Agile methods in Cloud Computing

Shrinidhi Anil Varna
Dept.of Computer Science and Engineering
National Institute of Technology Karnataka
Mangaluru, India
shrinidhivarna.171co145@nitk.edu.in

Vikas B N
Dept.of Computer Science and Engineering
National Institute of Technology Karnataka
Mangaluru, India
vikas.171co152@nitk.edu.in

***Abstract*—This is a research paper on agile methods and the applicability of Cloud Computing in solving the problems commonly faced in two of the most used agile methods, namely SCRUM and Extreme Programming (XP). The approaches involving SCRUM have been done by Shrinidhi, and those involving XP have been done by Vikas. Here, we first give brief introductions about the respective software development life cycles (SDLC) involved, their working and the advantages and unique factors in using them. Then, Cloud Computing is considered, and the various components of the cloud are discussed. Then, each of the SDLC is discussed, and the problems faced by the Developers, owners, and clients in these methods are analyzed. There are four case studies discussed in detail in the approach section. The main purpose of taking these real-life case studies was to implement our approach on practical projects and to suggest some better solutions to those projects using SCRUM/XP (Extreme programming) with the cloud. The outcome of using our proposed agile method is to bring down the challenges faced during the project that are similar to those completed projects by adopting a modern and revised SDLC (Software development Life cycle). After these problems are identified, they are considered in the context of Cloud Computing and approaches to solve the identified problems are discussed. Then, the outcomes of Applying cloud computing as an approach are discussed. Finally, the research is complemented by discussions about how the research was done, what proposed research work could be continued and discussion about how else (what another context) could the research be done?**

Key abbreviations used in the mini project:
1. SDLC Software development life cycle
2. App application software
3. RE requirements engineering
4. XP extreme programming
5. SW software
6. BP models Business process models
7. SRS Software requirements specification
8. UML unified modelling language
9. SaaS software as a service
10. PaaS platform as a service
11. IaaS Infrastructure as a service
12. IT - information technology
13. SSM - Soft systems methodology
14. RUP - Rational unified process
15. SLA - Service level agreement
16. TaaS - Testing as a service
17. AUT - Application Under Test
18. SCM - SCRUM controller and management
19. VCS - version control system
20. PUB - Publish

I. Introduction

This research paper emphasizes on cloud computing enhancing the already existing SCRUM framework of Agile software development life cycle (SDLC) model for adopting in various software development projects. The meaning of the word agile can move quickly and easily. We will find out in the course of this paper whether this method is the one it means or needs slight modifications to be known by that name. Firstly, the reason behind choosing the SCRUM framework in agile SDLC over other famous SDLCs is that in the modern era, software development has ever-changing requirements which could not fit into other SDLCs. Thus it is clear that a flexible model was needed that could quickly adapt to this demand, and hence agile SDLC model came into the picture. In this, my proposed work is related to one such framework of agile which is SCRUM. The SCRUM is best suited to a team of developers who work in nearby geographical locations. This is because in this framework there are multiple sprints planned for the development phase which lead to multiple releases of a software application (app). To overcome the problem of developers of leading IT companies distributed in far away locations, the agile methodology had to be strengthened by giving the users an access which is a widespread access, i.e., cloud. This combination of agile and cloud provide a positive impact on IT companies. But for achieving quality, we need to test all functionality, so in our work, we emphasize testing assimilation using cloud services in an agile environment. Cloud team members can perform testing easily using Test-as-a-Service without increasing the cost of the project and get the results faster[5]. Agile management aims to reduce the problem of communication for the Sprint backlog list. The most important thing to be kept in mind while following agile SDLC is that customer is also a part of the team, and his/her requirements need to be developed and must be quickly delivered at the end of each sprint. Security, quick user feedback, time, reducing the overall cost of the product and enhancing the standard of the software are some of the features that are ensured when cloud and agile are combined. The agile method divides a project into smaller, achievable segments with requirements disclosed. These segments are well planned, developed and tested to maintain high-quality standards and to avoid any bug that might creep in any of the parts. To quicken this process, user requirements can be

delivered using cloud services. The software bugs can be removed quickly by running multiple tests on virtual machines which is a service provided by the cloud. The main goal of this paper is to show the benefits of linking the agile method with cloud computing. Firstly, a short description of agile methods and cloud services is given, and the gains of this connection are emphasized. The gains are demonstrated for the rest of the paper based on practical work. Suggested cloud models based on analysis of infrastructure and operation of the organization are also included in upcoming sections. After describing functionalities, the traditional agile development is compared with agile development in conjunction with cloud computing- The one is using only agile methods and the second one is using agile methods and cloud computing[4]. This comparison will conclude this paper.

II. Approaches and Methods

A. SCRUM

In this paper, SCRUM is the approach employed towards developing applications using agile methods. As discussed earlier that SCRUM gives flexibility for change in requirements, and yields some segment of an application at the end of each release or Sprint. The first stage in this framework is the requirements elicitation. Since most of the times the initial requirements are incomplete, they are overviewed by the developers. They then proceed this by keeping a planning phase where they plan what are the requirements that must be present in the application. They then design the application using UML diagrams or any such diagram or approach that can represent the app clearly by showing all the functional requirements in the form of diagrams. This is done in software designing phase. At the end of planning and designing phases, the whole project is divided into small segments known as sprints, and the development phase commences. The goals to be achieved during that Sprint are to be contained in Sprint backlog of each sprint, and accordingly, team members distribute their work among themselves and review it at the end. A meeting in is held after every sprint in which team members discuss what they have done and what needs to be done.

The testing phase plays an important role. Again there are testers who do testing in the planning phase, development phase, and post-release phase in the planning phase. The testers make the developers understand what their requirements are, how they have dependencies and ensure that every developer achieves the same level of understanding in the planning phase about what the software is all about. In the development phase, the testers test individual modules followed by dependencies and report the developer if there exist any bugs in their requirements in either way. This happens in the development phase of each sprint. The test cases can be automated. Some test cases that are better off being manually tested can also be tested. The last one is the post-release phase where more time is available for testing the app, and the feedback can be collected from the clients. This phase can be made us for finding any obstacles present in the modules and can also help in improving the next module.

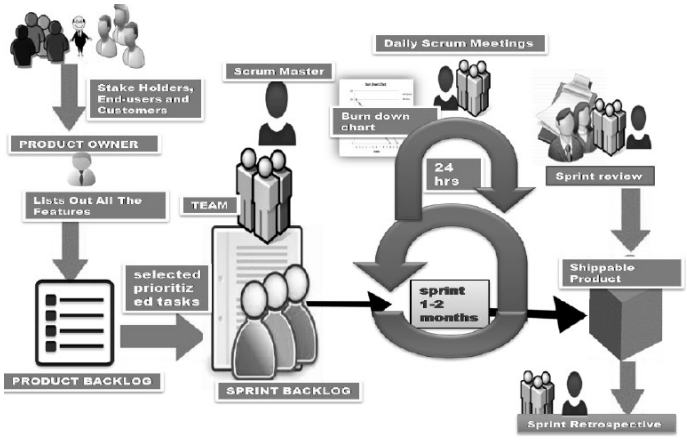


Fig. 1. SCRUM

B. Agile method and Cloud Computing

Some of the benefits or services that are provided to agile by cloud are:
Infrastructure : Cloud provides all those resources that are needed for the app at lesser expenses.
Frequent communication : Some project teams might be distributed across the globe and to make frequent communication happen between them, they require cloud services.
Automated testing : Some automated test cases can be generated and run on the app using cloud which saves manual work.
Servers : There is no need to wait till some departments work is over and they passing on the project to your department. Everyone can work simultaneously which saves a lot of time and can yield better quality app.
Distribution of software : It is easy to to distribute the software to each developers and clients using cloud.

C. Analyzing the strengths and weaknesses of SCRUM

Some other agile methodologies were taken for studying their relative strengths against those of SCRUM to find the weaknesses in SCRUM. We can always suggest the strengths of other agile methods to be included in SCRUM to optimize it further. To know all of this, we need an evaluation matrix. Evaluation matrix considered Soft systems methodology (SSM) and Rational Unified Process (RUP).
After evaluating based on these standards here are some of the weaknesses that need to be made a note of:
1. Process of working is not structured well
2. large-scale organizations are unsuitable
3. Lack of accurate documentation and artifacts
4. problem-solving phase being absent is a big problem
5. Not architecture-based

6. No activity of building prototypes or conceptual model of the product.[1]

We also realized some strengths of SCRUM which need to be appreciated and must be followed for making the best use of the practices, and they are:

1. To adapt or combine the new knowledge, flexibility of SCRUM can give better performance.
2. All iterative deliverables of the SCRUM can be evaluated by using the Sprint retrospective phase.
3. SCRUM is a simple framework that became famous by following the required agile principles.

D. Testing on Cloud[1]

Cloud-based software testing is the testing done on the cloud-based environment and infrastructure by combining cloud technologies. Testing in the cloud is a combination of the cloud environment and simulation of real-world user traffic as a means of stress and load testing websites.

1) *Objectives behind cloud testing:*
   i) The quality of application deployed in the cloud are ensured which includes their scalability, system performance, business processes, functional services, etc.
   ii) For validating software as a service in the cloud including security, scalability, performance, based on predefined Service level agreement (SLA)'s and certain economic scale.
   iii) The API of software as a service is checked along with their connections with each other.
2) *What is the significance of testing on the cloud?*
   i) Since virtual resources are used, there is no need to purchase too many hardware and software licenses. Hence bringing down the cost.
   ii) To perform effective large scale testing of web-based applications can be done on demand by test services.
   iii) Easy to test scalability and system performance.
   iv) Automated testing and development of resources help in reducing the cost of operating system and labor up to 50
   v) The setup time of development and testing is reduced.
3) *What is testing as a service (TaaS)?*
   It is the most effective way to provide the testing service to the organization or client as it accelerates the testing process by providing the third party with automated well-equipped test labs which includes testing tools, perform Application Under Test (AUT), and monitoring of data, automates the testing process which reduces the testing time and the cost and helps the companies to be faster in delivering the applications to their customers or clients.
4) *Characteristics:*
   i) TaaS provides with the test labs with full configurations, tools which helps the organization to deploy their test cases and complex web applications.
   ii) Ensures security by preventing unauthorized access by having a secure library.
   iii) TaaS has automated rich application for tracking errors in test cases and for diagnosis.
   iv) Since team members are geographically separated, they could require the test servers at different time, so TaaS serves the members at any time on demand.
   v) Test environment is created on demand along with testing tools and virtual machine which has all required virtual computing resources and infrastructure.
5) *Services[1]:*
   The different services are listed below:
   i) All the details of account management, access control, billing, pricing is included in TaaS management.
   ii) Test tool deployment, resource allocation, environment configuration, testware management are controlled by test environment management.
   iii) All the activities like scheduling, running, recording and bug reporting in a test are all controlled by on-demand testing.
   iv) Program behavior can be tracked through this service by the tester.

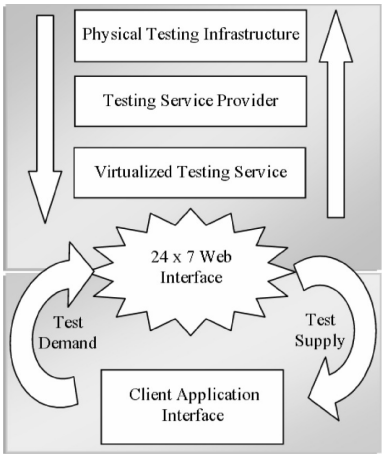


Fig. 2. Testing

6) *Differences between traditional testing and TaaS*
   Traditional testing needs labor which is costly whereas TaaS doesnt need labors. There is a focus on people in the former, but in latter, the focus is on service. The former is rigid whereas the latter is flexible and scalable on demand. Input-based pricing in done in former whereas in latter outcome-based pricing is promoted. Large resource sharing is a characteristic of TaaS whereas in former there exists limited sharing.

7) *Agile Development testing Using Cloud Computing to provide the necessary acceleration:*
   Templates allow the team members to duplicate the environment. While working in cloud developers can maintain a library of virtual machines which help them to combine the components or resources needed by them and template library should be updated with all the latest

available resources. With a cloud platform, there is no need to perform the task of connections or defining firewalls since cloud provides companies distributed and easily accessible source code management without managing the infrastructure. A copy of your development environment is created on a click. It also helps the testers to create many copies of code to be tested. While developing cloud helps the developers to work together with each other in the same environment in real time where all the bugs can be resolved, and testers can test iteratively without which there might be a possibility of code being able to run on one environment and not on others. Once the user tests the functional bit and approves it, then the virtual machine can be destroyed after fixing all the bugs.

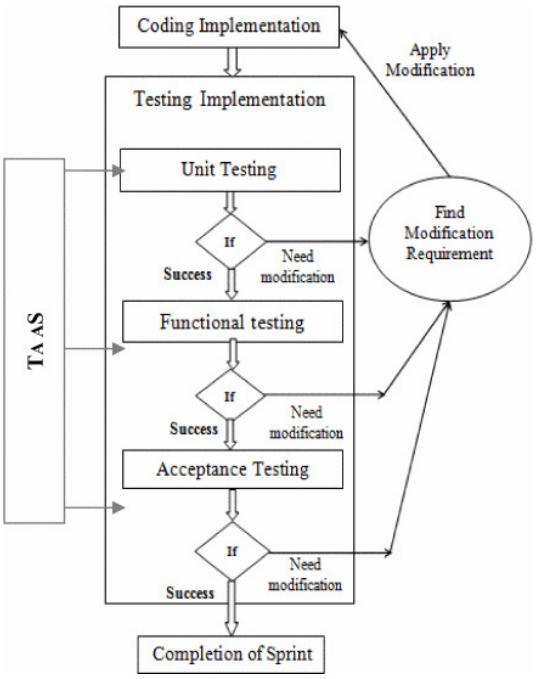


Fig. 3. Testing process in scrum[1]

8) *SCRUM over Cloud*

In this survey, I have chosen SCRUM over the cloud for development process management. Team members should work to achieve their goals in the environment where the needs of customer change very often. This is provided by SCRUM. As the organization is geographically dispersed, so cloud helps them to work together in a team. The agile management tools can be communicated with thousands of team members without managing IT infrastructure easily by using the cloud platform. The three phases of SCRUM are organized here on. Firstly, the user gives his requirements in the form of story to the organization and stores it in a storage space using storage as a service, of organization's private cloud in the pre-iteration phase that includes planning and designing of the software. After gathering the requirements validity and feasibility of the requirements are checked based on the practical scenarios and the availability of resources. After checking the feasibility of the user story, it is again stored in storage space, and now it is the job of the planning team to plan and design team to design the requirements for software and product backlog is also created. Now the development phase begins where the SCRUM master and the developers allocate certain requirements amongst themselves for each iteration called Sprint, and in each sprint, some amount of code is written which is integrated using automated development. Test cases are given to the individual modules and the integrated software in each sprint using TaaS to remove any bug. The role of the SCRUM master here is to keep the focus of his/her development team into developing the software and cut away all the possible distractions during this development phase. Then finally after developing and testing in each sprint, a release of the software is sent to the user for acceptance testing. Companies can also perform multi-platform testing with the help of virtual images and can run unit test parallel through cloud machines in spite of testing consecutive tests on one machine. Daily SCRUM meetings are held and managed by agile management tools. Lastly, in post-iteration phase once all the sprints are over, the software is integrated and tested using TaaS. The cloud is used in the first phase for storing the requirements those were first given to development team as user stories and later filtered into only the feasible requirements given by the planning and design team with the help of the SCRUM master. The development team during the sprints write few lines of code in each sprint, test it using TaaS and integrate it using automated development. Lastly, the software is given to the user for acceptance testing.

The phase SCRUM controller and management (SCM) completes the development. In this phase, all the backlog requirements are verified, and few steps are followed. Firstly, the dependencies of the modules are checked. The requirements that are needed for execution are found. Virtual machines are assigned as per requirements, and daily progress of each sprint is tracked. A developer can go for automated testing which will save time and money by not testing each test case manually. After testing of individual modules has been done, the individual modules are sent for acceptance testing along with the resources used for TaaS. After the client has accepted the modules, they are sent to Integration and Management control for integrating the modules.

9) *What is integration and management control[17]*

This unit is used for storing the sprint results and also to integrate the modules which are dependent on other

modules. After integrating such modules, the integrated module is tested and then given back to the developer. The controller is also used to deliver the results to the user acceptance testing. Tests like migration testing, user acceptance testing, performance testing, load testing can be performed on the application that has been developed after all the sprints are over.

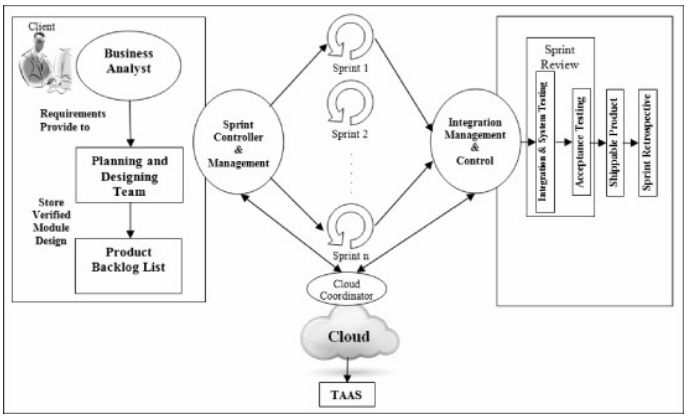


Fig. 4. Integration and management control and Sprint controller and management[17]

E. Illustrating the approach in practical real life scenarios:

Case study I:

Project name: Artisans android project

Aim: To develop an e-commerce app to minimize the cost of middlemen to the artisans.

Description: This application has a producer-consumer prototype. An artisan can register and display the products that he/she can manufacture with the necessary details of the product. The buyer can view this product and can place the order along with a delivery date. The order request is up to the artisan to accept based on various factors which include workload, time is given to manufacturing, illness, unavailability of the raw materials, etc.

Risk factor: Low

SDLC: Agile

Framework: SCRUM

SCRUM Team strength: 6

No. of Sprints: 2

The team delivered user stories on a weekly basis, and at the end of the first sprint, there was a review to list what are the user stories not completed. They were completed in the backlog week which is the week following sprint I. The sprint II began the day when backlogs were completed. The user stories in planned to be completed in sprint II were completed and were delivered to the product owner for the final review. If any backlogs were there, they had to be completed along with the testing of all the requirements.

Result: The application was developed quite well in this short period spanning three months. The scrum master and product owner gave their suggestions upon delivering the requirements at the end of each week and each sprint respectively. User acceptance testing was done in the end. The application went through the manual testing phase, and the bugs which were realized were fixed before the deadline. Still some bugs remained, but the project has come to a close. The designing and documentation of the requirements were done during the development phase. Git was used as a VCS throughout the project for software development and integration of requirements.

Violations:

i) The agile SDLC was not followed in the right order. Incident: The UML document was asked to be done after the development phase was begun and not during the design phase.

ii) The testing phase didnt last long enough to remove all the realized bugs.

iii) Only manual testing could take place.

iv) There was no planning and designing for the new requirements that were taken up during sprint II.

v) There was no automated testing for the app as a whole.

vi) The team coordination was poor.

vii) The SCRUM master wasnt successful in playing this role.

viii)The product owner merely focussed on the UI of the application instead of giving attention to the other core requirements that were taken up.

ix)The daily meetings never took place and there used to be a meeting every week.

x) The project was taken up with all the developers in the team having several commitments due to which the project wasnt developed as imagined.

Pros: The benefits of this project are also worth mentioning. The Scrum team learned about SDLC, android development and other aspects related to planning, designing, and documentation. The project almost neared its ultimate goal but fell short by a small margin which couldve been overcome if a testing phase was extended. The project gave the group of fresh developers in the field of Android development a belief that they could develop a lot having all the necessary tools and information.

Implementing the approach proposed in this paper on this Artisans project:

i) The first step in my approach would be to not violate agile SDLC.

ii) Secondly, the team members will be chosen in such a way that they dont have any other professional commitments. This will ensure that the SCRUM team will be focussed in one thing only and it will be easier for the SCRUM master to keep them focussed.

iii) An SRS is made. The project is planned to be completed in N sprints, and the user stories are collected in the beginning and divided across these sprints and assigned to the developers based on their experience or interest. Any new user story at the beginning of a sprint needs to be added to the SRS without resisting to adapt to any changes.
iv) The user stories will be given the importance in the daily meetings which can happen face-to-face or through video conferencing.
v) The sprint will begin with planning and designing the requirements followed by a development phase. Since my approach has recursive testing using the cloud(i.e., TaaS), this can be implemented during each phase in a sprint.
vi) After the development phase we keep a check on the backlog and try to minimize it since it isnt a good practice to have one or more backlogs after a sprint. The user stories that are listed in the backlog are developed during this phase and later tested. This phase is followed by an integration phase where all the individual requirements are integrated into software, in this case, app, and sent for automated testing. There might be a lot of difficulties faced while integration, and hence we use a VCS to save some of our time here which can be used as an when a requirement is completed. So there is no need to wait longer. We can have access to cloud platforms if needed anytime, anywhere. Once the integration of all the requirements are done, before delivering the sprint deliverable, we send the app for automated testing and based on the testing reports, the concerned developer or developer(s) in case of some dependencies are told to fix the issue.
vii)The app is shown to the product owner, and a user for critical analysis and any suggestion or changes required are incorporated in the coming sprint.
viii) We repeat steps 3-7 for N such sprints and finally deliver the app which can be now sent to the user for acceptance testing.
ix) Once the user is satisfied and there arent any user stories left, the project can be said to be closed.
x) The app needs to be maintained by a team of developers in case there is a need to add some more features to it, and users demand more or find a bug in it.
xi) In step 2, we have assumed the commitment of developers through till the completion of the project towards development.

Case study II: Distributed Scrum Project for Dutch railways
Aim: To build a PUB(publish) system that can centrally control information displays and audio broadcast systems in all stations.
SDLC: Previously waterfall
Proposed SDLC: Agile SCRUM using cloud
Risk factor: High
Failures when waterfall model was adapted for this SW project:
Detailed SRS was handed over to the IT vendor, expecting a fully built system to materialize without much further customer involvement. After three years, the project had to be canceled because the vendor failed to deliver a working system. The customer than demanded our company to build the PUB system from scratch.
New approach: a Agile approach using Scrum was introduced which focussed on close cooperation with the customer and open communication and working in small increments.

Challenges: It was difficult to find a single product owner. Two business analysts were nominated to be good proxy product owners to several groups of customer stakeholders. The project manager made high-level decisions about priorities and scope mandated by the customer who had less knowledge about requirements. In general, the model worked well except when the project manager used to change the priorities that had just been set during the planning meeting in his absence, and the meeting had to take place again. The person who has the final say about the priorities must be ideally present in the meeting.
Scaling up to distributed teams: The SRS was already present at this stage. It started by having a seven member team in 2-week iterations. The Indian developers worked on-site at the customer site for six weeks to become familiar with the application domain, key customer representatives and the rest of the development.
The important step here was for the team to agree to work together. To facilitate that we had a norming and chartering session with all team members including our colleagues from India. Pair programming, use of tools, quality targets, core hours, etc. were discussed in this session. In the first iterations, the team proved to be able to build, test and demonstrate user stories that formed the heart of the system. This pleased the customer, especially because - compared to the previous attempt - progress was demonstrated much sooner, and the customer had more control throughout the project.

After a couple of iterations we scaled up the project: the Indian developers returned home, and we added resources in India and the Netherlands to create two Scrum teams of 5 developers each, sharing a single tester. Later this scaled up further to 3 development teams with three testers. Key to our approach is that each Scrum team consists of resources in both India and the Netherlands. This model proved very successful in maintaining high velocity and quality. Skype was a medium through which the teams could communicate from different locations.
Challenges faced in my approach: To implement this distributed model the product owners did not feel comfortable speaking English. According to Scrum, the sprint planning meeting consists of two parts in the first part, the product owner clarifies the user stories with the team and sets priorities for the sprint. Due to the language barrier, we chose to do this meeting in Dutch without involving the Indian parts of the teams. The second part of a planning meeting normally consists of identifying and estimating tasks to implement user stories. This part was done in English together with the Indian developers using a Skype session, but without the product owners. Extra time was spent to communicate the contents

of the first meeting. We held our Sprint demos only locally and in Dutch. To update the team members in India, the Dutch team members wrote a newsletter about the demo, which was also distributed to the rest of the company. This newsletter proved to be a very popular item for all colleagues.

Requirements: Our Product Owners, who were business analysts, usually wrote extensive requirements documentation in Dutch. For our process, just user stories on a backlog and the explanations of the Product Owners during the planning meeting would suffice. However, the customer organization required extensive requirements documentation. To make this work with our Scrum process, we decided to translate the requirements to user stories in cooperation with the Product Owners. The result was that requirements were kept in two places: the requirements documents and the backlog.

Testing: We applied automated testing during the project to allow us to deliver tested software at the end of each Sprint, without regression bugs. Even as the system grew we managed to do this with only one tester per eight-person Scrum team while maintaining high quality: the external testing team found less than one defect per KLOC.

Outcome: The customer is very satisfied with the work we delivered. Hindsight shows that, as with most projects, the required functionality, time and budget shifted as the project progressed, making ”on time, on a budget” a vague measure of accomplishment. What's more important is that success factors were regularly discussed with the client while the work was going on, and they expressed satisfaction with the result. Unfortunately, its nation-wide deployment was hindered by problems in other systems that were also part of the rollout. The customer asked an external audit company to audit the software. Their conclusions: The maintainability of the system is very good, and the quality of the source code is very high.

Summary: These are the most important lessons we learned on this project:

i)hard to find a product owner with both detailed knowledge of the requirements as well as the mandate to set priorities and often unavoidable to have multiple product owners in large projects.

ii)it's important to make sure that the product backlog is complete and estimated before the deadline. For requirements, any estimate is better than no estimate, even if little information is available. This provides the necessary information for release planning.

iii)Scrum is well-suited to execution with multiple distributed teams. Having each Scrum team contain resources both in the Netherlands and India was good for team spirit and forced us to work on effective communication. For communication, off-the-shelf hardware and free software make the cost of implementing this low.

iv)It is useful to start a distributed project with an initial co-located session to reach an agreement on team practices.

v)Work that doesn't fit well in a Scrum Sprint (i.e., chasing down key people, interfacing with other customer departments) can be handled more effectively by a separate team. This allows the feature teams to focus on building the software. Using a dedicated technical writer also helps in this respect, even if it does add communication overhead.

vi)Although not needed for the software development process, extensive requirements documentation may still be required by the customer. In a Scrum project, however, this cannot replace the use of user stories. If both are used, the overhead of reconciling requirements in two places should be factored into planning.

v)Automated testing is vital to deliver software incrementally, unhindered by regression bugs. Before the project is over, the return on investment will outweigh the cost.

F. Issues in Agile models

There are some problems that exist in this method itself which have been found out by discussing with project managers, developers, and personal experience. Before we try using cloud computing to enhance the agile environment, it will be wise to make improvements in the existing method and later on take support from cloud to make the process quicker. Agile SDLC had three major problems which can be classified into three categories:

1. Development process conflicts,
2. Business process conflicts,
3. People conflicts.

We will be discussing these problems here one at a time.

1. Development process conflicts: It is difficult for the agile development process to take place along with other traditional processes since the other processes have longer development phase where quite a lot of development takes place at an optimum rate whereas in agile, the development takes place quickly because it aims at quickly delivering various functionalities. Hence it is difficult to accommodate these two together. The agile method gets the support in the testing phase and best suited as we have proved earlier. Agile is more suited for app maintenance. Application of agile processes to legacy systems, whether within maintenance or as new development, raises numerous issues. Legacy systems generally aren't easy to refactor or disassemble to accommodate agile replacements that need to build capability in increments. Legacy systems might also institutionalize awkward and often sclerotic business processes that are embedded in the culture and aren't easy to refactor away. The problems are also aroused by the differences in the way requirements are taken. In an agile method, generally, requirements are primarily functional and reasonably informal. The possible suggestions to avoid this kind of conflicts are to do some serious preparation upfront, build up processes instead of tailoring them down, define specific functionalities that are required to be addressed using agile process, redefine traditional milestone reviews to better fit an iterative approach, implement agile methods that support organizational priorities, determining the risk factor is the best way to assess how much agility is enough[2].

2. Business process conflicts[8]: Human resources- team oriented versus individual rewards, position descriptions and timekeeping, must be learned to accommodate by the organizations. HR departments and procedures frequently get

in the way of empowering people to pursue nontraditional approaches that require organizations to revisit legally vetted and audited policies and procedures. Progress measurement techniques might be sometimes inadequate to support the rapid pace of agile processes. An area of conflict for mature organizations will be in how agile processes will affect their ratings concerning CMMI, ISO, or other process standards. In my opinion, agile is in line with the concept of constantly adapting to improve performance, but most agile methods don't support the degree of documentation and infrastructure required for lower-level certification which makes agile methods less effective. Suggestions: Throughput accounting must be applied rather than cost accounting. Establish some standard characteristics and patterns that support separating stable requirements from those with evolving requirements. Development of incentives and contract provisions for client satisfaction that support agile environment. Work to eliminate as many clashes as possible while encouraging synergies while differentiating between compatible and incompatible clashes between agile and other traditional methods. Adopt empirical studies to identify which are the classes that are more unpredictable and hence more suited for agile method. Establish guidelines for safe and agility-compatible process maturity assessments[9].

3. People conflicts: This is by far the most critical conflict which needs to be addressed to adopt and integrate agile methods in our processes. Managers tend to associate employees with specific roles that might cause difficulty in the multitasking characteristics of agile team members and most agile methods, they play two primary roles: protector and coach, acting as a barrier between the organization and the team to minimize unnecessary perturbation during a sprint or development cycle and provide experienced technical help when necessary. Agile teams must nearly always be colocated since people are directly affected by some logistical issues. Pair-programming stations, walls for status charts and assignments, a layout that allows team members to easily converse to share information, and sufficient equipment to support continuous integration and regression testing are what is needed for a typical agile workspace. The negative impacts of how organizations handle the success of pilot projects are often overlooked in reporting outcomes: fire or promote the manager and split up the team. This in return destroys team relationships, both technical and personal, diluting the knowledge gained and lessons learned sending the message that trying new things might be hazardous to your career. From my personal experience, concerns of inadequacy or obsolescence surface, jealousy about assignments and business accouterments is aroused, and defense mechanisms rapidly deploy when there is a change in mechanism brought into the project. This can result in several destructive behaviors, including the cultural crucifixion of change agents or early adopters and the deliberate sabotage of projects through direct or indirect methods which can harm the team on the whole and overall morale of the team will go down.

Suggestions: Understand how communication occurs within a team of developers. Educate the stakeholders by sharing as much as you learn. Translate software and agile issues into management and customer language. Agile methods emphasize value in two ways. First, they negotiate and prioritize requirements so that expectations are managed and timeboxing can work. Second, they acknowledge the value of each team member, the team as an entity, and the products the team produces to the organization and the customer. Lastly, dont minimize the efforts put by a developer and the development team. Pick the right people and assign the right tasks based on the skillset of the team members.

G. So, what is the conclusion?:

Somethings that can be concluded here are that the cloud brings in a lot of benefits with it when we opt for cloud for developing a product. We have simulated SCRUM framework to develop an application which has been possible with less cost, less manual testing, all-time testing environment, less amount of time, less investment in storage hardware needed to store immense amount of data, data security that was ensured, serviced-based automated testing, TaaS implementation in our framework, integration of individual modules, etc. Overall there have many huge boosts to our SDLC in general and our framework in particular. For future aspects, we may implement this method in other agile environments and do a comparative analysis. In this report, the approach that has been implemented includes recursive testing since we see quite a lot of testing involved in each phase and given quite a lot of significance. By following this approach, the SCRUM process can get quicker than it is when followed in traditional way.

H. XP - Characteristics and Advantages

Extreme Programming(XP) is an SDLC model, one of the most prominent among the several agile methodologies. Just like other methodologies[6], XP differs from conventional methods basically in giving a higher priority to adaptability than on predictability. Supporters of XP say that frequent changes to requirements are a natural aspect of software development. They believe that adapting to changing requirements at any point during the development of the project is a much realistic and a better approach than trying to define all requirements at the beginning of a project and then putting effort to control changes to the requirements[8]. This method prescribes a set of day-to-day practices for managers and developers; the practices are meant to embody and encourage particular values. These traditional software engineering practices are taken to extreme levels, resulting in the making of a development process. This process more is responsive to the consumer needs than the other traditional methods, while creating software of similar quality.

Though XP and SCRUM seem similar, there are specific differences between the two[12].

i) SCRUM teams work in iterations (here, sprints) of

duration anything between two weeks and two months whereas, in XP, the iterations are one or two weeks long.
ii) In SCRUM, once a sprint is planned, no change in the sprint is possible. But, in XP, changes can be made within the iterations involved.
iii) XP teams work in strict priority order. The team follows the priorities given to the requirements by the Product Owners. This is not the case with the SCRUM method, where the developers have the liberty to choose, reorder and prioritize the requirements according to their comforts and understandings.
iv) SCRUM does not prescribe any of the engineering practices. But, XP prescribes various practices, including pair-programming, focusing on automated testing, test-driven development, refactoring, etc..

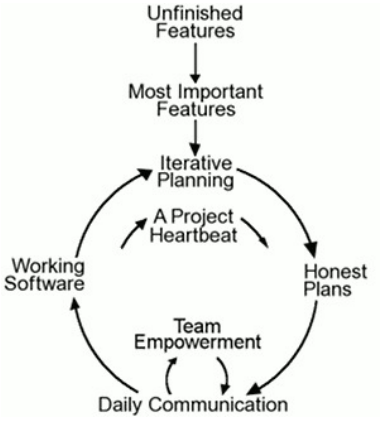


Fig. 5. Flowchart of an XP method[3]

As time passed by, the high discipline required by more traditional methods made them less interesting, eventually leading them to be sidelined. Agile methods, on the other hand, havent remained the same. Through time, theyve been improving from valuable lessons from experts working in the field.
The purpose of XP is to reduce the development time and the cost of change in the requirements involved. In traditional methods, all of the requirements are decided before the development phase begins and theyre fixed afterward, making it least supportive of changing afterward. XP works towards lowering the cost of change by introducing fundamental values, practices, and principles.
XP recognizes five values. They are:
i) Communication
ii) Simplicity
iii) Feedback
iv) Courage
v) Respect

The building of a software system needs communication between the developers of the system. In traditional SDLCs, this is accomplished by documentation. XP methods are

Fig. 6. XP values

methods to rapidly build knowledge among team members. The goal is to provide the developers with the idea that the users have about the project. XP favors simplicity in design, a collaboration between users and developers.
In XP, initially, the most simple solution is designed first. Then, the other efficient solutions are arrived at. The difference between this approach and more traditional SDLCs is the importance given to coding and designing for the necessities of the present and not for those of the future. Users of XP admit the disadvantage that this can sometimes result in putting more effort tomorrow to change the system. But they also claim this is negligible compared to the risk of investing in a requirement keeping the future in mind and then that requirement changing before it comes into the picture. About the communication value of XP, simplicity in coding and design should improve the quality of communication. Most programmers in the team can easily understand a simple design with a simple code.[11]
Feedback happens in three different ways,
i) Feedback from the system (Flaws in the system obtained)
ii) Feedback from users/stakeholders (i.e. User Stories)
iii) Feedback from the team (estimation of required time and effort)
Feedback is most useful if it is done rapidly. Its critical in learning and making changes in the software from time to time.
Many practices embody courage. One instance where the developers courage is tested is while designing and coding for the present and not the future. This effort is to avoid getting stuck and requiring a lot of effort in making the ends meet in case of a change in requirements. Another courage is to know when to throw a developed code away, irrespective of the amount of effort taken to design and develop it, when it becomes obsolete. Also, courage means tolerance and perseverance. A developer might get fixed on a difficult situation of a problem for a whole day, and then find the solution and solve the problem, only if the programmer is persistent.
In XP, team members need to respect each other because programmers are never supposed to commit changes to the

software that affect compilation, that makes its testing fail, or those which delay the work of their teammates. Members need to respect their work by always trying for high quality and by seeking for the best design possible by constant refactoring.[6]

XP is most suited for projects like

i) Projects in which requirements change rapidly, and there are chances of incurrence of implementation problems.

ii) Projects which involve research where the result is not any software but knowledge or inferences made by research.

iii) Projects which are small and can be managed by easy informal methods.

Before considering the identification and analysis of problems

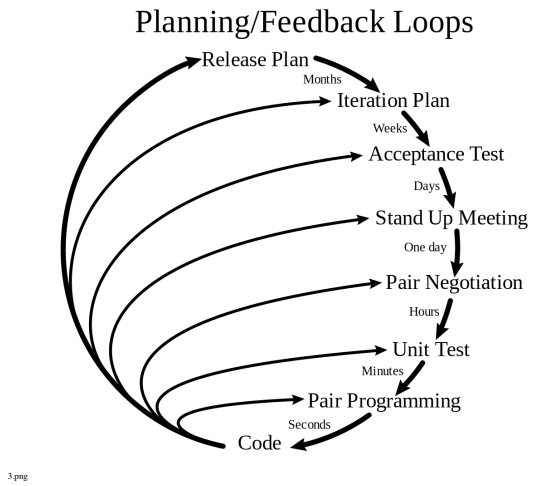


Fig. 7. The iterative working of XP

in XP, we'll look at some real-life case studies of projects done in XP.

Case Studies.:

Study 1

Womens Era (Web application)

Aim:

To develop a web application portal for women empowerment, support, and safety

Description:

This is an online portal, especially for women where they get knowledge about the laws and Legal Matters related to Women, they get various notifications regarding academics from Universities in single point access, they get expert lectures and study material. They get information and knowledge about healthcare, related to the precautions and preventions of various diseases and also get suggestions and tips from doctors and they get to know about various government schemes related to women and children.

Risk factor: Low

SDLC: Agile

Framework: XP

Problems/Challenges identified:

1. It is difficult to predict in advance which software requirements will persist and which will change. It is equally difficult to predict how customer priorities will change as a project proceeds.
2. For many types of software, design and construction are intertwined.
3. Analysis, design, construction, and testing are not very predictable.
4. The iterations should be adaptable to change or re-prioritization of the requirements in an iteration.

Solution:

If requirement takes ¿3 weeks for development, can be divided into smaller requirements, prioritizing them as the developers want, without intrusion from customers until testing. This improves adaptability and invokes a sense of independence in the development team. Moreover, prioritizing from the product owners can be minimized, further easing things up for the development phase.

Study 2:

Telegram App (Mobile application and web portal, with cloud storage)

Aim: To develop an application for secure messaging and calling

Description:

This is a messaging and media app, supported on mobile platforms as application and browsers as a portal. The platform has a feature for creating channels, chat groups, stickers, and calling. The messages are completely cloud-stored, with minimum caching in the devices storage.

Risk factor: High

SDLC: Agile

Framework: XP

Problems/Challenges identified:

1. Imagining the scale and the amount of risk we are taking in a requirement beforehand(i.e., before design) is difficult.
2. Suggests the inclusion of one more phase- Analysis and Risk management.
3. Also suggests that XP is not much suitable for high-risk projects.

Solution:

1. This phase should be after planning.
2. This phase provides support for component-based development.
3. This phase encourages independence and provides support for a distributed development environment.
4. And hence, it encourages us to have a large team size.
5. It results in better documentation helping to software engineering team during and after development (e.g., Risk Management).

The end-product/result of the risk management phase will be the SRS (Software Requirement Specification) document. The improved XP process model is better than XP because it eliminates the limitations of development of reusable

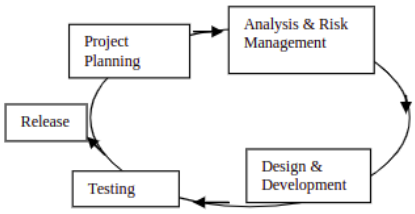


Figure 1 An Improved XP Process Model

from 2019-04-09 23-08-57.png

Fig. 8. The suggested new phase in XP

components, large development teams, documentation, medium and large Software projects.

Problems identified in XP:

Currently, the most common problems identified in project management using XP are as follows :
1. Disregard about analysis and design work by developers.
2. The developers are forced to work one person at a time to avoid the mistake of different people working on different versions of the project.
3. The iterations, sometimes, become too fast to accommodate proper planning for each requirement involved. It just gets back to meeting the deadline hurriedly without a proper insight on what is to be developed, which was one of the key reasons agile methodologies were adopted.
4. As XP is an agile method, owing to the lack of a framework, new inexperienced developers will have no exposure and fail to understand and comprehend the methodology.
5. Again, there are high chances that newbies arent familiar with the required technical and various features of the project under development.
6. In XP, if constant, clear communication in between the development team (leaders and designers inclusive) and that between them and the users is absent, there are high chances that the project isnt completed.
7. In XP, the priorities for the requirements in every iteration is pre-defined by the product owners, which needs reasonable flexibility for the developers to be able to prioritize or choose to do what they are good at or those that they want to do.
8. When proper documentation is disregarded, there can be chances of the method losing its significance and the stakeholders being unsatisfied with the work, especially when the iterations yield little or dont yield at all.
9. The real virtue of XP lies in its principles like swiftness, feedback, unit tests, and pair programming. Non-co-operation between the team members leads to the sidelining of these principles, which eventually is a hindrance to the software under development.
The above problems sum up the frequent complaints and problems in corporate software development. Most of these can account to communication and coordination hindrances, incompatibility with the schedule and lack of effort in certain areas.

When we try to solve these problems, that is when cloud computing kicks in. With cloud computing, there can be a whole new approach to solve the problems discussed above. Cloud computing and XP can go hand-in-hand, complementing and contributing towards the betterment of each other.
How cloud computing could contribute towards XP will be
1. A development platform designed for the development of the software, specializing in the easing of the development of the software.

This can be accomplished by the Infrastructure as a Service (IaaS) model of the cloud where the developers can create and manage projects, with various tools for programming. This helps in simplifying the coding part, where the platform is intelligent, i.e., they can smartly suggest code for auto-completion. They also have in-built templates for module creation. This platform is a great help in making programming easy and smooth, especially for people with less experience in project development.

2. Frequent communication between the team members, project management and users.

This feature proves to be a virtue for the team and the stakeholders, as it pivots the communication service. This is accomplished by the Software as a Service (SaaS) model of the cloud with efficient, user-friendly messaging components. This will cut short the costs of communication by a very big margin. It enables the team to be in touch with each other, having proper discussions and sharing ideas about the requirements. The best part is that the team can be in touch with the users too. And that too continuously throughout the development. In the case of any anomalies, they just clarify directly with the user. With this, feedback gets a lot easier now. Moreover, its not that this wasnt possible earlier. But, these platforms have made the communication super fast. And theyre not just messaging platforms but they create a vibe of a workspace in themselves, and team assistance becomes a lot smoother.
3. Model sharing (Version Control Systems)

This is to provide a platform for the team members to contribute their requirements to the main project synchronously and continue developing on sync with the other members of the team. Version Control Systems (VCS) are instrumental in providing these requirements. It is a Platform as a Service (PaaS) model.
This is the main game changer in development. We cant even imagine all the developers working on completing their requirements on different versions of the same project. This would create a lot of confusion as it goes back again to the non-integrated phase, with integration becoming much more

complex. VCS is now a relief for the development team. It helps in synchronization of the project between all the team members. Whenever a team member updates a project, others can work on that version instead of the version they were earlier working on. The developers can update and change the project in the cloud without actually having to go and manually delete or replace the individual files. The VCS is compatible with the development platform, and the developers can do operations in the cloud platform by using commands in the development software itself.

4. Testing and Integration

The cloud can catalyze arguably the most important phases of any software development, testing, and that too with perfect ease. Testing as a Service (TaaS) model is the type of cloud which offers this solution. All of the testing services, if done on the cloud, will save a lot of money spent on testing on actual physical devices. Moreover, this doesnt need the amount of labor and time required by the conventional testing methods. Here, the cost/pricing is dependent on the outcome, rather than the input, which is the conventional method. Technically, cloud platforms provide an outsourced virtual testing environment where the test environment is set-up, and extensive virtualization is done, as much as possible. Use cases are designed and the inclusion of certain testing tools, the environment is done only on demand. Documentation is dynamically done, rather than the painstaking process involved in conventional testing. This is rapid, and also efficient as some have claimed that we can save 40-60% when testing is done on the cloud.

III. Results

A. SCRUM

In this paper, there has been an explicit comparison between the agile scrum framework with the other structures of agile. The reason behind choosing SCRUM over all the traditional SDLC models has been discussed. We have first introduced the SCRUM framework in its current existing way.

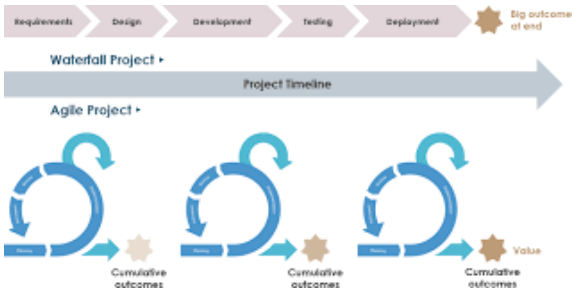


Fig. 9. Traditional SDLC v/s SCRUM

A comparative study is between SCRUM, and the other agile models were done to find the strengths and weaknesses existing in the SCRUM. The objective here is to try to reduce the number of defects and keep the powers (strengths) as it is.

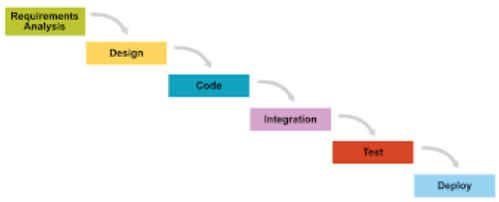


Fig. 10. Phases in SCRUM

One of the tools that can support this mission is cloud computing which brings along with it an immense amount of benefits. I have compared the SCRUM with cloud assistance with the SCRUM which is traditionally followed. The fact that the cloud increases the rapidness of the agile process proves the objective behind introducing cloud for SCRUM. The testing phase is now made a part of every sprint in the SCRUM and made recursive to proceed to the next sprint without having any bug(s) in the software. This testing is way faster than the manual testing which exists in the traditional method. Automated testing enables several test cases to be tested in a software parallel. In return, it brings down the cost spent on manual labor for performing tests hence bringing down the time taken in each phase for testing purposes. TaaS has thus served its purpose on this regard. The best way cloud supports the development team is by connecting the developers spread wide across the globe by providing them an environment which is available at low costs at all times and any changes in the code can be made at any time by any of the team members. There is no need for any team member to wait for someone to push their code and work on it. Instead, everyone can simultaneously work and advance their repositories.

Fig. 11. The new approach addition to the SCRUM framework, i.e., Cloud testing

Then the spotlight moved towards listing all the loopholes that exist in the SCRUM framework. There have been some suggestions that have been given which when followed can make the SCRUM function much more smoothly and can be adapted by development teams for a wide range of SW

development projects.

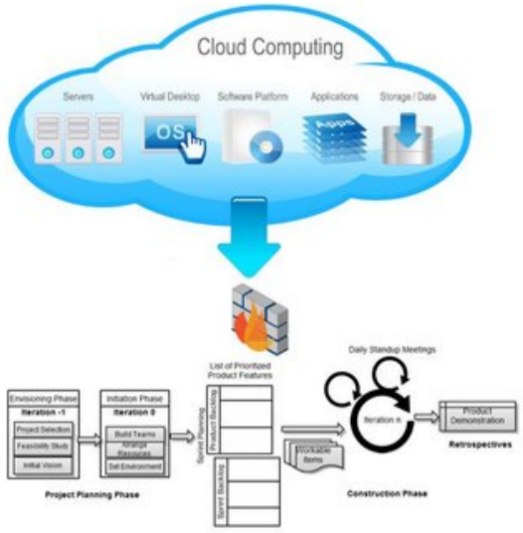


Fig. 12. Fig. Cloud support in SCRUM framework

At present, SCRUM is adopted by developers into those types of projects which have low-risk factor. This speaks about the mindset of the development team while choosing this method. When they feel like something can go well when some kind of experimentation is done by collecting requirements as and when they come, they opt for this method. Generally, this kind of applications is risk-free while they are being developed and the functionalities added to it are delivered as and when they are developed. But the sprints must be optimized too. For this, there must be better planning done before the commencement of any sprint which is also one of the suggestions that have been listed in the approach. The conflicts that arise are too many when the SCRUM isnt followed properly. It is not the team with the best developers which succeeds but the team which has good coordination. To remove such conflicts at the people level, organization level, and business level, the approach that has been discussed must be followed and taken seriously. As a result, it brings the best out of a developer and keeps the team morale high, eliminates any jealousy that might creep in some time while working on the project and ensures the deserved appraisal for the efforts that have been put irrespective of what the progress of the sprint is. The ideal way to adopt SCRUM is to make sure that the team which is picked is ready to work for the interests of the team keeping aside their individual goals and not thinking about individual benefits. If this is taken care then automatically a lot of things will come into place and the efforts as an individual and as a team will be appreciated. The best takeaway out of the solution that we have proposed is the effectiveness of the framework. As a result, we have obtained a team which has best developers, ready to coordinate with whatever is required to bring the best out of their team and the team's main goal is to work towards making the project a great one having all the modules fixed with uniformity being the common thing among all the modules. We have been successful in finding the exact role of a SCRUM manager, and he/she also has the task cut-out in managing the project members. If they stick to the approach that has been proposed in the paper, the people level and the organization level conflicts will be sorted out. The developers must always keep sharing the information which they get and want to convey to their clients and product owner to keep them updated with what is going on in the project. The stakeholders play an important role during the process. The user/customer acceptance testing is the final testing done on the deliverables of every sprint, and thus, the user satisfaction report is obtained. The best practice would be to have a client who is available throughout the project duration to critically analyze the software. The improvements suggested must be worked upon during the backlog phase.

B. Extreme Programming

The exploration of XP is done and presented as it is. The aim of this research is to offer solutions to the persisting problems existing in XP. We saw a potent approach towards solving the identified problems using cloud computing. According to Matt Stephens, a software engineering expert, XP is a symbiotic process: i.e., you need to do all of XP or none at all. There is no in-between. The theory is that every one of its individually problematic practices reinforces each other to produce something stronger.

Now, the challenge is that this statement can also be turned the other way around. That is, stop doing one practice and then the list of challenges lengthens. It becomes challenging to stick and adhere to the rules of XP. This clearly demands changes in the XP model for good. Hence, we try to incorporate changes in the model applying cloud computing. A comparison is made with the conventional method, which facilitates in providing an understanding of the improvements achieved when cloud computing was incorporated.

The problems encountered in the approaches are the main, decisive components of the success of the XP method. Solving them with cloud computing is the most feasible solution not only because it eases the XP life cycle, but also is very cost-effective in the current scenario. Most of these major problems can be solved using the cloud but some of them can be generally solved too.

For the problems discussed in the approaches, the proposed solutions are:

1. Disregard for design and analysis of the requirements, documentation through the development.

When cloud computing is adopted, most of the documentation and analysis is done dynamically, by the cloud testing service, which relieves the developers of the huge paperwork involved in testing and documentation. This can be a solution and also a motivation to be involved actively in the design phase, understanding the level of importance needed as this affects how the project is developed, rather than what, which remains

the same.

2. Synchronization problems between team members work.
Concurrent Version Systems(CVS), as discussed, provide a solution to this problem and also the time management crisis between iterations. These cut the time wasted while obtaining updated software from a teammate, which can also overlap with multiple people involved. This provides a centralized project management system for any kind of a team.

(11).png

Fig. 13. Fig. Working of CVS

3. Developers having less technical knowledge about the software under development.
Software development platforms provided by cloud computing service (IaaS) provide infrastructure to develop software with minimum knowledge. The software facilitates programming by real-time error checking (detecting errors and warnings before running) and also by offering suggestions and autocomplete the code. This makes it easy for developers who do not know much about the working of the programming/designing languages involved in the development of the software.

4. Communication hindrances.
Cloud computing provides messaging platforms (SaaS) for the development team and users to have frequent conversations and clarifications with each other. This is a major component, catalyzing the development cycle and facilitating the collection of user stories, inter-team discussions, feedback, prototype delivery and much more. This will both cut off the cost of communication and increase the time-efficiency of communication with a huge margin.

5. Testing, prototyping, and integration
Testing and prototyping can be more than complemented by cloud computing by the TaaS service. It provides a (normally outsourced) testing atmosphere with simulated use cases, efficient and fast testing platforms and moreover, it is virtual, which makes it a lot cheaper than conventional testing protocols practiced. Prototype creation gets a lot easier with the developing platform supporting emulations.

Testing as a Service(TaaS) features

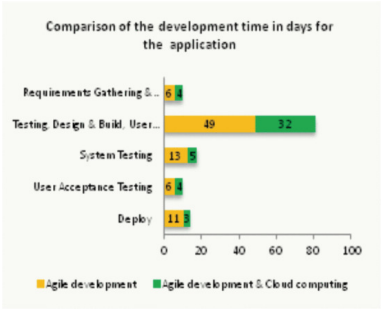
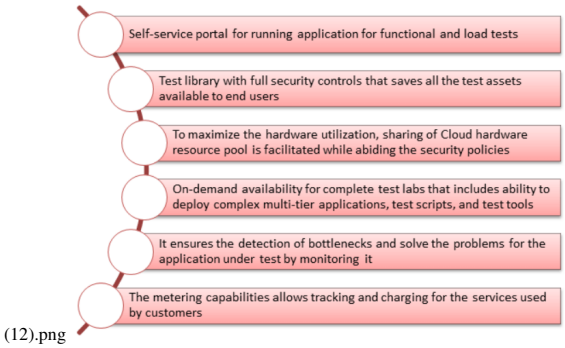


Fig. 14. Comparison of software development with and without cloud services

Moreover, when testing and development is done on cloud computing platforms, dynamic and automatic documentation takes place. This will reduce the amount of effort to be put by the developers during development. As it also facilitates communication, unit testing, pair programming also become easy and swift.
Overall, cloud computing facilitates the XP method by making it fast and reliable for software development.
Here is a comparison of the development time of an average project with and without cloud computing:



(12).png

Fig. 15. Fig. TaaS features

That diagram summarises the impact cloud computing can have on XP. Shorter durations of all the phases show that the project was successful in exploiting all the resources cloud computing can provide. The maximum percentage of improvement was evident in Deployment. Hence, the approach proves to give good positive change in the methods of XP. This success will in turn tackle and solve problems like coordination between team members, lack of interest in developers, stunted interaction between users and developers, etc..
It can be conclusively said that cloud computing is an effective supplement for the agile XP method, affecting it. The future of Information Technology will be cloud computing or software applications that are delivered and used through a web browser. By adopting cloud computing, XP methods deliver superior quality of software in lesser development time.

IV. **Discussions**

The approach here has been proposed and implemented only in one such scenario which seems to be the ideal one. This new approach must get a go-ahead in many projects by introducing it from the low risk to high-risk levels. Due to not having sufficient time, we couldnt try out this process on a large scale which would then expose the real problems and real-life scenarios. The strengths of our approach have already been discussed which have yielded wonderful results. But will this happen in all real-life projects is still something to ponder over? Due to a limited number of case studies, nothing can be said on how well this approach can be applied or followed in any given case study. The weaknesses of trying out this approach are still unknown unless deployed to a vast scale. Otherwise, this is a proven method which has quite a lot of tools like cloud computing that have been already deployed all across the world and are just enhancing the users experience.

The references had some wonderful ideas related to software testing using TaaS and making testing a part of all the sprints. The agile approach is used to have a separate testing phase after the development phase. By the time testing phase would come, there could be a lot of bugs that are to be fixed before delivering the product for user acceptance testing. Therefore, if testing is to be included right from the beginning through all the sprints up to the explicit testing phase, quite a lot of bugs if not all would have been already resolved and it will not take much time during the testing phase to fix any bug. The approach which I especially liked in the paper I referred was that if the dependent modules have some issues to be resolved, then it is best to simplify our work by resolving the issue when it occurs and not wait till the testing phase arrives.

Many other papers expose many the loopholes of the SCRUM and XP framework which give room to learn that there are defects which need to be removed and propose something efficient in place of it. This paved the way for the proposed work. The idea of automated testing and cloud environment can save a lot of time, energy and labor which can make the managers role a lot simpler. Having a small team with efficient developers with right knowledge required to the tasks they are assigned is another good idea which I picked from a work that I went through while referring to one of the papers.

In this paper, the methods are analysed based on the applicability of cloud computing in providing solutions to the problems identified in it. The analysis also happened in an application based context. Moreover, the involved research was on specific characteristic backlogs in the methods. The problems can be approached from a different context like including frameworks from other SDLC models, etc..

V. References

Raj, G., Yadav, K. and Jaiswal, A., *2015, February. Emphasis on testing assimilation using cloud computing for improvised agile SCRUM framework. In 2015 International Conference on Futuristic Trends on Computational Analysis and Knowledge Management (ABLAZE)* (pp. 219-225). IEEE.

Boehm, B. and Turner, R., 2005. Management challenges to implementing agile processes in traditional development organizations. *IEEE software*, 22(5), pp.30-39.

Kalem, S., Donko, D. and Boskovic, D., 2013, May. *Agile methods for cloud computing. In 2013 36th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)* (pp. 1079-1083). IEEE.

Livermore, J.A., 2007, March. Factors that impact implementing an agile software development methodology. In *Proceedings 2007 IEEE SoutheastCon* (pp. 82-86). IEEE.

Miller, G.G., 2001, July. The characteristics of agile software processes. In *tools* (p. 0385). IEEE.

B. Boehm, "Anchoring the Software Process", *IEEE Software*, vol. 13, no. 4, pp. 73-82, 1996.

Fitriani, W.R., Rahayu, P. and Sensuse, D.I., 2016, October. Challenges in agile software development: A systematic literature review. In *2016 International Conference on Advanced Computer Science and Information Systems (ICACSIS) (pp. 155-164)*. IEEE.

Dikert, K., Paasivaara, M. and Lassenius, C., 2016. Challenges and success factors for large-scale agile transformations: A systematic literature review. *Journal of Systems and Software, 119,* pp.87-108.

Holmstrm, H., Fitzgerald, B., gerfalk, P.J. and Conchir, E.., 2006. Agile practices reduce distance in global software development. *Information systems management, 23(3)*, pp.7-18.

Schatz, B. and Abdelshafi, I., 2005. Primavera gets agile: a successful transition to agile development. *IEEE software*, 22(3), pp.36-42.

Newkirk, J., 2002, May. Introduction to agile processes and extreme programming. In *Proceedings of the 24th International Conference on Software Engineering. ICSE 2002*(pp. 695-696). IEEE.

Kalem, S., Donko, D. and Boskovic, D., *2013, May. Agile methods for cloud computing. In 2013 36th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)* (pp. 1079-1083). IEEE..

Al-Zoabi, Z., 2008, April. Introducing discipline to XP: Applying PRINCE2 on XP projects. In *2008 3rd International Conference on Information and Communication Technologies: From Theory to Applications* (pp. 1-7). IEEE.

Ordoez, H., Escobar, A., Velandia, L., Cobos, C., Ordoez, A. and Corrales, J.C., 2015. Eliciting Requirements in Extreme Programming (XP) Through Business Process Models.

Hashmi, S.I. and Baik, J., 2007, August. Software quality assurance in XP and spiral-A comparative study. In *2007 International Conference on Computational Science and its Applications (ICCSA 2007)* (pp. 367-374). IEEE.

R. Shriver, "Agile Cloud Development", The Virtualization Practice, LLC, 4. June 2012 http://www.virtualizationpractice.com/agile-cloud-development the-future-of-software-16226/.