International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 05 Issue: 09 | Sep 2018 www.irjet.net p-ISSN: 2395-0072

Distributed & Collaborative Software Engineering

Rutvik ManishKumar Patel

Graduate Student, Department of Computer Science, California State University Sacramento, California, USA

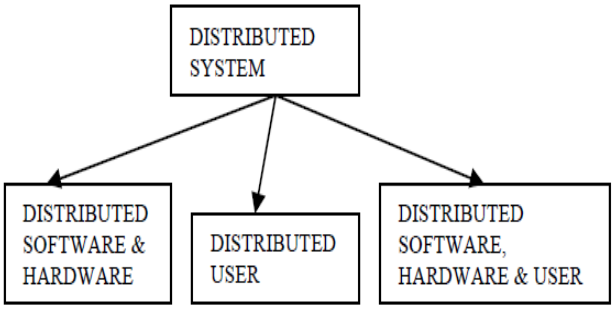
Abstract - *This paper introduces us to two of the most important and highly researched topics of the modern era-distributed and collaborative software development. Distributed software development deals with the development of large software projects, developed across different locations and environments having several teams working on some task of it. Whereas while working on large software project distributed over multiple locations, it is important to develop integration and co-operation among the various teams, developing shared understanding regarding various modules. This is how collaborative and distributive development works in order to develop large software projects at low cost by utilizing modern technologies like multi-programming and multi-processing. This paper discusses the benefits, issues and challenges faced by distributed and collaborative software systems.*

Key Words: Distributed System, Collaborative development

1. INTRODUCTION

Distributed software engineering means working on a distributed system that is a large computer based system where data is operated and processed over different processors or completely different systems at multiple locations. These systems may work with different computer languages, operating systems and hardware environment. It is the responsibility of the integrating team to collaborate various modules to develop a product independent of any particular environment. Distributed systems are of three types. Figure-1 clearly depicts the 3 types of systems.

1. Distributed software and hardware
2. Distributed User
3. Distributed User and Software, Hardware



Collaborative techniques generally are used to remove the limitations caused due to human errors. While working on a huge project, we require several people for fast and better outcomes. However while working collaboratively, we are not always able to keep track of what everyone is working on and also the human language is quite ambiguous. As a result, it leads to errors, duplication of work. Hence it is important to have a single architecture and design.

2. BENEFITS OF DISTRIBUTED SYSTEMS

According to researchers, using a distributed system for development of a software project has the following advantages. Since most of the computer systems in the modern world are distributed, it is important to consider the following aspects.

1. Sharing of Resources:-

All the various kinds of resources located across the entire network of computers are shared with each other. This includes all the hardware and software resources that are disks, memory, printers, files, etc.

2. Widely Accepted:-

Distributed systems are developed using some standard set of protocols which means they are

open systems. Hence we can use software and other resources developed by multiple vendors for our distributed system.

3. Scalable:-

Distributed systems are scalable. It means that we can add new resources as per our needs. Resources like computers, memory can be added to cope up with the changing requirements of our system as specified by the client. However scalability highly depends on the network capacity which may not be adequate sometimes. Also various other issues are faced wile scaling of distributed systems but that is another topic.

4. Concurrent:-

Concurrent operation of various processors is possible at the same time. It means that more than one module or process may be running on different computers but on the same network at the same time. The processes may communicate with each other.

5. Fault Tolerable:-

It is an important aspect of distributed software that is generally not supported by normal software or systems. Distributed software can tolerate some amount of failures- both hardware and software since it has several computers available which means that data can be easily replicated from one machine to other to provide regular services to the user. However when there is a network degradation, then the system is unable to provide service.

As a result of these advantages, distributed systems have largely replaced the normal single processor systems.

3. DISADVANTAGES OF DISTRIBUTED SYSTEM

1. Highly Complex:-

In terms of organization and testing, distributed systems are more complex than the single processor systems. Hence it becomes difficult to measure performance of such systems. In Single processor systems, performance depends on speed of a single processor while for distributed systems it depends on the bandwidth of the network and speed of execution of all the processors included in the network.
Also resource handling and transferring resources from one system to another is another issue which directly affects the performance of the system. Hence high amount of bandwidth and resources are required to maintain the performance of the system.

2. Security and Safety:-

The distributed system consists of data and information that can be accessed from more than one computer. Hence we can say that it is an interdependent system. To apply security policies to interdependent system is a tough task and has to be handled carefully. Also the data can be highly sensitive. Increased traffic is another problem which leads to compromise of privacy. Hence due to various problems it becomes very difficult to maintain the integrity of the system and prevent the system from threats such as eavesdropping, denial of service attack, degraded performance, etc.

3. Management:-

Distributed systems are heterogeneous in nature. End systems differ from each other as each of them have different capacities. Resource management is always and will be a issue for distributed systems since they are at different locations. Routing management issues will occur at network layer. Synchronization of thousands of computers is difficult to manage and handle. Current techniques like semaphores, monitors, process calls, message passing, etc. are not very handy.
In case of failure it becomes very difficult to recognize and handle it as different machines operate on different operating systems. It may cause unexpected performance issues in other machines.

4. Quality of Service:-

The Quality of Service delivered to the end users will highly depend on the type of the process executed and the workload processor which executes it. Performance, reliability and availability are important points to be considered to ensure good quality of service.

5. Transparent:-

The main goal of a distributed system is to ensure transparency of the services provided by the system. The transparency depends on how well the system manages to hide the inner complexity and distribution of workload.

6. Testing:-

To perform testing of an entire distributed system is a very challenging task. The first issue that needs to be addressed is that which module should be tested first and which after. We need to be very precise in forming the sequence of modules to be tested otherwise loops may start forming. Also it is very important to test each component separately as well as the entire system after that. There may occur some redundant testing as well.

7. Task Allocation:-

Allocating tasks to different processors to evenly distribute the workload is very important since unreliable task allocation may produce uneven workload, crashing of system or downgraded performance.

4. DISTRIBUTED SYSTEM ARCHITECTURES

In this paper, we discuss two types of architectures for distributed system to overcome the designing issues that prevail within the underlying hardware and software components. The two types of architectures discussed here are:-

1. Client-Server Architecture:-

Here the distributed system is considered as a service and the users of the distributed system are considered as clients, hence the name client-server.

In this type of model the client is usually aware of all the services that are available by different systems. Although they do not need to worry about the other clients that are existing and using the same services. Hence we can view two different processes here-the client and the server. 1:1 mapping between the server process running and the processors is not necessary.

Figure-2 depicts a model based on the client server architecture.

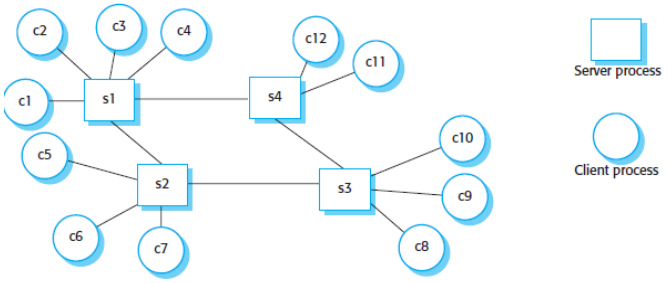


Figure-2: Client-Server Architecture

It is very important aspect to develop a logical structure of the software being developed using client-server architecture. Generally we partition the structure into 3 layers to follow a designing approach.

Presentation:-The first layer is the presentation layer and it will present information and data to the user and allow the user to interact with the system.

Application Processing:-It allows deployment the logic involved in the software.

Data Management:-This layer is concerned with management and performing various database operations.

The two-tier architecture that consists of clients and servers can be classified in two different types of models

Thin-Client:-In this type of model, all the operations related to data and processing of applications is done by the server that is the distributed system, while the user only works at the layer of presentation.

Thick-Client:-Here the system will only manage the data. The client does all the other work like working out the logic and interacting with the software.

Example of this kind of a model is ATM machine. Two-tier architecture is a simple form of architecture. Depending on the complexity of application, we can extend our system to three tier as well as multi-tier framework

2. Distributed Object Architecture:-

Unlike the previous one, here we do not have any distinction between the system and its users. The system is treated as an object or a set of objects that interacts with the user. In client-server model, clients have to be aware about the services that are provided by various systems as they need to contact the servers directly for that. This kind of approach is not very scalable and flexible. A better approach is to keep no distinction between the client and server and treat each of them as objects. Objects are distributed across the entire network. If they want to communicate, they need to use middleware. It is also known as request broker and acts as a medium of communication between objects and also a medium to remove or add objects.

No distinction is required here between thin and thick clients as in case of client-server approach and it provides a very scalable and flexible approach.

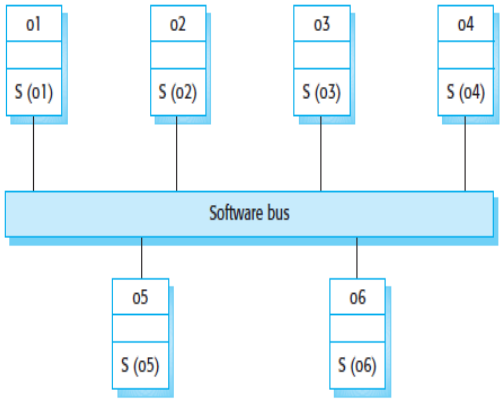


Figure-3: Distributed Object Architecture

5. AGILE PRINCIPLES FOR DISTRIBUTED SOFTWARE FEVELOPEMENT

Distributed systems are inevitable in the field of software development. As the scope and requirements of a project increases, it becomes necessary to scale up the project and hence distributed systems come into the picture. Agile methods divide tasks into short phases of work and support frequent reshaping and adaptation of plans. Agile practices advance improvement iterations, open collaborative effort, also, handle flexibility for the entire duration of the life cycle of the undertaking.

Adjusting these practices in a circulated environment can help appropriated advancement tackle the difficulties of social contradictorily, initiative battle and absence of trust. This section discusses the use of agile practises in the field of distributed software development.

Some of the standard agile policies and practises that need to be implemented within your system are as following.

1. Fast and uninterrupted delivery of useful application or software.

2. Progress is constantly measured by continuous availability of the prototype model of working software.

3. Flexibility in requirements.

4. Co-operation and mutual understanding between developers, managers, business heads and consumers.

5. Proper selection and involvement of project members who can always be relied upon.

6. Technical problems should be continuously handled.

7. Simple.

8. Face-to-Face conversation is the most relied upon form of communication.

9. Well-organized.

10. Adaptive to the varying situations.

11. Remote Collaboration to enable shared file storage using web services or messenger services.

12. Manage proxy servers over distributed systems.

6. GOALS OF COLLABORATIVE SOFTWARE ENGINEERING

Various goals have been set throughout the entire life-cycle of software development.

Define scope and ability of project:-

Developers along with the clients and other fund providers or stakeholders should discuss what they really want the product to do and their requirements from the product. Without such discussions, the project may not proceed forward as planned.

Single design and architecture Model:-

Software designers must make sure that everyone has agreed upon and can relate to a single architecture model.

Management of Dependencies:-

It includes the phase where the process is sub-divided among various modules and each module is assigned a date of formation. Also the modules are prioritized and their dependencies are stated as per their needs and availability. Controlling and assessing various modules is included here.

Error Identification and Solving:-

Blunders and ambiguities are conceivable in all software products, and numerous methodologies are created to discover and record their presence. The collaborative procedures available are investigations and surveys, where various individuals are united so that their numerous points of view can distinguish errors, and their inquiries can surface ambiguities. In testing, where one gathering makes tests to reveal mistakes in programming created by people is a collaborative mistake gathering method. Clients also work together in the ID of mistakes, whether in express beta testing programs, or through typical use, when they submit bug reports. Bug frameworks license specialists to save issues, and deal with the procedure of determining them.

Recording memory of employees:-

In a long term collaborative venture, individuals may come and go. Collaboration also includes recording what those individuals knew, so that current employees can benefit from this information now, and later on.

Change logs, are a type of storage memory used in collaborative development. Process models additionally save memory, portraying practices for the most effective method to create applications.

With the upcoming of new technologies, software developers are not holding back and have started to adapt new communication mediums for collaborative project.

7. COLLABORATIVE TOOLS

Four different types of tools are available for collaboration in software engineering.

Model Collaboration:-

Software engineering consists of various phases like the requirements, design, architecture, test cases, end product, etc. Model based collaboration allows us to create a separate and unique model for all the different phases involved in the process. It allows to represent the work done in a more user friendly way.

Collaborative tools available for the requirement phase are e-Requirements, Requisite-Pro, Raven-Flow, etc. Designing tools follow a UML based approach and Argo-UML, Gliffy are a few examples. Testing involves various features like granting access to the prototype of model to the users and record bugs experienced by them. One such other approach is to allow multiple user to test the different functionalities of the software and record their feedback and comments. Tool for the same is bug tracking tool. A tool called XLinkit allows to create traceability links among various modules.

Process Based:-

Such tools allows to have a pre-organized structure for various phases to be carried out within a large software project. It defines the sequence, prioritizes them, assigns different roles to different engineers involved in the process, monitors their progress and hence reduces time for co-ordination, integrity, etc. Examples are Marvel, Endeavours, etc.

Awareness Tools:-

Every member of the software developing team is assigned his own workspace where he carries out his work related to a specific module. The workspace allows developer to work in a more fruitful way as it is independent of other members. However it does not allow engineers to work in a parallel way with coordination. Augur is an awareness based tool that spreads awareness regarding the work done by different engineers in the working environment and hence allows them to work in coordination without any conflicts.

Infrastructure Collaboration:-

The goal of various infrastructure technology based tools is to coordinate the work of software tools by creating repositories for control integration, data integration, awareness among various tool activities, etc.

8. INTEGRATING DESKTOP & WEB BASED ENVIRONMENTS

Researchers have shown that collaborative development can benefit largely by moving applications to the web. Web applications support large amount of integration, user interaction by use of rich programming languages such as JavaScript, AJAX.

The support of a desktop like user friendly interface in the newly arrived Web 2.0 and integration of data among various sites has started a new paradigm in software engineering. Web based software development tools provide various APIs and designing formats but they cannot fully replace desktop based IDEs. However bug tracking and test cases are better on web based but compiling, debugging of software is yet more secure on the desktop based IDEs. Hence we need to create an open working environment where in both the services can interact with each other seamlessly and use best features of each of them to develop better software.

9. CONCLUSION

Distributed software development is inevitable as the world is moving towards use of big data, cloud computing which requires high computational powers. As much as advantages it offers, it is not yet completely safe and performance oriented. There are issues and challenges which need to be handled for an error free, high performance approach. Agile practises offer a high degree of advantage by conveying early, improving

communication and permitting the business to react rapidly by changing the product. Attempting to disseminate an improvement venture in an agile way is not very simple and will include bargains however the points of interest are awfully numerous. To the degree that forwards in software engineering group joint effort can lessen inadvertent troubles characteristic in the coordination of vast groups of individuals, and can better influence the exceptional abilities and capacities of every group part, new work around there will enhance the profitability and nature of software undertakings.

REFERENCES

[1] J. C. Carver, "First International Workshop on Software Engineering for Computational Science & Engineering," Computing in Science & Engineering, vol. 11, no. 2, pp. 7–11, 2009.

[2] H. Rhinow, E. Koeppen, and C. Meinel, "Prototypes as Boundary Objects in Innovation Processes," in Design Research Society 2012:Bangkok. Conference Proceedings, vol. 4. DRS, 2012, pp. 1581–1590.

[3] Issues in Testing distributed component -based systems, Sudipto Ghosh,Aditya P. Mathur, Software Engineering Research Centre,West Lafayette, March 1999.

[4]Scheduling Problems for a Class of Parallel Distributed Systems,Hiroshi Tamura, Futoshi Tasaki, Masakazu Sengoku and Shoji ShinodaNiigata Institute of Technology, Japan , Faculty of Engineering, Niigata University, Japan, IEEE 2005.

[5] Sureshchandra, Kalpana Shrinivasavadhani, Jagadish ," Adopting Agile In Distributed Development", Global Software Engineering, 2008. ICGSE 2008 , page(s): 217-221

[6] S. McConnell, "Lifecycle Planning," in *Rapid Development:Taming Wild Software Schedules* Redmond, WA: Microsoft Press, 1996.

[7] R. Grinter, "Systems Architecture: Product Designing and Social Engineering," in *ACM Conference on Work Activities Coordination and Collaboration (WACC'99)*, San Francisco, California,1999,