communicate face to face	12. Reflect and adjust continuously

Incorporating changes at a later stage of SDLC increases cost of the project has been described in Figure 2. Adding number of programmers at a later stage does not solve the schedule problem as an increased coordination requirement slows down the project further [4, 1]. The cost of change according to feedback will be increased from requirement to maintenance phase shown in Figure 3 (Scoot W. Ambler, 2006). It is very important that requirements gathering, planning and design of the SW is done involving all the parties from the beginning. That's why several agile process models like Adaptive Software Development (ASD), Extreme Programming (XP), Scrum, Crystal and Adaptive etc. have been introduced in mid 1990s to accommodate continuous changes in requirements during the development of the software. These agile process models have shorter development cycles where small pieces of work are "time boxed", developed and released for customer feedback, verification and validation iteratively [2, 5]. One time-box takes a few weeks to maximize a month of time [17]. The agile process model is communication intensive as customer satisfaction is given the utmost importance [10].

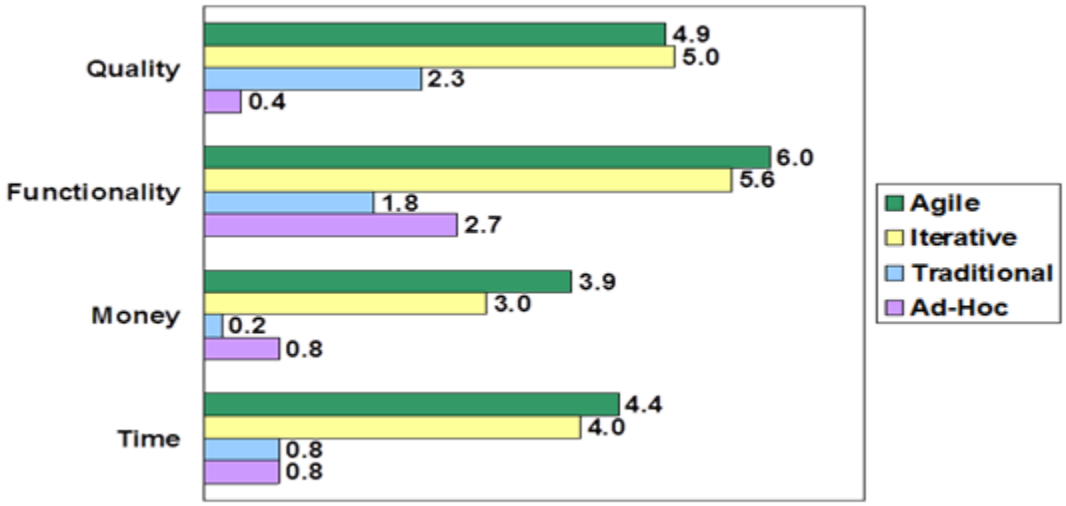

Figure 4. Emergence of Agile Software Development

Manifesto for Agile Software Development is followed by 12 principles has been described in Table 1. In this paper we assume, that these principles are important to consider for software development process to be recognized as agile [20]. We do not question their validity or sufficiency and accept them as it is. We use these principles as a base for identifying possible bottlenecks in different agile software development methods [1]. Dr. Dobb's Journal (DDJ) 2008 Project Success Survey described that agile teams have an average success rate of 70% compared with 66% of traditional/waterfall teams has been

summarized in Figure 4. Agile teams produce higher quality work, are quicker to deliver, are more likely to deliver the right functionality, and more likely to provide greater ROI than traditional teams. There are different features in software project but all of them are never used has been summarized in Figure 5.

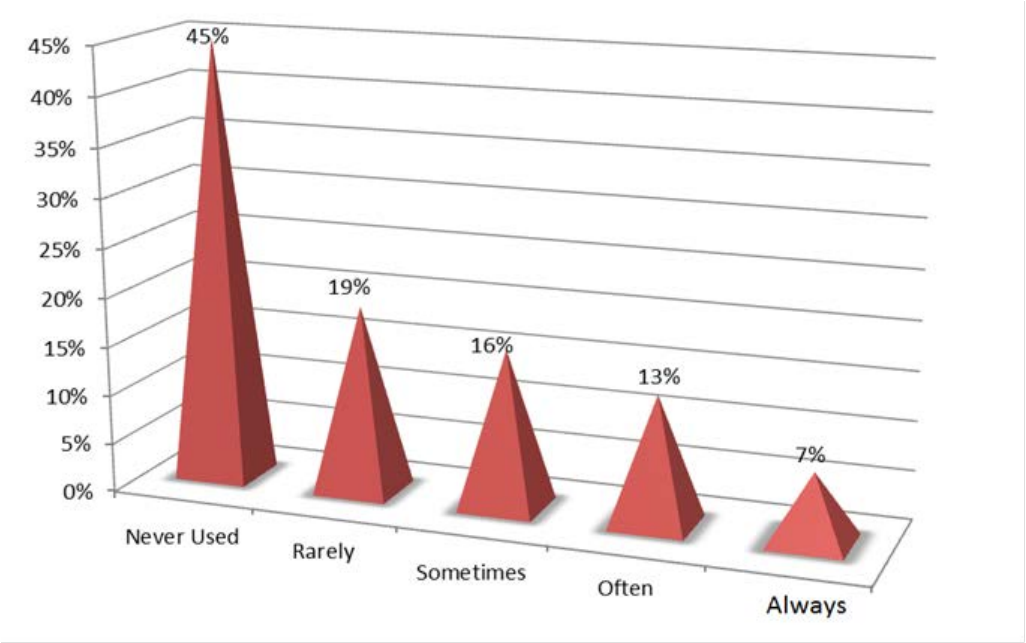

Figure 5. Usefulness of Development of Software

The project success rate is increasing now, according to a survey conducted by Standish Group the success rate was 16% in 1994, 28 % in 2001 and it has been reached to 31 % in the year 2003. The result of conducting the survey has been described in Figure 6 according to different CHAOS reports.

Figure 6. Success Rate of Software Project

Based on this analysis the there is need of agile software development for cloud computing, it will also incorporate reusability in the development as a component based development.

4. Adaptive Cloud Development (ACD) Model

Innovative software engineering is required to leverage all the benefits of cloud computing and mitigate its challenges strategically to push forward its advances. Here we propose an extended version of Adaptive Software Development (ASD), an agile process model for cloud computing platform and name it Adaptive Cloud Development Model [Figure 7]. A model capable of developing cloud based applications with agile software development and incorporating reusability by retrieving the components from the agile repository by using pattern matching algorithms and various retrieval methods. There are some different 6 retrieval methods are available for the classification of components in the software library. This model will help to make searching faster based on classification of components and develop the cloud based application according to cloud customer requirements to improve customer satisfaction as compared to traditional software development approach. The flow chart of proposed Adaptive Cloud Development Model is shown in Figure 8.

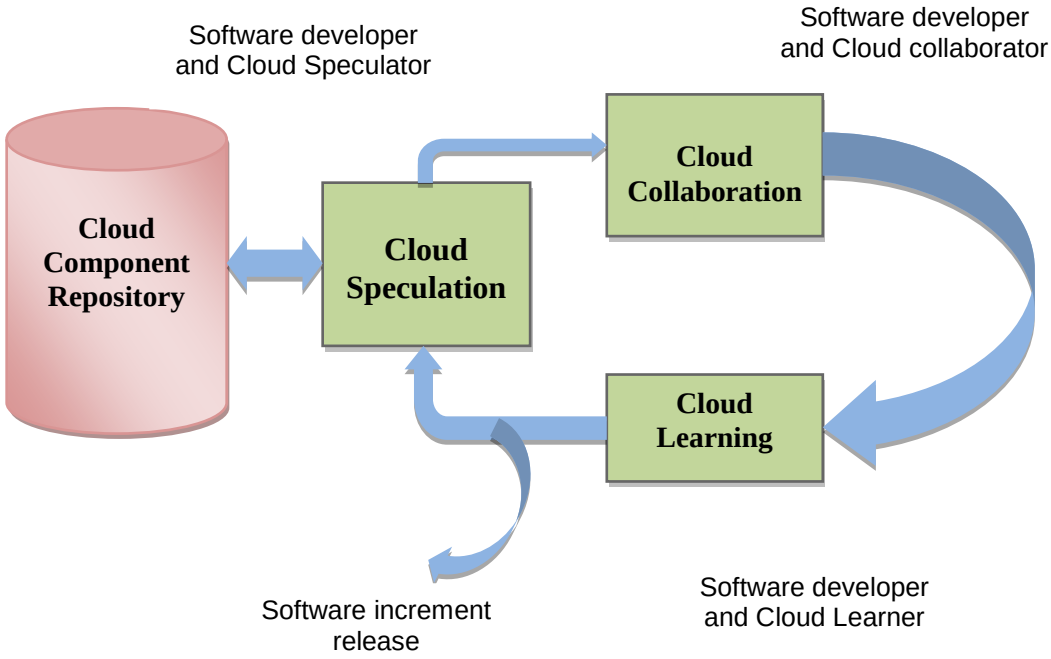

Figure 7. Adaptive Cloud Development (ACD) Model

4.1. Cloud Speculation

During cloud speculation, the cloud application is stated and adaptive cycle development of cloud project is conducted. It uses project instigation- the cloud customer mission statement, cloud application constrictions (e.g., delivery dates or cloud user details) and simple requirements to describe the set of release cycles (software increments) that will be essential for cloud development. Cloud speculator started to develop a project plan basis on cloud user demands.

4.2. Cloud Collaboration

Motivated software developers and cloud collaborator work together in the way that multiplies their talent and creative output beyond their absolute numbers. This cloud collaborative approach is a periodic subject in all agile models. It is not simply

communication among software developers and cloud collaborator, effective communication and coordination is also a part of the ACD. It requires a solidified team for actual collaboration to happen. It is not a rejection of individualism, because individual creativity plays an important role in collaborative thinking. Cloud collaborator and software developer working together must trust on one another to 1) disapprove without acrimony; 2) contribution without offense; 3) work as harder as they do; 4) have the skill set to contribute to the work at hand; and 5) communicate difficulties or anxieties in a way that leads to effective action.

4.3. Cloud Learning

As participants of Adaptive Cloud Development (ACD) team, cloud learner start to develop the components that are part of an adaptive cycle, the emphasis is on learning as much as it is in progress toward a completed cycle. Cloud learning will help cloud learner as well as software developers to improve their level of real understanding. The ACD team learns in three ways:

4.3.1. Learner Group: The cloud learner, cloud user and cloud customer provide feedback on software increments that are being delivered. This provides the indication of level of cloud customer satisfaction.

4.3.2. Learner Reviews: ACD team members review the software components that are developed, improving quality and learning as they proceed.

4.3.3. Cloud postmortem: The ACD team becomes introspective, addressing its own performance and process.

4.4. Cloud Component Repository

The approved developed components to the agile cloud development model will be stored in cloud component repository and retrieved it at a later stage for other cloud application development. There is some different storage and retrieval methods [22] (Information retrieval methods, Operational semantics methods, Descriptive methods, Denotational semantics methods, Topological methods and Structural methods) are available for the classification of components in the software library. This model will help to make searching faster based on classification of components.

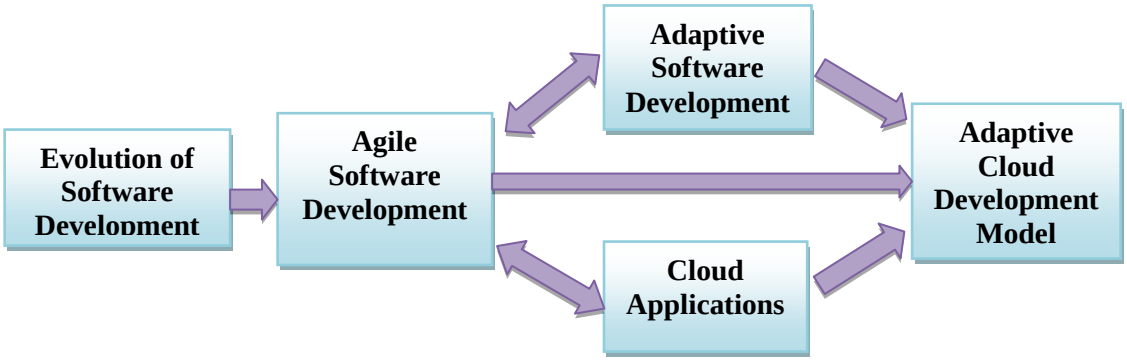

Figure 8. Flowchart of Adaptive Cloud Development Model

5. Results and Discussion

Table 2 sums up how the principles of the Agile Manifesto (Table 1) support the framework of different aspects of software development (Analysis and requirement gathering, Design and architecture, Implementation, Testing, Project management and Customer satisfaction) along with cloud computing. The result of the improved agile method has been described in the Table 2.

Table 2. Phases of Agile Manifesto Principles with Adaptive Cloud Development Model

Principles/Phases	1	2	3	4	5	6	7	8	9	10	11	12
Analysis and Requirement Gathering	++	+/-	++	++		++			+	+	++	
Design and Architecture	++	+/-	++			++			++	++	++	
Implementation	++	+/-	++			++			++	++	+	
Testing		+/-	+			+			+	+		
Project Management	++		+	++		+	++					+
Customer Satisfaction					++	+		++	+		+/-	+

Abbreviation + means that a principle supports the mentioned issue, ++ means strong support and +/- means that a principle can either support or not support the mentioned issue. An empty cell that a principle does not address mentioned issues.

Table 3. Assessment of the Adaptive Cloud Development Model

Design Criteria	Phase supporting Criteria	Remarks
Supporting Commonality (C)	Domain investigation	Recommended C & V methods
Reachable via internet	User Interface design and deployment	UI design for browser and Deployment for internet access
Providing Whole Functionality	Domain investigation	Technique to scope SaaS
Supporting Multitenant Access	Database Design	Architectural consideration
High Reusability	Domain Analysis	Proposed C & V methods
High customer satisfaction	Architectural Design	Architectural consideration for these quality concerns
High Quality	Architectural Design	
Reduce Time to market	Software Reuse	White and black box reuse
Reduce cost	Component based development	Product line Architecture

With Adaptive Cloud Development Model, the level of cloud customer satisfaction has been increased as compared to traditional agile development. The services provided by cloud provider will easily change according to requirements of cloud user. The evaluation of Adaptive Cloud Development Model has been summarized in Table 3 and the roles of various members of the adaptive cloud development model are described in Table 4.

Table 4. Separation of Roles Adaptive Cloud Development Model

Development Phases	Roles	Description
Requirement Gathering	Cloud Provider, Software Developer	Elicitation, Resource accounting
Analysis	Cloud Speculator, Software Developer	Software Architecture, Planning
Design	Cloud Speculator, Software Developer	Interface Design, Estimation
Coding	Cloud Collaborator, Software Developer	Construction, Implementations
Testing	Cloud Learner	Integration and System Testing
Deployment	Cloud Learner, Cloud Provider	Operation and Maintenance

6. Advantage of Proposed Approach

Essence of Adaptive Software Development is rapid software development with reduced overheads. This proposed Adaptive Cloud Development model will help to 1) developing application quickly 2) improve customer satisfaction 3) reduces cost 4) improves reusability 4) reduce time to market and make searching faster based on classification of components and introducing reusability in Agile Software Development. It will be accepted widely if pattern based architecture designing, design patterns, UML based analysis and designing is incorporated. The six important ways Adaptive Cloud Development Model enhances cloud based development:

- Adaptive cloud development model provides an unlimited number of tested and staged cloud based applications.
- It turns agile development into a truly parallel activity by software reuse.
- It encourages innovation and experimentation, if a feature or a story looks interesting, a team can spawn a development instance quickly to code it and test it out.
- It makes more development platforms and external services available.
- It eases code branching and merging.
- It enhances continuous integration and delivery of cloud based components.

7. Conclusion

In this paper, we have presented an Adaptive Cloud Development Model. The objective is to minimize the complexity, cost, time to market and increase quality, reusability, resource utilization and cloud customer satisfaction. Adaptive Software Development is encouraging future of the software industry and is capable of fulfilling the requirements of the cloud

industry. Thus, at times it compromises with quality and is incapable of providing reusability of its cloud based developed components. Adaptive Software Development offers some particular solutions whereas Reuse and Cloud based Development believes in generalized solutions to satisfy the demands of cloud customer. Cloud computing is a standard shift over the traditional way of developing and deploying software. This will make software engineering more challenging as cloud customer has to relate to a third party called the cloud provider. The amount of effort required for evolving software will diminish but communication and coordination requirement will be added to the cloud provider which makes software development project more difficult. The main objective of this paper is that the leading software process models should incorporate this new dimension of interaction of the cloud provider and separate roles namely software engineers, cloud speculator, cloud collaborator and cloud learner. A new Adaptive Cloud Development Model is proposed in this paper which includes the expected communication requirement with the cloud provider, cloud speculator, cloud collaborator and cloud learner which will diminish all the challenges of software development on a cloud computing platform and make it more beneficial to develop and deploy software on the cloud computing platform. Cloud based software engineering and agile development is an open research area in fast growth.

8. Future Work

Future scope of this work is to analyze and to incorporate risk factors in Adaptive Cloud Development systematically and find critical success factors of the Adaptive Cloud Development process and also identify various risk factors using risk analysis of introducing reusability in agile cloud development and offer a model that will help us to achieve reusability in Adaptive Cloud Development. Reusability can also be automated in agile cloud development using an automated tool. The automated model will reduce the development cost as well as increase the reusability and customer satisfaction to a large extent.

References

[1] K. Beck, "Manifesto for Agile Software Development", Agile Alliance, **(2012)** June 14.
[2] Adaptive Software Development: A Collaborative Approach to Managing Complex Systems, Highsmith, J.A., New York: Dorset House, 392pp, ISBN 0-932633-40-4, **(2000)**.
[3] Agile Project Management: Creating Innovative Products, Addison-Wesley, Jim Highsmith, ISBN 0-321-21977-5, **(2004)** March.
[4] O. Salo, "Enabling Software Process Improvement in Agile Software Development Teams and Organizations", ESPOO 2006, VTT Publications 618, pp149 +app.96 pp.
[5] A. Manifesto, http://www.agilemanifesto.org, last accessed: **(2012)** May 8.
[6] A. Murauskaite and V. Adomauskas, "Bottlenecks in Agile Software Development using Theory of Constraints (TOC) Principles", Master's Thesis, Gothenburg, Sweden, **(2008)**.
[7] S. Cronholm, "Using Agile Methods? Expected Effects", 17th International Conference on Information Systems Development (ISD 2008), Paphos, Cyprus, **(2008)** August 25-27.
[8] X. Ge, R. F. Paige, F. A. C. Polack, H. Chivers and P. J. Brooke, "Agile Development of Secure Web Applications", Proceedings of the 6th International Conference on Web Engg., **(2006)**, pp. 305-312.
[9] R. S. Pressman, "Software Engineering", 7th edition, McGraw Hill Education, **(2009)**.
[10] J. H. Chuang, "Potential-Based Approach for Shape Matching and Recognition", Pattern Recognition, vol. 29, **(1996)**, pp. 463-470.
[11] H. Gomma and G. A. Farrukh, "Composition of Software Architectures from Reusable Architecture Patterns", Foundations of Software Engineering, Proceedings of 3rd International Workshop on Software Architecture, Orlando, Florida, US, **(1998)**, pp. 45-48.
[12] K. S. J. and V. R., "A New Process Model for Reuse based Software Development Approach", Proceedings of the World Congress on Engineering, London U.K, vol. 1, **(2008)** July.
[13] A. Monaco, (7 June 2012 [last update]), "A View Inside the Cloud", The institute.ieee.org (IEEE). Retrieved, **(2012)** August 21.

[14] D. Turk, R. France and B. Rumpe, "Limitations of Agile Software Processes", 3rd International Conference on XP and Agile Processes in Software Engineering (XP 2002), **(2002)** May.
[15] B. Prasad Rimal, A. Jukan, D. Katsaros and Y. Goeleven, "Architectural Requirements for Cloud Computing Systems: An Enterprise Cloud Approach", Springer Science Business Media B.V. 2010, J Grid Computing, vol. 9, **(2011)**, pp. 3-26.
[16] R. Guha, "Impact of Web 2.0 and Cloud Computing Platform on Software Engineering", 2010 International Symposium on Electronic System Design, pp. 213-218.
[17] F. Paulisch and A. G. Siemens, "Software Architecture and Reuse – an Inherent Conflict?", 3rd International Conference on Software Reuse, **(1994)** November, pp. 214.
[18] www.win.tue.nl/~mchaudro/cbse2007/managing%20CBSE%20and %20reuse.pdf.
[19] D. Garlen, R. Allen and J. Ockerbloom, "Architectural Mismatch: Why Reuse is So Hard", IEEE Software, vol. 12, no. 6, **(1995)** November, pp. 17-26.
[20] M. A. Awed, "A Comparison between Agile and Traditional Software Development Methodologies", Available at: http://pds10.egloos.com/pds/200808/13/85/A_comparision_between_Agile_and_Traditional_SW_development_methodologies.pdf, **(2008)**.
[21] K. Jochen, "Agile Portfolio Management", [E-book] Microsoft Press, Available at Books24x7 http://library.books24x7.com/toc.asp?bokid= 27540.
[22] S. Singh and I. Chana, "Enabling Reusability in Agile Software Development", International Journal of Computer Applications, (0975 – 8887), vol. 50, no. 13, **(2012)** July, pp. 33-40.

Authors

Sukhpal Singh obtained his B.Tech. (Computer Science and Engineering) Degree from G.N.D.E.C. Ludhiana (Punjab) in 2010. He joined the Department of Computer Sci. & Eng. at North West Institute of Engineering and Technology, Moga (Punjab) in 2010. He obtained the Degree of Master of Engineering in Software Engineering from Thapar University, Patiala in 2013. Presently he is pursuing Ph.D. (Cloud Computing) degree from Thapar University, Patiala. His research interests include Image Compression, Software Engineering, Cloud Computing, Operating System and Database. He is an active member of ACM and IEEE.

Dr. Inderveer Chana is Ph.D in Computer Science with specialization in Grid Computing and M.E. in Software Engineering from Thapar University and B.E. in Computer Science and Engineering. She joined Thapar University in 1997 as Lecturer and has over fourteen years of experience. She is presently working as Associate Professor in Computer Science and Engineering Department of Thapar University. Her research interests include Grid computing and Cloud Computing and other areas of interest are Software Engineering and Software Project Management. She has more than 50 research publications in reputed journals and conferences. She is currently supervising eight Ph.D. candidates in the area of Grid and Cloud Computing. More than 23 Master's theses have been completed so far under her supervision.