International Journal of Engineering Science Invention (IJESI)
ISSN (Online): 2319 – 6734, ISSN (Print): 2319 – 6726
www.ijesi.org ||Volume 7 Issue 10 Ver IV || Oct 2018 || PP 17-23

Collaborative Approach In Distributed Software Systems

Choudhary Ravi Singh[1], Gaurav Singh[2]
[1]*Department of Computer Science & Engg., Rakshpal Bahadur College of Engg. & Tech., Bareilly, India*
[2]*Department of Computer Science & Engg., Rakshpal Bahadur College of Engg. & Tech., Bareilly, India*
Corresponding Author: Choudhary Ravi Singh

Abstract: *Distributed and Collaborative software development, today is highly researched concepts. In distributed software development several software development teams work over the same software project across different geographical locations and environments. Over the same software project, the work is done from different locations by different teams so it is necessary to develop incorporation and collaboration among them for the success of a software project. In distributed software development the software is developed in modules so shared understanding of various modules are necessary for all the different team members. Now the most important fact is how to enable communication, cooperation, and sharing of resources, understanding of modules, among various team members. Collaborative software development is an approach designed to fulfill all above-said needs so that various team members involved in a common task can reach their goals. This paper focuses on the pros of distributed and collaborative software systems as well as challenges and issues faced during the development of distributed and collaborative software systems.*

Keywords- *Agile Practices, Collaborative Systems, Collaboration tools, Distributed Systems, Web based Environment*

Date of Submission: 17-10-2018 Date of acceptance: 03-11-2018

I. INTRODUCTION

Distributed software development means working on a large computer-based software system from different locations with various software developer teams where different processors or fully different systems are used to process the data at several locations which are geographically different. The computer languages, operating systems, and hardware environment for these systems can be different. Now we come to collaborate and it becomes the full responsibility of the integrating teams, which will merge different software modules to develop a software product that is independent of any specific environment. We have three different types of distributed systems and figure 1 clearly represents all three types.

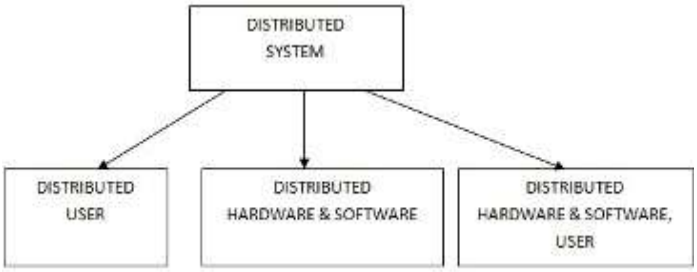

Figure 1: Distributed system types.

When we work on a large software project, it requires several people for fast and good results. The collaborative approach generally is used to eliminate the restrictions caused due to human errors. What everyone is working on, it is very tough to keep this track while working collaboratively and we know human languages can also be ambiguous. As an outcome, it heads to errors, duplication of the work, therefore the most important is we must have a single architecture and design.

II. PROS OF DISTRIBUTED SYSTEMS

Many researchers have found that distributed system for development of a software project has the following advantages. Maximum of the computer systems in the present world are distributed so it is necessary to consider the following features.

2.1 Scalable

Scalability plays an important role in the success of a software product. Distributed systems support scalability it means we easily can add new resources to the existing resources whenever we need. Resources like memory, computers can be added to face up with the changing requirement of our existing system as described by the client. We know scalability highly depends on network capacity and sometimes it can be inadequate.

2.2 Sharing of Resources

Resources available across the entire network of various computers are shared from different locations. All the hardware resources like disks, printers, memory, and software resources like files etc. can be shared.

2.3 Widely Accepted

Distributed systems are open systems because they are developed using some standard set of protocols. It means software and hardware resources developed by many vendors for our distributed system we can easily use them.

2.4 Fault Tolerable

Some amount of failure can be tolerated by distributed systems because they are not centralized. This is the most important feature of distributed software systems and is not generally supported by normal software systems. Distributed software systems can tolerate some amount of failure both software and hardware because it has several computers available it means data store on one machine can be easily replicated onto another machine to provide regular services to the user. But if we have network degradation the system becomes unable to provide its services.

2.5 Concurrent

Concurrency means the ability to perform several jobs on various processors at the same time. In a distributed software system at the same time, more than one module or process may be running on different computers on the same network.

III. CONS OF DISTRIBUTED SYSTEMS

3.1 Service Quality

Availability, performance, and reliability all are important factors to ensure good quality of services. The services that will be delivered to the end users mostly depend on workload processor that will execute them as well as types of the process used for execution.

3.2 Safety and Security

We know that in distributed system data can be accessed from different locations via different computers. Here each computer is an independent system and it becomes tough to provide security to each independent system than a single centralized system. Data that is transferred from one system to other is very sensitive. Huge traffic is one major problem that leads us to compromise with privacy. So due to many problems, it becomes very tough to take proper care of data integrity and provide protection to the system from unwanted threads like Eavesdropping, Byzantine attack, performance degradation.

3.3 Highly Complex

If we come to testing and building distributed systems are highly complex than centralized systems. It is tough to measure the performance of each individual module in distributed systems. Its performance highly depends on network bandwidth and speed of execution of the processors that are part of the distributed network. Here one system sends resources to another system so resource handling and transferring them from one to another system is one another issue that directly affects network performance. So it is necessary to have a high amount of bandwidth and resources to maintain distributed system performance.

3.4 Transparency

Transparency of services provided by the distributed system is an important factor. It highly depends on how a system manages the distribution of workload and how well the system manages to hide inner complexity.

3.5 Testing

Performing testing of a single system is an easy task, but in a distributed system we have the number of distributed modules so to test a complete distributed system becomes more challenging. First, we have to find

out which module will be tested first and which after it. So it is necessary first we should apply unit testing of each individual module and then integrated testing, system testing of the entire system.

3.6 Task Allocation

In the distributed system we have many processors, therefore allocating tasks to different processors for even distribution of the workload is an important thing because the wrong task allocation may lead to uneven workload and crashing of the system. Due to the uneven workload performance of the entire system may be degraded.

IV. ARCHITECTURE OF DISTRIBUTED SYSTEMS

To overcome the designing issues which misguide the underlying software and hardware components in this paper, we decide to discuss two types of distributed system architectures.

4.1 Client-Server Architecture

In this distributed system architecture users of the distributed system are considered as clients and distributed system, itself is considered as a service or server so the name is client server. In this model, the client generally knows about all the services which are available by different systems. Here client does not need to worry about the other clients who are present and availing the same services over the same network. In this network model, we can view two different processes the client and the server. The one to one mapping between the server processes and the client is not always necessary. Figure 2 clearly represents the client-server model.

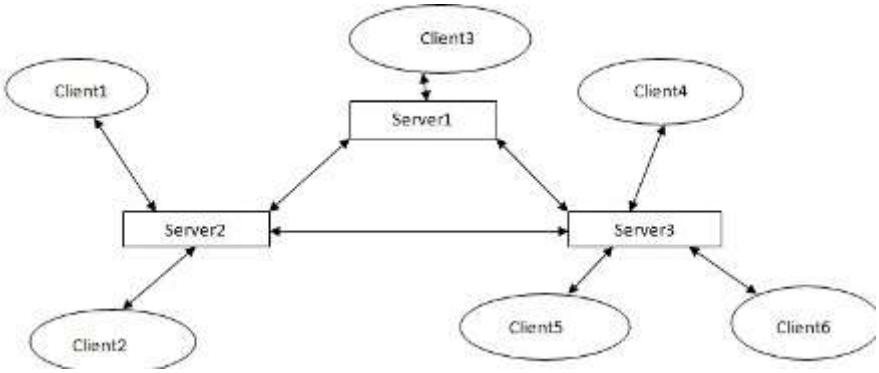

Figure 2: Client – Server Architecture.

A logic structure of the software being developed using client-server architecture is necessary. Usually, this structure consists of 3 layers.

(1) Presentation Layer

It is the first layer of the logical structure of the software being developed using client-server architecture and its job is to present information and data to the users and allow the users to interact with the system.

(2) Application Layer

Logic involved in the software is deployed using this layer.

(3) Data Management Layer

This layer aim is to perform various database operations. This layer also deals with the management of data.

We know that two-tier architecture that has both client and server is classified into two different categories.

4.1.1 Thin Client

In thin client model all the operations related to data and processing of different applications are done by the server (distributed system) and the user (client) for work only deals with the presentation layer.

4.1.2 Thick Client

In this model responsibility of the system is to manage the data and client performs all the other remaining work like working out the logic and interacting with the software.

Two-tier architecture is a simple form of client-server architecture and depending on the complexity of application further, we can extend our system to three-tier and multitier framework.

4.2 Distributed Object Architecture

Like client-server architecture in this architecture model, we don't have any difference between the client and the server. The system is considered as a collection of objects that interacts with the users. We know that in client-server architecture client must be aware of the services provided by different systems because they don't need to contact directly to the servers for it. This approach is not good if we come with flexibility and scalability. So the right approach is to make zero difference between client and server and consider each of them as objects. In this model, objects are distributed over the entire network and if objects want to communicate with each other they use middleware. This works as request broker and acts as a medium of communication between objects and also a medium to add or remove objects.

Now in this architecture model, we have no difference between thin clients and thick clients that were in case of client-server architecture so this model provides a high level of scalability and flexibility.

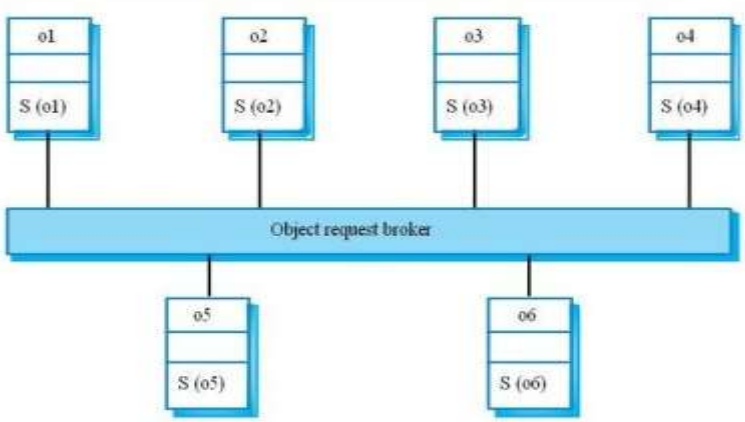

Figure 3: Distributed Objects Architecture.

V. AGILE PRINCIPLES OF DISTRIBUTED SOFTWARE DEVELOPMENTS

Nowadays distributed systems are the needs of software development. When the requirements and scope of a project increase it becomes necessary to scale the project and distributed system comes into the existence. Agile concept splits the tasks into small phases of work and supports further reshaping and adaptation of ideas. Agile practices handle flexibility during the entire period of the life cycle of the undertaking and help to improve open collaborative efforts. If we adjust Agile practices in a circulated environment, these can help in handling difficulties of social contradictorily and absence of trust. Here we discuss the use of Agile practices in the area of distributed software development. Some Agile practices and policies that require to be implemented within your distributed system are described as follows:

(1) Continuous availability of the prototype model of working software is necessary to constantly measure its progress.

(2) Fast and uninterrupted delivery of useful applications is necessary for the success of a project.

(3) Requirements must be flexible.

(4) There must be mutual understanding, cooperation among developers, business heads and users and managers.

(5) Continuous handling of technical problems.

(6) Proper selection and involvement of project members those have faith in each other.

(7) Face to face communication is the most trustable way of communication.

(8) All plans must be well organized.

(9) Management of proxy servers over the distributed system is necessary.

(10) Remote collaboration is required to enable shared file storage by web services.

(11) Everything must be clear and organize in an easy way.

(12) Adaptive to frequently changing situations.

VI. NEEDS OF COLLABORATIVE SOFTWARE ENGINEERING

Researchers' have found collaborative software engineering fulfills the following needs during distributed software development.

6.1 Strengthen Team Relationship

One sure way to build an effective working relationship among employees is to have them work together to complete tasks and projects. The collaborative programs are well designed to do that and making it easy for teams to work together as a concerned unit with a single purpose. This makes teams confident and comfortable in working with one another when it comes to completing common goals.

6.2 Enhance Project Management

To keep a handle on a project, a team and its members and their progress is not a simple job. The communication and the coordination among various teams and their members are most important if we want a project completed. The Collaboration provides a guarantee that you get the optimum performance of your team. It provides lines of communication and makes sure that everything is on the right track to avoid further mistakes.

6.3 Save Time

Time plays an import factor during software development. The more you save time the more you save on unnecessary expenses which are not better for your organization. Time is gold and in business, it is identical to money. The Collaboration among employees makes possible the fast completion of goals that would not be achievable in any other way. The Collaboration software can help in cutting costs on projects by saving time.

6.4 Improve Organisation

The major issue for a business is how to keep the things and tasks on the right track?
The collaborative program helps in maintaining the order and managing the stages in the workflow via open communication channels and coordination among concerned individuals and teams.

6.5 Define Scope & Availability of Project

Software developers with clients and fund providers should discuss what they want the product to do and what their requirements for the software product are. Software project will not be processed further without such discussions.

6.6 Single Design & Architectural Model

Software designers must make it sure that everyone is agreeing upon a single architectural model.

6.7 Management of Dependencies

It is the phase of dividing the process into smaller sub-processes and each has an assigned date of completion. Sub-processes or modules are further assigned priorities and their dependencies are clearly defined.

6.8 Error Identification & Solving

Errors can occur in any software product because of mistakes and ambiguities. Many methods are used to discover them and solve them. In collaborative approach different surveys, investigations are performed to find out errors and ambiguities. In collaborative software development, various individuals are united so that their different points of view can find out errors.

6.9 Recording Memory of Employees

In a collaborative environment, individuals can join and go. This approach also records that individual knew so that present employee can get benefit from this information now and later.

VII. TYPES OF COLLABORATION SOFTWARE

In this paper, we discuss three types of collaborative software.

7.1 Communication

This software helps in exchange of communication between various groups. Many researchers have shown that file sharing is the most common and useful feature in collaboration software. Tools which are used for communication are systems and applications for email hosting, file sharing, project management or a website that can be readily accessed. Example tools are email, voice mail, video calls, etc.

7.2 Conferencing

Here we have groupware software tools which make the real-time discussion among various teams through a virtual meeting room and a mediator who supervises sharing of the information.
Conferencing allows real-time collaboration among project members. Example tools are video conferencing, social media groups chat, etc.

7.3 Coordination

For complex independent tasks to obtain a common goal coordination software solutions are used. Systems applied here are used for time management, online proofing, and project management so that team members are aware of deadlines and are properly coordinated and to check the status of the project. Example tools are time trackers, calendars, client portals, and status updates, etc.

VIII. COLLABORATIVE TOOLS

In this paper, we discuss four types of tools for collaborative software engineering.

8.1 Model Collaboration

Software engineering consists of many phases like a feasibility study, requirement gathering, and analysis, designing, coding, testing, and maintenance. Model-based collaboration enables us to make a separate, unique model for all the different phases. It allows representing the work done by the team members in a more user-friendly way.For the requirement phase, we have some collaboration tools like Requisite-Pro, E-requirements, and Raven-Flow, etc. UML based approach is followed by designing tools some examples are Gliffy and Argo-UML. Testing tools have various features like recording bugs experienced by them. One other approach to perform testing is to allow different users to test the different functionalities of the software and record their comments and feedbacks. To create traceability links we have a tool called Xlinkit.

8.2 Awareness Tools

Every member of the software developing team has assigned a unique workspace where he/she perform his/her work related to a specific task or module. The workspace allows individuals to work in a more convenient way because it is independent of the other members. These tools spread awareness regarding the work done by the different members in the working scenarios and allow them to work in cooperation and coordination without any conflict.

8.3 Process-Based Tools

These tools provide pre-planned structure for various software development phases that will be performed during huge software developments. It explains the sequence, assigns priorities to them, and defines roles of different engineers that are part of software development. It also monitors their progress and reduces the time for integrity, coordination, etc. Example tool Endeavours.

8.4 Infrastructure Collaboration

These tools are to coordinate the work of software tools through data integration, awareness among various tool activities and creating repositories for control integration, etc.

IX. LATEST COLLABORATION SOFTWARE TRENDS

In this paper, we only listed four different types of latest software collaboration trends. Real-time collaboration, People-centric collaboration, Handling Big Data, Collaboration Bots: they are used to automate the messaging and chat software but may soon find the ways in collaboration software to automate tasks and run areas in project workflows. It will make the collaboration process less time-consuming.

X. INTEGRATING DESTOP & WEB BASED ENVIRONMENT

In researching, we have found a collaborative development can benefit largely by moving applications to the web. Web applications support large amounts of user interaction, integration by use of high-level programming languages such as AJAX, JavaScript.

Web-based software development tools provide various designing formats and APIs, but they cannot completely replace desktop-based IDEs. Errors tracking and test cases are better for web-based software but debugging, a compilation of source code is more secure on the desktop-based IDEs. That's why there is a necessity to make an open working environment wherein both the services can interact seamlessly with each other and use the best features of each of them to develop better quality software.

XI. CONCLUSION

In present scenarios, the world is moving rapidly towards the use of Big-Data. Cloud computing, Grid computing, and Cluster computing each of them requires the highest computational powers. The huge advantages each of them offers but it is not fully safe and performance oriented. The issues and challenges present here must be handled for an error-free high-performance approach. The highest degree of advantages is offered using Agile practices by conveying early, improving communication among teams and their members and allowing the business to react quickly by changing the product. Attempting to expand a reformed venture in an Agile way is not so simple and will add bargains yet the interest points are awfully numerous. Group joint efforts can lessen perfunctory difficulties characteristic in the cooperation and coordination of a massive group of individuals and can better influence the remarkable abilities and capacities of every group member and also of every group part. Presence of new work around there will add the profitability and the nature of software undertaking.

REFERENCES

[1]. Sudipto Ghosh, Aditya P. Mathur, Software Engineering Research Centre, West Lafayette, March (1999), Issues in Testing distributed component -based systems.
[2]. Hiroshi Tamura, Futoshi Tasaki, Masakazu Sengoku and Shoji Shinoda Niigata Institute of Technology, Japan , Faculty of Engineering, Niigata University, Japan, IEEE 2005, Scheduling Problems for a Class of Parallel Distributed Systems.
[3]. Adopting Agile In Distributed Development, GlobalSoftware Engineering, (2008), ICGSE (2008) , page(s): 217-221, Sureshchandra, Kalpana Shrinivasavadhani.
[4]. E. Koeppen, H. Rhinow, and Meinel, Prototypes as Boundary Objects in Innovation Processes in Design Research Society (2012), Bangkok. Conference Proceedings, vol. 4. DRS, (2012), pp. 1581–1590.
[5]. S .McConnell, Lifecycle Planning In Rapid Development Taming Wild-Software Schedules Redmond, WA-Microsoft Press, (1996).
[6]. C. Carver, First International Workshop on Software-Engineering for Computational Science & Engg., Computing in Science & Engineering, vol. 11, no. 2, pp. 7–11, (2009).

Choudhary Ravi Singh "Collaborative Approach In Distributed Software Systems "International Journal of Engineering Science Invention (IJESI), vol. 07, no. 10, 2018, pp 17-23