Journal of Advancement in Software Engineering and Testing
Volume 5 Issue 3

Enabling Extreme Programming (XP) in Global Software Development (GSD) Practice

Trailokya Raj Ojha[1]*, Prajwal Chaudhary[2]
[1]Assistant Professor, Department of Computer Science and Engineering
Nepal Engineering College, Bhaktapur, Nepal
[2] Department of Computer Science and Engineering
Nepal Engineering College, Bhaktapur, Nepal.

****Corresponding Author***
Email Id: *- trailokyaro@nec.edu.np*

ABSTRACT
Many software development companies currently are expanding their operations globally. Competitive advantage and the financial profits it brings to an organization are the driving forces behind the globalization of software. Organizations are benefiting from GSD, but communication has been a problem that has limited its expansion. In contrast to co-located projects, miscommunication and misunderstanding caused by the distance between development sites occur considerably more frequently in GSD projects, which ultimately affects customer satisfaction and software quality. Agile Methods are seen as excellent processes for GSD because of their flexibility. The most popular Agile approach, Extreme Programming (XP), is examined in our study, and different factors supporting GSD are suggested.

Keywords: *Extreme Programming, global software development, agile, success factors, virtual software development*

INTRODUCTION

The software sector has seen a consistent trend toward company globalization over the past few decades as the global economy has expanded. The schedule has become essential to the success of organizations launching products as IT development has grown globally spread. Recent years have seen a tremendous expansion of the worldwide software development industry, bringing with it innovations that affect application development initiatives. Because of Global Software Development's (GSD) advantages in a variety of areas, including cost reduction and the availability of competent labor, it is currently one of the most alluring approaches to software development. The software development activities are carried out at remote development sites using the GSD technique. When compared to GSD projects, in-house projects perform better. In-house software performs better in terms of quality than, GSD in terms of cost. In the GSD approach, this is caused due to less frequent communication between cross-site workers, language barriers, time zone, and intercultural issues.

Improving the different factors such as; frequent communication, proper development strategy, and proper set of key performance indicators, the cross-site project can perform better than the in-house project performance [1]. Today's extremely challenging software development process is made much more challenging by problems with trust and commitment, longer feedback loops, asynchronous communication, and knowledge management [2]. These problems appear to make it impossible to employ methods like agile techniques that rely on informal communication [3].

Agile Methods' introduction, with their focus on adaptability, informal teamwork, and working code, gave GSD a new way of software development. To gain the advantages of both, software development companies have been attempting to combine GSD projects with Agile Methods. The most popular of them, Extreme Programming (XP) [4], follows the principles of software development outlined in the Agile Manifesto [5] but goes beyond. XP is a collection of twelve distinct software development methods that were originally developed for small teams working on projects with plenty of change but have since been effectively extended to larger teams. However, there are several fundamental distinctions between XP and GSD.

GSD has presented its own distinct set of difficulties. Today's extremely challenging software development process is made even more challenging by problems with trust and commitment, longer feedback loops, asynchronous communication, and knowledge management [2]. These problems appear to make it impossible to adopt processes that rely on informal communication, like agile techniques. A GSD team's ability to communicate, especially informally, is essential to its success. In this study, we investigate the characteristics of XP and GSD initiatives to discover their shared interests and potential fusion areas.

BACKGROUND

In this section, a brief description of requirement engineering in global software development practice is illustrated. The background knowledge about agile methods and extreme programming is also presented in this section.

Global Software Development and Requirement Engineering Practices

Global software development (GSD) is currently one of the most enticing methods for software development due to its benefits in several areas, including cost reduction and the accessibility of qualified workers. Using the GSD technique, the software development tasks are completed at remote development sites. In-house initiatives function better than GSD projects. GSD is more expensive than in-house software while performing better in terms of quality. According to the GSD strategy, this is a result of fewer cross-site communications, language difficulties, time zone differences, and multicultural issues [6]. Improving the different factors such as; frequent communication, proper development strategy, and proper set of key performance indicators, the cross-site project can perform better than the in-house project performance [7].

Requirements-related activities, such as negotiation and requirements definition, design, and project management, are one type of software development activities directly impacted by communication issues. One of the biggest challenges in international firms is requirements engineering [8]. The several aspects that have an impact on the efficient management of requirements in GSD are identified by in-depth field research of requirements engineering in a global firm [9]. To get a shared understanding of the necessary functionality, the language barrier alone could be a major obstacle. Terms that have diverse connotations in various organizational contexts may not be correctly understood until the very end of the project, with potentially disastrous outcomes. Major difficulties affecting the entire software development process also result from the multicultural interactions between developers and clients, time delays, and the difficulty to keep track of the working environment at remote sites. Budget and schedule overruns and, eventually, strained client-supplier relationships emerge from a lack of understanding of all necessary system requirements, diminished trust, and an inability to successfully address disagreements [9].

Since distance has a major impact on requirements management operations [9], GSD projects benefit by having well-defined requirements specifications at the outset of the project [10], preventing the need to reaffirm feature comprehension. This specification is crucial for the structure and management of distributed projects and is frequently used in planning processes [8]. Frequent deliverables are also considered good practices in GSD. The complexity of the project environment will increase with the number of stakeholders, which will have an impact on how quickly the project moves forward. This issue becomes even more challenging when project teams are dispersed throughout the globe, there are several stakeholders in various nations, and there are individuals with diverse cultural backgrounds involved.

Agile Modelling and Extreme Programming

Software development techniques have evolved through time along with our culture. The goal of meeting environmental needs is demonstrated by the progress of development from the traditional Waterfall method to iterative and agile methods as shown in figure 1. The chaordic, practice-based methodology for effectively modeling and commenting software systems is known as agile modeling. It does not describe how to create the model; rather, it describes how modelers might be efficient. It is chaotic because it combines the randomness of straightforward modeling with the inherent order of software modeling artifacts.

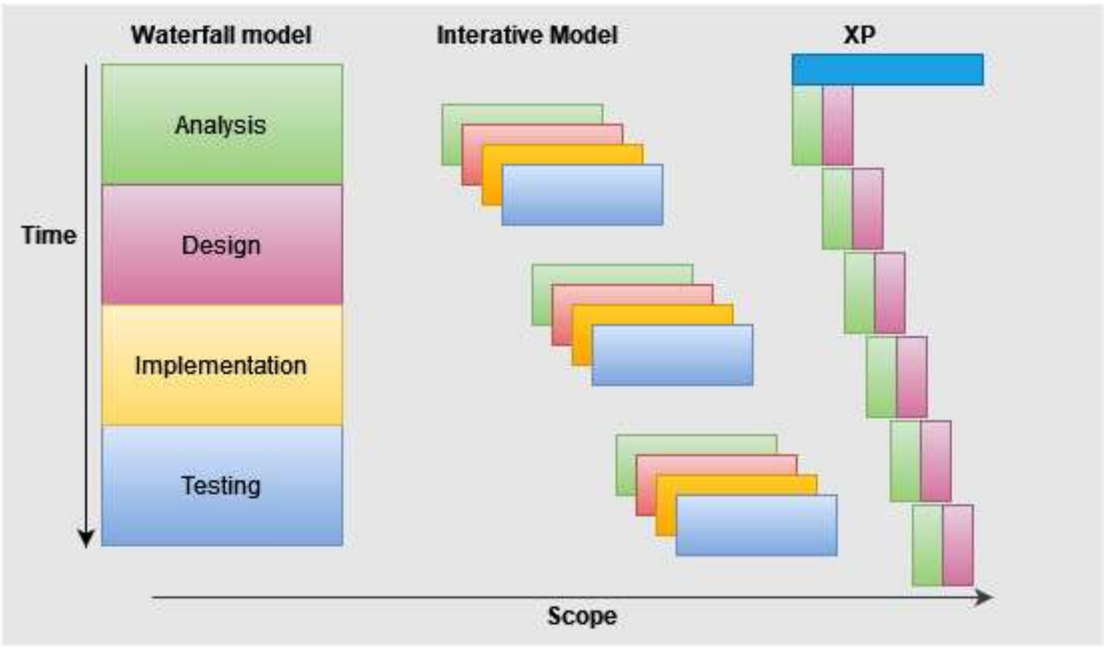

Fig.1:- *Evolution of Software Development Models [4]*

The best thing about agile modeling (AM) is that it allows you to choose the best characteristics from many software processes that are already in existence and model them using AM to create a process that meets your requirements. Although

AM is independent of other models like XP or UP, it significantly improves those processes. Agile modelers are anyone who uses AM practices by the agile methodology's principles and values. A developer who uses the agile method for software development is known as an agile developer. As a result, not all agile developers are agile modelers, but all agile modelers are agile developers.

The significance of project plans and documentation are emphasized in conventional project development methods. At the outset of the project, they work to identify every requirement and manage unforeseen changes as they arise. However, the development team frequently does not influence significant changes in requirements, scope, and technology in the present dynamic business context. The authors of [11] note that a project's ability to better manage unavoidable changes throughout its life cycle is frequently the matter at hand rather than how to limit changes in a project. Agile methodologies provide their solutions as a potential answer to this conundrum. Numerous agile methodologies that have been discussed or used for some time are listed in the Agile Manifesto, including Dynamic Systems Development Method (DSDM) [12], Feature Driven Development (FDD) [13], Internet-Speed Development (ISD) [14], Extreme Programming (XP) [4], SCRUM [15], Crystal [16], and Pragmatic Programming (PP) [17]. The most popular agile methodology among these is Extreme Programming.

A short planning horizon characterizes the iterative development approach known as XP (1–2 week iterations, 3 months releases). A release in XP refers to a stable, deployable version of the software that consumers can use. Iterations are shorter development increments in which specific tasks are given to developers and a functioning system prototype is developed and frequently assessed by project stakeholders. The creation of detailed requirements or design documentation before beginning development is not a part of the XP methodology. As a result, XP significantly relies on regular stakeholder communication and close feedback loops to describe feature implementation, provide clarity, and adapt to change. This paper's main point is that ongoing communication might be difficult for GSD teams. In XP, user stories are used to represent functional requirements. On an index card, user stories are informal, plain-language explanations of system functionality. The customer is in charge of creating each user story and designating the importance of each one. Each user story has a matching customer acceptance test (CAT) that, when passed, declares the user story to be finished. User stories are continually clarified and improved during the development process by developers and consumers because the first user narrative frequently lacks the exact information required for implementation.

The following points can be used to explain why XP is so widely used: First, XP covers the majority of the software development life cycle. Second, XP supports context appropriateness, which means it can be adjusted to meet the needs of specific projects. Third, and perhaps most importantly, XP is fully backed by real-world examples, unlike most methodologies. This research makes XP a realistic strategy by balancing these benefits against the traits of GSD.

APPLYING AGILE METHOD AND EXTREME PROGRAMMING IN GSD

Good communication is achieved in co-located software development projects by XP methods. Due to the delays in client input, it is more challenging to adapt to worldwide development projects. In XP initiatives, continuous contact between clients and developers is expected. However, when customers and engineers are not physically close to one another, their inability to provide timely feedback might pose a serious challenge. In our study, the best outcomes came from adapting and integrating certain current approaches with the best practices of XP, creating a channel of communication that is quicker.

Virtual Software Teams and XP
In virtual teams, members are located in various parts of the world and we are required to work on different components of a project which are independent of each other. Face-to-face interactions are not important for these types of teams. A virtual team and a distributed team are distinguished by the fact that members of a virtual team collaborate on the same projects [18].

Virtual teams have been proven to have great advantages in terms of business values and low financial requirements. Greater adaptability in locating necessary resources when they are needed is one of the primary benefits. Another one is the reduced cost as a result of outsourcing to locations with less expensive labor and lower training costs [19]. For applying XP in virtual software teams there are a few work processes that need to be considered. They are; project coordination, synchronous communication, and active notifications information routing, and integrated process execution with knowledge management [20].

Project coordination: Therefore, project coordination assistance is crucial for virtual teams. The XP team should be able to do to allocate tasks to development teams, set deadlines, and gain an overview of the project's present status. Team members should readily be able to access their to-do lists and find the information they need to complete their assignments.

Synchronous Communication
In XP, face-to-face communication is adopted rather than communication documentation. Since face-to-face communication is impractical for virtual teams due to location restrictions, it needs to be replaced by some technological tools. Synchronous communication tools like text chat, audio, video calls, etc. are used with occasional use of emails also.

Active Notifications Information Routing
Rather than simply making material available for pull access, it would be beneficial to push important information to users as soon as it becomes available. When key events take place in a project, notifications should be included in this push strategy.

Active Notifications Information Routing
There is a strong demand for training new employees on their tasks and maintaining reliable knowledge sources for the company due to frequent changes in the members of virtual teams. Maintaining the contents of an experience base up to date is a tough effort that needs to be linked as much as possible with the routine procedures of carrying out processes as software development often struggles due to rapidly changing technology.

EXAMINING EXTREME PROGRAMMING
This section examines XP practices within the context of global software development and discusses which practices may and cannot be included in GSD. We list the XP components that are essential for GSD project adaptation.

XP's Benefit in Communication

All stages of the lifecycle of software development require communication. Another fundamental principle of the XP discipline is communication. The techniques of XP are centered on enhancing various forms of communication. Table 1 provides an overall summary of which types of communication are benefited from XP techniques in general. As we can see, communication is beneficial to the majority of XP methods [21]. Due to the various types of barriers in GSD, some benefits of communication in globally distributed software development are difficult to achieve. The most necessary communication types needed in software development are listed below.

- Customer and project manager communication
- Customer and developer communication
- Developer and project manager communication
- Developer and developer communication
- Customer and customer communication.

Table 1:- *XP's common practices for benefiting communication [21]*

Practices	Benefits
Planning game	Communication between the project manager, developer, and clients is advantageous.
Small release	Benefits from quick customer and developer feedback.
Metaphor	Gives developers, project managers, and consumers a platform for simple, clear communication.
Simple design	Communication between developers and project management is made easier.
Tests	Provide rapid feedback between customers and developers.
Refactoring	Facilitates communication between clients and developers.
Pair programming	Instantaneous communication between paired developers is provided.
Continuous integration	Gives quick feedback to developers on the quality of the code.
Collective ownership	Communication between developers is beneficial.
On-site customer	Benefits from the improved customer, project manager, and developer communication.
40-hour weeks	Not identified
Open workspace	Communication is advantageous between developers and between developers and project management.

Examination of XP's Practices

XP's practice examination includes different techniques such as on-site customers, planning game, small release, simple design, and collective ownership. These techniques are discussed in this section.

On-Site Customer

One of the most significant XP practices is on-site customer support. The majority of XP initiatives demand an on-site customer. Customer availability throughout the majority of the project phases is crucial for the success of XP initiatives. He works with the development team and belongs to part of the team. It is quite challenging to have a customer available at all times while implementing XP methods in a GSD project. When the requirements are ambiguous, the on-site customer's role is to assist the developers [22]. A developed module or component of the software from the client is also clarified and approved by the developers.

In most circumstances, several clients will be involved in providing the development team with all of the requirements. Setting up on-site clients is expensive when a project is scattered around the globe. Additionally, there are other concerns, like getting a foreign visa, travel duration, etc. It's difficult to guarantee timely customer presence, especially in an emergency. Customers must travel between the sites when the project is spread across multiple locations, which lowers productivity and raises costs. To apply this strategy to GSD projects, a technology that can deliver the customer's virtual on-site presence is required. Globally dispersed development teams and customers can effectively communicate with one another through email, instant messaging, and conference calls [21].

Small Releases

The customer needs a lot of time to validate all of the new features when the code is delivered in a bulk release. Additionally, work could be put on hold as developers wait for the customer's approval to move on to the next release. Smaller releases simplify things. As a result, the validation period's length is likewise shortened [23].

Collective Ownership

Coordination of developers' activities in software development projects should include collective ownership and coding standards. The approach that controls how developers manage their work and contribute to the teamwork outputs of their colleagues is known as collective ownership. In complicated processes with high levels of reciprocal interaction, like software development, coordination of such activity is essential. Developers might not feel obligated to keep an eye on the software they collaborate on with others if there is no collective ownership. If such a sense of collective ownership is not there, errors or inefficient software code (such as functionality duplication) may go undetected.

Developers may be very protective of the code they are responsible for if there is no collective ownership. Any modifications to a specific piece of code must be negotiated with the person in charge of it [4]. As a result, the development process could experience bottlenecks. In conclusion, we anticipate that software project teams with common ownership will create software code that is of higher technical quality (fewer coding mistakes) than software project teams without collective ownership, everything else being equivalent [24].

Planning Game

The stories that have not yet been finished or scheduled for an iteration are all listed in the release planning. Users can utilize this to group particular stories that will be added to the current or upcoming iteration. Users' tasks and assignments can be displayed to them on the Iteration display. In the majority of XP initiatives, users are often allowed to sign up for a specific job as soon as it becomes available.

The disadvantage of this strategy is that it could be challenging for all users to be present every time a new job is added. This is particularly true for big distributed projects where the development team is often spread over many time zones [25].

Simple Design

XP keeps things simple, stays away from complex requirements, and uses straightforward designs up front, which provide clear and concise documentation that helps developers better understand the software project. XP practices put a greater emphasis on delivering functional software rather than producing vast quantities of documentation that add nothing to the actual software development process and impede the creation and delivery of sustainable software [26]. Accepting that their idealized system almost usually has extraneous or unnecessarily complex features is the customer's challenge. Two crucial early lessons for customers are learning to trade off features to deliver a story and assessing the anticipated development effort for a given feature in discussions with programmers. Project managers must work to control programmers' inclinations to write code that is more sophisticated than is required [27].

SUCCESS FACTORS

In a global software development scenario, usually, we think the team means a group of people working on one site but in actuality, it is a group of people working on various sites to achieve a common goal. While implementing XP in GSD, it is noted that teamwork and cooperation are the most significant success factors. A few common success factors are team, process, project, and project outcome factors.

Team Factors

Team factors in GSD always have a great role in the successful completion of the project. Proper implementation of XP in GSD has a great impact on the team for motivation towards the project by enabling scattered teams with a good communication channel, shared ideas, and problem-solving attitude.

A customer should work closely with the team throughout the whole life cycle, according to XP. This procedure necessitates that the client has a solid grasp of the needed program. In most circumstances, numerous clients will be involved in providing the development team with all of the requirements. Setting up on-site clients is expensive when a project is scattered around the globe. Additionally, there are other concerns, such as getting a foreign visa, travel duration, etc. It's difficult to guarantee timely customer presence, especially in an emergency. Customers must travel between the sites when the project is spread across multiple locations, which lowers productivity and raises costs. To apply this strategy to GSD projects, a technology that can deliver the customer's virtual on-site presence is required. Globally dispersed development teams and customers can effectively communicate with one another through e-mail, instant messaging, and conference calls. Telecommunication can take place at the start and conclusion of each release and iteration for teams that are more than five time zones apart, as well as whenever necessary. Email is less effective than face-to-face communication, yet it can still be answered within a day. Additionally, when there is a language barrier, writing is frequently preferred to vocal communication. Accommodating all the barriers helps in the positive growth of the project.

Process Factor

The effects of technologies utilized in the software development process are discussed in this section. The technological aspects, such as the software development methodology, project management, techniques for preventing and eliminating bugs, external/system testing, language, and reusable materials, provide an overview of the software development approach.

In a research conducted by [28], the team adopted many XP process principles and modified others to suit their needs. Test-driven development (TDD) was used by the team, and all code was unit-tested. According to the XPlanner tool, more than 75% of the work was completed in pairs while adhering to strict coding standards. Every eight hours, a regression test suite was run on the build machine on which the developers' work was integrated at least once a day. Additionally, the developers engaged in collective code ownership, even if a single pair would frequently complete most of the work on a given piece of code. In contrast to the typical XP procedure, planning sessions were only attended by the lead developers [28]. The success of the GSD project also relies on how and what kind of communication, management, and technical tools are used in the project. Among all, communication mechanisms and tools play a vital role as teams are distributed around the world in the GSD approach.

Project Factors
To get a better understanding of the project size and scope, the project's characteristics should be summarized and stated clearly. In a study [28], the project team delivered 65 user stories in the final product. Because the team was not familiar with XP, there was a wide range in the real amount of labor required for each user story. The development of a graphical depiction of hardware components for one user narrative in this project, for instance, took 45 hours. By contrast, it took 6 hours to validate an input instruction. The project required a total of 7.62 person-months of work, although this work was not dispersed equally. The level of effort rose as the release deadline drew near, peaked during the last revision before the release, and then began to decline as the project moved into its maintenance phase. The addition and removal of staff was the main cause of the effort variations. Four developers were initially part of the team, which grew to seven as the delivery deadline drew near. After delivery, there were just two developers left on the team, and they spent most of their time repairing bugs rather than developing new features. The modest size of the development company made it necessary to move workers between projects as needed to satisfy impending client requests. Ten working days were allocated to each iteration.

Project Outcome
Project outcome measures that concern the business-oriented results of the project are also a critical success factor for XP in GSD. Quality, productivity, and customer satisfaction are considered major factors for project success. If the outcome of the project addresses all these factors, the project is considered as successful. The project outcome should satisfy the customers based on the requirements presented by the customer.

CONCLUSION
From the study, it is clear that even while XP places a strong emphasis on communication and GSD is bred with a communication gap, the two can be combined to benefit from each other. Project management procedures are required when utilizing XP because it does not support it. We discovered that project management of crucial components makes it easier to install XP. These essential components include information on the project, the project site, the project team members, the user story, the project release strategy, the project iterations, and the project events. By giving each stakeholder a comprehensive picture of the project, the management of this

information aims to reduce the need for and difficulties of communication. For gaining more benefits by implementing XP in GSD, different factors such as factors team, process, project, and project outcome factors should be considered carefully. We believe that informal communication-centric approaches can be used to deliver effective initiatives despite time, language, and distance limitations.

REFERENCES

1. Setamanit, S. O., Wakeland, W., & Raffo, D. (2007, August). Improving global software development project performance using simulation. In *PICMET'07-2007 Portland International Conference on Management of Engineering & Technology* (pp. 2458-2466). IEEE.
2. Damian, D. (2003). Global software development: growing opportunities, ongoing challenges. *Software Process: Improvement and Practice*, *8*(4), 179-182.
3. Cockburn, A. (2001). Agile Software Development. *Addison-Wesley Longman, Reading, MA*.
4. Beck, K. (1999). Embracing change with extreme programming. *Computer*, *32*(10), 70-77.
5. Manifesto for Agile Software Development, http://agilemanifesto.org/. [Accessed on: December 03, 2022].
6. Ojha, T. R., & Mainaly, B. (2022). Implementation of Key Performance Indicators (KPI) for Global Software Development-A Comprehensive Study. *Journal of Advancement in Software Engineering and Testing*, *5*(2).
7. Setamanit, S. O., Wakeland, W., & Raffo, D. (2007, August). Improving global software development project performance using simulation. In *PICMET'07-2007 Portland International Conference on Management of Engineering & Technology* (pp. 2458-2466). IEEE.
8. Prikladnicki, R., Nicolas Audy, J. L., & Evaristo, R. (2003). Global software development in practice lessons learned. *Software Process: Improvement and Practice*, *8*(4), 267-281.
9. Damian, D., & Zowghi, D. (2003). Requirements Engineering challenges in multi-site software development organizations. *Requirements Engineering Journal*, *8*(1), 149-160.
10. Tiwana, A. (2004). Beyond the black box: knowledge overlaps in software outsourcing. *Ieee Software*, *21*(5), 51-58.
11. Highsmith, J., & Cockburn, A. (2001). Agile software development: The business of innovation. *Computer*, *34*(9), 120-127.
12. Stapleton, J. (1997). Dynamic systems development method -The method in practice: Addison Wesley.
13. Palmer, S., & Felsing, J. (2002). A practical guide to feature-driven development. Prentice *Saddle* Hall. *Upper Hill*.
14. Baskerville, R., & Pries-Heje, J. (2001). Racing the E-bomb: How the Internet is redefining information systems development methodology. In *Realigning research and practice in information systems development* (pp. 49-68). Springer, Boston, MA.
15. Schwaber, K. (1995). Scrum Development Process. OOPLSA’95 Workshop on Business. *Object Design and Implementation. Austin*.
16. Cockburn, A. (2001). *Writing effective use cases*. Pearson Education India.
17. Thomas, D., & Hunt, A. (2019). *The Pragmatic Programmer: your journey to mastery*. Addison-Wesley Professional.
18. Jalali, S., & Wohlin, C. (2012). Global software engineering and agile practices: a systematic review. Journal

of software: Evolution and Process, 24(6), 643-659.

19. Maurer, F. (2002, August). Supporting distributed extreme programming. In Conference on Extreme Programming and Agile Methods (pp. 13-22). Springer, Berlin, Heidelberg.
20. Maurer, F., & Martel, S. (2002, May). Process support for distributed extreme programming teams. In ICSE 2002 workshop on global software development.
21. Tian, Y. (2009). Adapting Extreme Programming for Global Software Development Project (Master's Thesis). Auburn University, Auburn, AL 36849, United States.
22. Shah, S. M., & Amin, M. (2013). Investigating the Suitability of Extreme Programming for Global Software Development: A Systematic Review and Industrial Survey.
23. Xiaohu, Y., Bin, X., Zhijun, H., & Maddineni, S. R. (2004, May). Extreme programming in global software development. In Canadian Conference on Electrical and Computer Engineering 2004 (IEEE Cat. No. 04CH37513) (Vol. 4, pp. 1845-1848). IEEE.
24. Turk, D., Robert, F., & Rumpe, B. (2005). Assumptions underlying agile software-development processes. Journal of Database Management (JDM), 16(4), 62-87.
25. Reeves, M., & Zhu, J. (2004, June). Moomba–a collaborative environment for supporting distributed extreme programming in global software development. In International Conference on Extreme Programming and Agile Processes in Software Engineering (pp. 38-50). Springer, Berlin, Heidelberg.
26. Rashid, N., & Khan, S. U. (2018). Agile practices for global software development vendors in the development of green and sustainable software. Journal of Software: Evolution and Process, 30(10), e1964.
27. Bailey, P., Ashworth, N., & Wallace, N. (2002, May). Challenges for stakeholders in adopting XP. In Proc. 3rd International Conference on eXtreme Programming and Agile Processes in Software Engineering-XP (pp. 86-89).
28. Layman, L., Williams, L., Damian, D., & Bures, H. (2006). Essential communication practices for Extreme Programming in a global software development team. *Information and software technology*, *48*(9), 781-794.