www.ijecs.in
International Journal of Engineering and Computer Science
Volume02Issue 01, Jan2013, PageNo.183,-187 ,ISSN:2319-7242

Enhancing Distributed Software Development with Collaborative Technology

Bharat C. Patel[1]
patelbharat99@yahoo.co.in
[1]Smt. Tanuben & Dr. Manubhai Trivedi college of Information science, Surat, Gujarat, India.
Jagin M. Patel[2]
[2] M. K. Institute of Computer studies, Bharuch, Gujarat, India.
, jagin_2k@yahoo.com

Abstract:
Distributed Software Development Has Become Increasingly Common In Recent Years Due To Its Potential For Cost Savings And Access To A Larger Talent Pool. However, It Poses Significant Challenges Related To Coordination, Communication, And Collaboration. Collaborative Technology Can Enhance Distributed Software Development By Enabling Effective Communication And Coordination Among Team Members. This Paper Explores The Concept Of Enhancing Distributed Software Development With Collaborative Technology, Including An Overview Of The Various Collaboration Tools, Such As Version Control Systems, Issue Tracking Systems, And Project Management Tools. The Paper Serves As A Useful Reference For Researchers, Practitioners, And Stakeholders Involved In Distributed Software Development.

Key words: Collaboration, Distributed Software Development, collaborative development tools
.

1-INTRODUCTION:

Software engineering is by its very nature a team-based, collaborative activity [1]. Software engineers must work together, communicate, and coordinate at all stages of a project. However, the great majority of currently available software engineering tools, from complex CASE tools to programme editors, are solely intended to support single-user activity [1]. Such tools make it harder to collaborate and share knowledge, which limits the potential advantages of teams of developers working closely together [2].Our limitations as humans make it difficult for us to develop practically any kind of software. We are slow and prone to mistakes while working at high levels of abstraction, such as when defining requirements, designing software, and writing code, or developing test cases. As a result, we need to collaborate in order to finish big tasks on time and rely on others to see our errors. As soon as we start working together, other issues may arise. [3] discuss the objectives of software engineering collaboration which is helpful in the full development lifecycle. Distributed software development has become increasingly popular over the years as companies look to leverage the skills and expertise of developers located in different parts of the world. Although this strategy has several advantages, such as cost savings and access to a larger talent pool, it also has several problems. One of the biggest challenges is ensuring effective communication and collaboration among team members who are geographically dispersed. Collaborative technology has emerged as a key solution to this challenge. By leveraging tools and platforms designed specifically for distributed teams, companies can enhance their ability to communicate, coordinate, and collaborate effectively. These tools can range from simple chat apps to more sophisticated project management platforms that facilitate real-time collaboration on code, documents, and other project assets. [4] proposed GWSE (Global Working in Software Engineering) system to support geographically distributed teams which are flexible, resilient and scalable for use in demanding geographically distributed project environments. Whitehead [3] categorized collaboration tools into four categories namely model-based, Process centered collaboration, Collaboration infrastructure, and Awareness tools. To encourage teamwork on their projects, software developers have created a wide range of model-oriented technologies. These technologies cover every stage of the lifecycle, and they include col-

laborative requirements tools [5], software configuration management systems [6] etc. The goal of the WebDAV [7, 8, 9] project was to provide the Web with open interfaces for generating content. This would make it possible to integrate data among various software engineering tools. Additionally, WebDAV will lower the price of interorganizational and workgroup collaboration [9].In this context, the purpose of this paper is to explore how collaborative technology can enhance distributed software development. Specifically, we will discuss some of the key benefits of using these tools, as well as some of the challenges that teams may face when implementing them. We will also explore some of the best practices and strategies for successfully integrating collaborative technology into the software development process.

1. Collaborative Development Tools:

Here we will look into various Collaborative Development Tools in detail.

i. Web 2.0 Applications

Web 2.0 applications are now widely used. They serve as a useful tool for increasing team members' exchanges of information. For instance, wiki platforms have become a useful, affordable solution for building and preserving group documentation. Web 2.0 expands the capabilities of conventional collaborative software by allowing for direct user input, in-depth user interactions, and community development. Several web 2.0 applications can enhance distributed software development with collaborative technology. One example is SourceForge[9] which was launched in 1999 and quickly became a popular platform for hosting and distributing open-source software. Source Forge offered version control and collaboration tools including code repositories, issue tracking, and project management. Some other essential Web 2.0 applications include Blogging platforms e.g. Word Press, micro blogging platforms e.g. Twitter[10], social networking sites e.g. LinkedIn[11], Online marketplaces e.g. Amazon, etc.

ii. Project management tools

Project management collaboration tools are software applications that enable teams to plan, organize, track, and collaborate on tasks and projects. These tools provide managers with an overview of the status of a project at various detail levels, including the location and contact details of team members. Web-based interfaces are provided by collaborative project management software like ActiveCollab [12] and WorldView[13]. WorkSpaceActivityViewer uses data taken from developers' workspaces to present a summary of ongoing project activities [14].

iii. Version-Control Systems:

Version control system collaborative tools are software applications that enable teams to manage changes to source code and other types of files. These tools are commonly used by software development teams to collaborate on code and keep track of changes made to it.

To effectively manage the configuration of distributed software engineering, it is essential to use a version-control system. This system allows team members to share artefacts in a regulated way. Distributed file sharing is made possible using the well-known open-source version-control system. However, in recent years, Peer-to-peer (P2P) version-control systems have become increasingly popular. A peer-to-peer version control system is a type of version control system that operates in a decentralized manner, allowing for collaboration between team members without the need for a central server. In a P2P version control system, every developer has a complete copy of the project's repository on their local machine, rather than just a limited view of the data.This approach to version control provides several benefits, including increased resilience to network failures and improved collaboration in distributed teams. It allows developers to work on the codebase independently, without being dependent on a central server. P2P version control systems also enable developers to easily share changes with their peers and merge those changes seamlessly.

Examples of P2P version control systems include Darcs[15], Git[16], and Mercurial[17]. These tools have gained popularity in recent years due to their flexibility, scalability, and support for distributed

teams.

iv. Tracker collaborative development tools

Tracker collaborative development tools are software applications that enable teams to track and manage issues, bugs, tasks, and other types of work items related to software development. These tools are commonly used by software development teams to collaborate on projects and keep track of progress.Distributed trackers are software applications that work in a decentralized manner. In a distributed tracker system, every team member has a complete copy of the project's work items database on their local machine, allowing for independent work and offline collaboration.Distributed trackers provide several benefits over traditional centralized tracker systems. They offer increased resilience to network failures, improved collaboration in distributed teams, and better security by eliminating the need for a central server. Distributed trackers also enable developers to work on their local copies of the database and synchronize changes with their peers in a peer-to-peer manner.Examples of distributed tracker systems include Jira[18] and Bugzilla[19]: Jira is a tool that provides issue tracking and project management. Bugzilla is tool that provides bug tracking and team collaboration features.

v. knowledge centres

These content management systems enable the online sharing of explicit knowledge among team members. A knowledge centre may include frequently asked questions (FAQs), internal papers, technical references, standards, and best practices.

vi. Communication tool

Effective communication is critical for coordinating efforts, sharing information, and resolving issues. By providing a platform for real-time communication, information sharing, and collaboration, these tools can help to improve productivity, reduce errors, and ensure successful software development outcomes.Some examples of communication tools are Instant Messaging, Email, newsgroups, web forums, social media etc. that can be used as collaborative development tools:Email remains a popular communication tool for software development teams. It can be used to share project updates, send reports, or coordinate efforts with stakeholders who are not part of the development team. Social media tools, such as Twitter or LinkedIn, can be used to share project updates or to connect with other professionals in the industry. These tools can help to build relationships, facilitate information sharing, and promote collaboration.

vii. Requirements engineering

Requirements engineering tools used by engineers and stakeholders to support the process of eliciting, analyzing, documenting, validating, and managing software requirements. For example, DOORS and Borland CaliberRM [20].

viii. Test Tools

A testing team and a development team often work together during testing. Test collaborative tools used by software testing teams to manage and coordinate their testing activities, exchange information, and collaborate on testing tasks. TestLink[21] is test management tool that provides support for test case management, test plan management, and test execution.Selenium is a tool suite for automating web application. It provides a set of tools and libraries that enable developers to write automated tests for web applications. It supports multiple programming languages such as Java, C#, JavaScript, etc., making it a versatile tool for automating web applications in different environments.

ix. Customer Relationship Management (CRM)

Customer Relationship Management (CRM) tools help businesses manage their interactions with customers, automate sales and marketing processes, and improve customer satisfaction and retention. CRM tools are used to manage customer data, track customer interactions, and provide insights that can be used to improve customer engagement and loyalty.

2. Developments in Collaboration Tools

Nearly every day, new collaboration technologies and related best practices are developed. Mainly there are two key developments seen. First, all en-

gineering tools will have functionality for teamwork. These characteristics support the team's use of individual tools, but because they differ amongst tools, data integration is not possible. Improved engineering tool federation is a second, related development. Federation of engineering tools refers to the integration of different software applications or platforms used in engineering, to allow for seamless data exchange, collaboration, and workflow automation. It includes computer-aided design (CAD), product lifecycle management (PLM), and computer-aided manufacturing (CAM).There is currently no tool or Collaborative Development Environments (CDE) that handles all the tasks required for software engineering. Users must give priority to the tools they need to fulfil their collaborative needs.In the field of software engineering technology, nowadays, the idea of a Portable Common Tool Environment (PCTE) is widely accepted. The objective is to provide an enhanced, user interface integration while standardising a Public Common Tool Interface for software tools [22].

Conclusion:

Any business with scattered resources should prioritise providing effective tool support for collaboration. The most effective, dependable, and safe method for sharing software is with the use of the proper tool assistance.Overall, the use of collaborative technology has revolutionized the way software development teams work together, enabling them to work more efficiently and effectively. As technology continues to evolve, collaborative technology will continue to play an increasingly important role in distributed software development.

References:

[1] Vessey I and Sravanapudi A P: "CASE tools as collaborative support technologies", Comms ACM, Vol. 38, No 1, 1995, pp 83-95,1995

[2] Sellars P: "IPSEs in support of teams, in automating systems development", Benyon and Skidmore, Plenum Press, New York, 1987.

[3] Whitehead, Jim. "Collaboration in software engineering: A roadmap," In Future of Software Engineering (FOSE'07), pp. 214-225. IEEE, 2007.

[4] Gorton, I., Hawryszkiewycz, I. and Ragoonaden, K. Collaborative tools and processes to support software engineering shift work. BT Technology Journal 15, 189–198,1997

[5] B. Boehm and A. Egyed, "Software Requirements Negotiation: Some Lessons Learned," in the 20th International Conference on Software Engineering (ICSE'98), Kyoto, Japan, pp. 503-507, 1998

[6] Bolcer, Gregory Alan, and Richard N. Taylor. "Endeavors: A process system integration infrastructure." In Proceedings of Software Process, pp. 76-89. IEEE, 1996.

[7] L. Dusseault, "WebDAV: Next-Generation Collaborative Web Authoring", Prentice Hall PTR, 2003.

[8] E. J. Whitehead, Jr. and Y. Y. Goland, "WebDAV: A Network Protocol for Remote Collaborative Authoring on the Web," in 6th European Conference on Computer Supported Cooperative Work (ECSCW'99), Copenhagen, Denmark, pp. 291-310, 1999

[9] Whitehead, E. James, and Meredith Wiggins. "WebDAV: IEFT standard for collaborative authoring on the Web." IEEE Internet Computing 2, no. 5, pp. 34-40, 1998

[10] http://sourceforge.net

[11] http://twitter.com/

[12] www.linkedin.com

[13] www.activecollab.com

[14] A. Sarma and A. van der Hoek, "Towards Awareness in the Large," Proc. Int'l Conf. Global Software Engineering (ICGSE 06), IEEE CS Press, pp. 127–131, 2006

[15] R. Ripley, A. Sarma, and A. van der Hoek, "A Visualization for Software Project Awareness and Evolution," Proc. Int'l Workshop on Visualizing Software for Understanding and Analysis (VISSOFT), pp. 137–144,2007

[16] www.darcs.net

[17] www.git-scm.com

[18] http://mercurial.selenic.com

[19] www.atlassian.com
[20] www.bugzilla.org
[21] Borland Software Corporation, "CaliberRM 2006 User Tutorial", 2006
[22] http://testlink.sourceforge.net
[23] Campbell, Ian. "Portable common tool environment." Computer standards & interfaces 8, no. 1: 67-74, 1988