International Journal of Engineering Research and Technology. ISSN 0974-3154, Volume 14, Number 10 (2021), pp. 999-1004
© International Research Publication House. http://www.irphouse.com

Descriptive Analysis of the Agile Methodology Extreme Programming (XP) for its Implementation in Software Development

Holman Montiel Ariza[1], Luis F. Wanumen Silva[2] and Lely A. Luengas Contreras[3]

Facultad Tecnológica, Universidad Distrital Francisco José de Caldas, Bogotá D.C, Colombia.

[1]*ORCID:* 0000-0002-6077-3510, [2]*ORCID:* 0000-0002-8877-5681
[3]*ORCID:* 0000-0002-3600-4666

Abstract

Agile methodologies are one of the most booming tools in recent decades; this is because its application in industries not only in the software development sector, but in all areas has resulted in the optimization of processes within project management. Therefore, throughout this document, one of the most used methodologies is Extreme Programming (XP), whose structure is specified in detail, covering various factors such as: values, principles, and practices that when implemented are focused on strengthening product quality, customer satisfaction, assertive communication between team members and others involved, and reducing downtime based on continuous feedback. Additionally, emphasis is placed on the phases of the methodology; where the activities that make up each of them and the roles necessary for the development of the different processes that compose it are described.

Keywords: Agile Methodology, Project Management, Extreme Programming, Software Development.

1. INTRODUCTION

The advance of technologies and the continuous development of innovation has generated that organizations manage in an ideal way the resources available for the execution of their projects; this with the aim of achieving efficiency in less time, with reduced costs and meet the changing needs of customers [1].

There are two types of methodologies used for project management within companies, traditional methodologies, and agile methodologies. Initially in the software development industries traditional methodologies were applied such as: classical waterfall model [2], iterative waterfall model, spiral model [3] or RAD model, which were focused on performing sequential processes whose phases depended on each other, and the construction of the product began once the design phase was defined [4].

On the other hand, due to the rapid growth of the software development industry in 2001 a group of 17 expert process analysts discussed the standard characteristics of the applied methodologies and as a result the Agile Manifesto [5][6] was obtained, which was written in order to define the basic, modern and simple principles and values for agile software development [7].

Given the above, agile methodologies focus on continuous change, implementing small changes that reflect customer requirements through iterations allowing a results orientation; also the organizational structure that conforms them is collaborative, based on the adaptability and cross-functionality of the teams, acquiring effective feedback responding quickly to the environment [8].

Currently, agile methodologies are not only applied in software development; the search for agility within organizations has generated that they are implemented according to the criteria or needs that arise within the processes during the development of a project; so there are several types such as: Scrum, Crystal, Kanban, Extreme Programming (XP), Dynamic Systems Development Method (DSDM), Prince 2, among others [9][10][11][12].

This paper addresses Extreme Programming (XP) which is a lightweight software development methodology created by Kent Beck [13], who developed a book where he explains each of its components [14], because its structure is based on simplicity, communication and feedback of the developed code [15], therefore, it will be analyzed in detail presenting each of the features that make it for the application within any organization.

2. MATERIALS AND METHODS

First, the structure of Extreme Programming (XP) will be analyzed, which is made up of three parts: Values, Principles and Practices, which will be described in detail in the development of the document as they are the starting point. Additionally, the phases of the methodology will be broken down where the processes that make up each of these phases and the roles involved in the implementation will be identified. Finally, the advantages of the application of Extreme Programming (XP) in organizations are exposed.

3. DEVELOPMENT

The Extreme Programming (XP) methodology among its main objectives seeks to promote interpersonal relationships through the strengthening of teamwork; ensure the quality of the software developed, improve productivity and minimize risk by acting on project variables (cost, time, scope and quality) [16][17]; these objectives are oriented to assume the change naturally to adapt immediately. Likewise, the life cycle of the XP process [18] is defined as shown in the Figure 1.

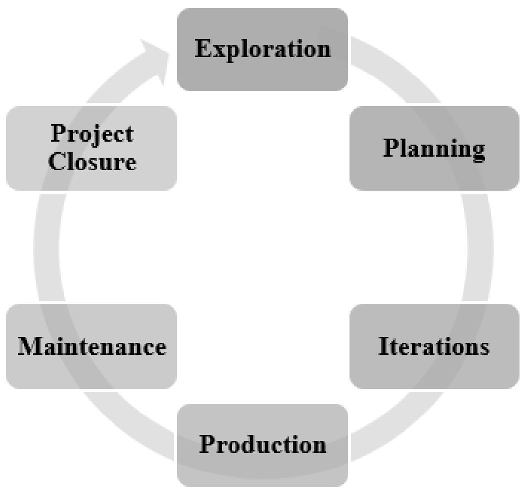

Fig. 1. XP Process Life Cycle

- **Exploration:** It consists of the approach of the client's requirements through user stories which are important to develop the first delivery of the product.

  User stories are basically a written representation of the requirements as seen from the end user's perspective; they are intended to describe for developers in a practical way what the end user really wants in relation to the functionality, see Table 1.

Table 1. Example of a user story

Number:	User:
Name of the Story:	
Priority:	Developmental Risk: High () Medium () Low ()
Estimated points:	Assigned Iteration:
Responsible programmer:	
Description: Like: I want to: To:	
Validation:	

- **Planning:** The priority of each user story is established, and the effort involved in their development is estimated.

- **Iterations:** In this stage, continuous revision of the generated product is performed, so tests are performed and a feedback and integration process is maintained.

- **Production:** Additional tests are carried out and the performance of the product is evaluated before it is presented to the customer.

- **Maintenance:** The updates required by the client are developed.

- **Project closure:** At this point the client does not require to include more user stories, so the final delivery of the product is done [14].

In contrast, the structure of the methodology is composed of three parts: values, principles, and practices, each of which is described below, see Figure 2.

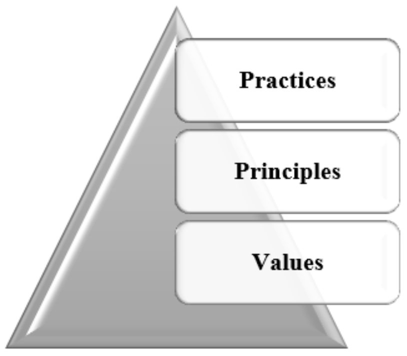

Fig. 2. XP Structure

I. Values

By means of the values, it is sought that the work teams emphasize a collaborative attitude and effectively face the changes generated in the project. Therefore, the methodology is based on the following values.

➢ **Communication:** This value is not only focused on communication between the work team but also on communication with developers and customers; to obtain a continuous exchange of information and detect in the shortest possible time any error.

➢ **Simplicity:** It looks for the simplest solution by developing the functions for each process at the indicated time; at the same time, it focuses on the creation of simple programming codes with the objective of facilitating the understanding of the complete equipment.

➢ **Feedback:** By means of this value the client can elaborate the necessary critiques to obtain the requested product; in the same way, the processes are executed in short cycles with the purpose of verifying the code and delivering advances of the product to the client.

➢ **Courage:** It is related to the willingness to acknowledge mistakes within the product development, analyze the working methods and sometimes start coding again.

➢ **Respect:** It refers to the cordiality of the team, as well as not to harm the members of the group with negative modifications that affect the development of any member [19].

II. Principles

The principles are the connection between values and practices, i.e., they are the link between the abstract and the concrete, they come from the values and the methodology mentions 14:

- **Humanity:** The software is developed by people.
- **Economy:** Ensures the economic value of the product made.
- **Mutual benefit:** Practices should benefit everyone involved.
- **Self-similarity:** Apply known patterns to the solution of different problems.
- **Improvement:** Seeks excellence through continuous review to deliver the expected product.
- **Diversity:** Getting different ideas for the development of a product broadens the vision of this and generates new opportunities for its development.
- **Reflection:** The cohesion between the work teams allows to analyze the processes and not to hide the mistakes.
- **Flow:** Enables continuous delivery of value to the customer.
- **Opportunity:** Every problem creates an opportunity for change.
- **Redundancy:** Complex parts should be analyzed several times.
- **Failure:** Each failure brings new learning.
- **Quality:** The reduction of quality does not represent greater speed in the execution of a process.
- **Small steps:** Each advance within the project must contribute to the client's requirements and add value to the product.
- **Accepted responsibility:** Assigning tasks to a specific member allows them to take responsibility for their developments [20].

III. Practices

Extreme Programming proposes practices of planning, organization, communication, and software development, which combined with the values and principles generate a culture of excellence. Kent Beck in the second edition of his book classifies practices into two groups: primary and complementary practices; the combination of both optimizes effectiveness.

Primary

These practices provide immediate benefits in a safe manner. There are 13 of them and they are detailed below:

a. **Sit Together: It** is characterized by being an open space where the whole team works, where meetings are also held, generating greater productivity, and strengthening communication.

b. **Whole Team:** Seeks to build trust to foster support, growth and learning among team members.

c. **Informative Workspace:** It is a space assigned to view the progress of the project graphically, in turn, user stories are displayed and grouped according to their status.

d. **Energized Work:** This refers to performing activities in such a way that the employee achieves a high level of productivity during assigned work hours and effective time management.

e. **Pair Programming:** The code produced must be written in pairs, in order to analyze, design, test and elaborate the corresponding improvements, clarifying ideas and reaching the required standards [21].

f. **Stories:** User stories should be written on small cards indicating the name, description and estimated time and should be placed in a visible place.

g. **Weekly Cycle:** It is a short cycle that starts with a meeting at the beginning of the week to evaluate the progress to date, the tasks are formulated and assigned responsible; in the same way, the tests to be performed during the week are scheduled once the user stories are complete to be integrated into the project.

h. **Quarterly Cycle:** It is a follow-up meeting held every three (3) months in order to evaluate the overall progress of the project, this is to identify bottlenecks and choose the stories to be developed during the next quarter.

i. **Slack:** It raises a time frame to soften the tension of meeting impossible commitments, enhances credibility, and builds relationships based on honest communication.

j. **Ten-Minute Build:** It is based on ten minutes to build the entire system and run the tests, this process should be automated; as well as the deployment and release of new features in production. Moreover, testing should be performed for all parts of the system not only for the newly added parts.

k. **Continuous Integration:** The objective is to obtain a system ready to be launched without any problem, for this the programmed changes must be integrated, there are two types of integration:

- Synchronous integration: This is the one where the pair of programmers upload the changes a couple of hours after their elaboration; waiting for the build to be completed and that all the tests have been passed without any problem.
- Asynchronous integration: The daily build is elaborated at night, a new version of the system is built and if errors occur, the corresponding alerts are generated to provide the appropriate solution.

l. **Test- First Programming:** It is an evolutionary approach that seeks to generate quality code, based on writing the test first and improving it through refactoring.

m. **Incremental Design:** It is the gradual design of the product with the objective of improving its functionality at each stage; increasing its value and bringing it closer to the product expected by the customer [22].

Complementary

These practices should be implemented once the primary practices have been mastered and are classified into four groups as shown in Figure 3.

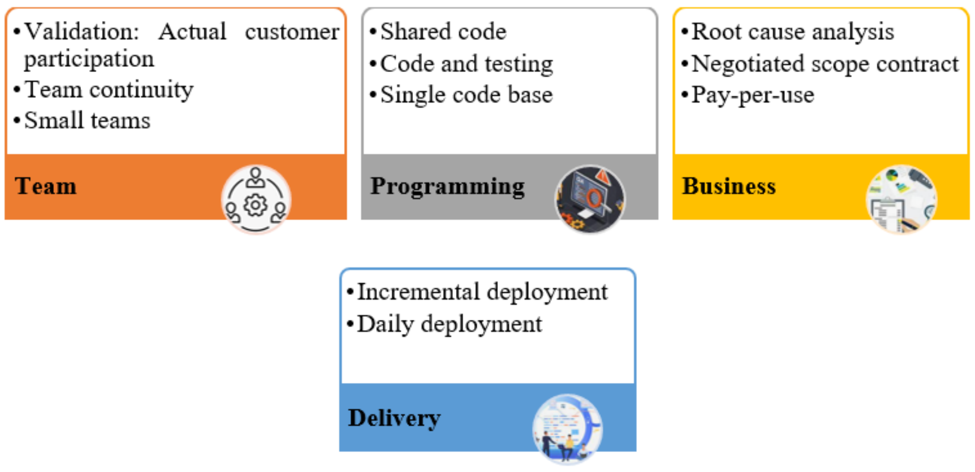

Fig. 1 Complementary Practices

4. DISCUSSION AND APPLICABILITY

From another perspective, the Extreme Programming (XP) methodology consists of four phases [23] as depicted in Figure 4, which contain specific activities for each phase and roles related to the activities for its implementation.

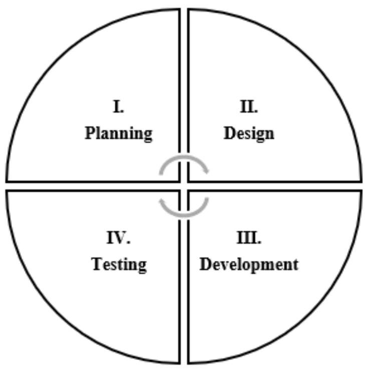

Fig. 2XP Methodology Phases

I. **Planning**: This phase is focused on obtaining an effective communication between those involved in the project; and in this way to know the needs presented by the client for the development of the product. On the other hand, a delivery schedule is established to start with the iterations process. The planning has activities or processes that help in the development of the product and complement the phase, these activities are:

- User Stories: It is a document written by the client which details the needs of the system; it is assigned a priority level for its development and according to this the costs are estimated and generally calculated according to the weeks needed for each story. These should be able to be scheduled between one and three weeks, if it exceeds the time should be divided into several stories and if less than a week should be merged with another [24].

- Delivery Plan: In the delivery schedule is planned the union of user stories to be delivered and the order of these according to customer priorities.

- Iteration Plan: It starts with a meeting to plan the iteration, there the user stories are selected and decomposed into tasks; which are assigned for development and testing within each cycle in the established order.

- Daily follow-up meetings: Meetings are held with the objective of maintaining communication between the team and sharing the difficulties and solutions presented.

II. **Design**: In this phase simple and clear designs are made to facilitate the development, likewise the following concepts are used:

- Spike solutions: These are small test programs to examine various solutions when there is difficulty in estimating user story times or technical problems.

- Recoding: The process of writing part of a code again without changing the functionality, but with the aim of making it simpler and clearer.

- Metaphors: It is the way to explain in a simple way the purpose, structure, and architecture of a project.

In addition to the previous concepts, it is possible to use sequence diagrams, which facilitate in this phase the visualization of the application behavior from a functional perspective, see Fig. 5.

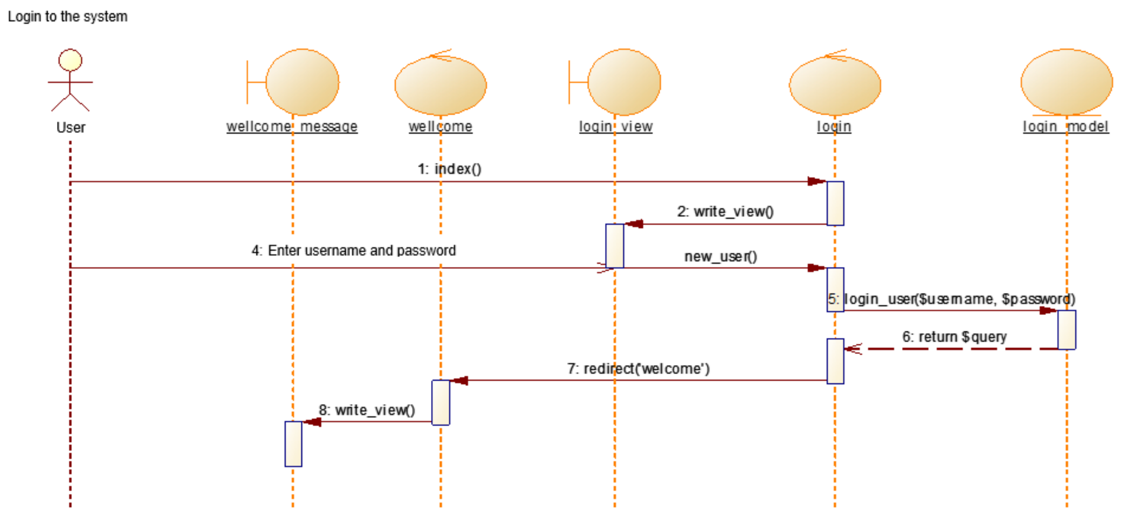

Fig. 5. Example of sequence diagrams

III. **Development:** In this phase the developers must design the unit tests that correspond to each user story; then the coding is elaborated so that in the implementation it passes the unit tests.

IV. **Testing:** It is the phase in which the elaborated code is implemented, initially it is verified that it does not generate errors; in case of generating them they must be corrected quickly and the acceptance tests are performed according to the user stories, verifying that they have been correctly implemented [25].

Finally, the methodology has specific roles to conform the work teams, which generates the assignment of responsibilities oriented to fulfill the objective of the project, the most relevant roles are:

- Developer: Writes the code and develops the activities of the project.

- Customer: Write user stories and set priorities.

- Tester: Performs the product tests and the quality of the product depends on him.

- Project Tracker: Monitors the progress of the software and detects problems in the product.

- Coach: Oversees teamwork and teaches how to implement more effective practices.

- Doomsayer: Tracks project risks and informs the team about them [26].

Conclusion

The use and implementation of agile methodologies such as XP methodology in software development, generates the minimization of errors due to the planning of short cycles or iterations; which allow feedback in each process including customer participation and contributing to the solution of problems in less time, thus reducing project risks and increasing product quality in each delivery. On the other hand; the increase of the productivity of the work teams is reflected thanks to the fact that the programmers can develop the projects in the way they consider optimal; either by carrying out iterations that allow the integration and validation of each of the components developed to meet the needs of the client; allowing the execution of their activities to be carried out within the assigned working hours with the objective of not having work overload for the collaborators.

REFERENCES

[1] V. Fustik, “The advantages of agile methodologies applied in the ICT development project,” Interntational J. Inf. Technol. Secur., vol. 9, no. 4, pp. 51–62, 2017.

[2] W. Van Casteren, “The Waterfall Model And The Agile Methodologies : A Comparison By Project Characteristics-Short The Waterfall Model and Agile Methodologies,” ResearchGate, no. February, pp. 10–13, 2017.

[3] S. Dhir, D. Kumar, and V. B. Singh, “Success and Failure Factors that Impact on Project Implementation Using Agile Software Development Methodology In: Hoda M., Chauhan N., Quadri S., Srivastava P.,” Softw. Eng. Adv. Intell. Syst. Comput., vol. 731, 2019.

[4] M. Sameen Mirza and S. Datta, “Strengths and Weakness of Traditional and Agile Processes - A Systematic Review,” J. Softw., vol. 14, no. 5, pp. 209–219, 2019.

[5] T. Krehbiel et al., “Agile Manifesto for Teaching and Learning,” J. Eff. Teach., vol. 17, no. 2, pp. 90–111, 2017.

[6] V. Stray, N. B. Moe, and R. Hoda, “Autonomous agile teams: Challenges and future directions for research,” ACM Int. Conf. Proceeding Ser., vol. Part F1477, pp. 1–5, 2018.

[7] S. Shaikh and S. Abro, “Comparison of traditional and agile software development methodology: A short survey,” Int. J. Softw. Eng. Comput. Syst., vol. 5, no. 2, pp. 1–14, 2019.

[8] D. Beerbaum, “Applying Agile Methodology to regulatory compliance projects in the financial industry: A case study research,” J. Appl. Res. Digit. Econ., vol. 2, no. Special Issue, pp. 1–11, 2019.

[9] A. López-Alcarria, A. Olivares-Vicente, and F. Poza-Vilches, “A Systematic Review of the Use of Agile Methodologies in Education to Foster Sustainability Competencies,” Sustainability, vol. 11, no. 10, pp. 1–29, 2019.

[10] D. Sánchez, F. Lizano, and M. Sandoval, “Integration of Remote Usability Tests in eXtreme Programming: A Literature Review,” Uniciencia, vol. 34, no. 1, 2020.

[11] R. Kumar, P. Maheshwary, and T. Malche, “Inside Agile Family Software Development Methodologies,” Int. J. Comput. Sci. Eng., vol. 7, no. 6, pp. 650–660, 2019.

[12] Ö. Uludağ, M. Hauder, M. Kleehaus, C. Schimpfle, and F. Matthes, “Supporting Large-Scale Agile Development with Domain-Driven Design. In: Garbajosa J., Wang X., Aguiar A.,” Lect. Notes Bus. Inf. Process., vol. 314, 2018.

[13] J. Choma, E. M. Guerra, and T. S. da Silva, “Developers’ Initial Perceptions on TDD Practice: A Thematic Analysis with Distinct Domains and Languages. In: Garbajosa J., Wang X., Aguiar A. (eds) Agile Processes in Software Engineering and Extreme Programming. XP 2018.,” Lect. Notes Bus. Inf. Process., vol. 314, pp. 68–85, 2018.

[14] K. Beck and C. Andres, Extreme Programming Explained: Embrace Change, Second Edi. Pearson Education, Inc., 2005.

[15] D. Gopaul, Software Methodologies: SCRUM vs Extreme Programming, Lulu Press. 2017.

[16] J. Pollack, J. Helm, and D. Adler, “What is the Iron Triangle, and how has it changed?,” Int. J. Manag. Proj. Bus., vol. 11, no. 2, pp. 527–547, 2018.

[17] S. Kunwar, “Enabling and Limiting factors in eXtreme Programming (XP) with Evaluation Framework,” SCITECH Nepal, vol. 14, no. 1, pp. 50–62, 2019.

[18] M. Ibrahim et al., “Presenting and Evaluating Scaled Extreme Programming Process Model,” Int. J. Adv. Comput. Sci. Appl., vol. 11, no. 11, pp. 163–171, 2020.

[19] J. C. Salazar, Á. Tovar, J. C. Linares, A. Lozano, and L. Valbuena, “Scrum vs XP : Similarities and Differences,” Tecnol. Investig. y Acad., vol. 6, no. 2, pp. 29–37, 2018.

[20] S. González and L. Fernández, “Programación Extrema: Prácticas, Aceptación y Controversia,” Culcyt / Softw., no. 14, pp. 55–62, 2006.

[21] O. A. Pérez A., “Cuatro enfoques metodológicos para el desarrollo de Software RUP – MSF – XP - SCRUM,” Inventum, vol. 6, no. 10, pp. 64–78, 2011.

[22] F. Anwer, S. Aftab, S. Shah Muhammad Shah, U. Waheed, S. M. Shah, and U. Waheed, “Comparative analysis of two popular agile process models: extreme programming and scrum,” Int. J. Comput. Sci. Telecommun., vol. 8, no. 2, pp. 1–7, 2017.

[23] L. Vázquez, A. Valdez, and G. Cortes, “Use of Extreme Programming to develop a system in the mining industry,” CienciaCierta, no. 61, 2020.

[24] M. Ecar, F. Kepler, and J. P. S. Da Silva, “Cosmic User Story Standard. In: Garbajosa J., Wang X., Aguiar A. (eds) Agile Processes in Software Engineering and Extreme Programming. XP 2018.,” Lect. Notes Bus. Inf. Process., vol. 314, pp. 3–18, 2018.

[25] N. Hasanah, M. B. Triyono, G. N. I. P. Pratama, Fadliondi, and I. G. N. D. Paramartha, “Markerless Augmented Reality in Construction Engineering Utilizing Extreme Programming,” J. Phys. Conf. Ser., vol. 1737, no. 1, 2021.

[26] R. Jeffries, A. Anderson, and C. Hendrickson, Exteme Programming Installed, Pearson Ed. Canada, 2001.