ISSN (Print) : 0974-6846
Indian Journal of Science and Technology, *Vol 12(35), DOI: 10.17485/ijst/2019/v12i35/146797, September 2019* ISSN (Online) : 0974-5645

A Proposed Approach for Tracing the Progress of XP Projects

Abed S. Alsahli[1] and Nagy Ramadan Darwish[2]*

[1]Faculty of Business, University of Jeddah, KSA, Jeddah–23218, Saudi Arabia; asalshle@uj.edu.sa
[2]Faculty of Graduate Studies for Statistical Research, Cairo University, Giza–12613, Egypt; nagyrd@hotmail.com

Abstract

Background/Objectives: Agile software development methods have gained popularity and are increasingly important to a significant number of software development organizations. eXtreme Programming (XP) approach is one of the most popular agile development methods. Tracing the progress of software projects is one of the most crucial success factors of these projects. This study is concerned with proposing an approach for tracing the progress of XP software projects. **Methods analysis:** This study introduces the values, practices, and life cycle phases of XP approach. Depending on simple statistical techniques, this study introduces a proposed quantitative approach to evaluate the degree to which various XP phases and activities are implemented. **Findings:** A software organization can use the proposed approach to track, evaluate, control, and improve the performance of XP software projects to increase the success rate of software projects. **Improvements/Applications:** The proposed approach can serve as a base for building a software tool to trace the progress of agile software projects.

Keywords: Agile Software Development, Extreme Programming, Project Progress, XP Phases, XP Practices

1. Introduction

Now, agile software development methods have gained more attention in the domain of software engineering[1]. Agile software development methods are the systematic and tested processes of delivering proper solutions to customer[2]. Agile software development methods are processes that are iterative, incremental, self-organizing and emergent[3]. It can be defined as a connotation of flexibility, nimbleness, readiness for motion, activity, dexterity in motion, and adjustability[4]. Agile software development methods deal with unstable and volatile requirements by using a number of techniques, focusing on collaboration between teamwork and customers and support early product delivery[5]. Agile Software Development (ASD) methods include five common methods: eXtreme Programming (XP), Scrum, Crystal, Feature Driven Development, Dynamic System Development Methodology, and Adaptive Software Development[6].

Kent Beck developed XP approach at Chrysler while working on a payroll project. Beck continued to improve XP approach after achieving the project until the new approach gained worldwide acceptance in 2000[2]. XP approach is one of the most popular agile software development methods. XP approach depends on incremental planning, short development cycles, continuous feedback, and reliance on communication and evolutionary design[1]. XP is a light-weight methodology for small-to-medium-sized teams developing software in the face of vague or rapidly-changing requirements[7].

XP approach depends on four important values that include: simplicity, communication, feedback and courage. These values are implemented with twelve core practices: planning game, small releases, metaphor, simple design, testing, refactoring, pair programming, collective code ownership, continuous integration, 40-hour week, on-site customer, and coding standard. In addition, XP

*Author for correspondence

life cycle phases include six phases: exploration, planning, iterations to release, production, maintenance, and death.

Tracing the progress of software projects is one of the most crucial success factors of these projects. Tracing the progress of software projects includes monitoring the achievements in every XP phase. Therefore, this study is concerned with proposing an approach for tracing the progress of XP projects. This study proposes a quantitative approach based on a combination of simple statistical techniques to evaluate the degree to which various XP activities are implemented. Software organizations can use the proposed approach to track, evaluate, control, correct, and enhance the performance of XP software projects to increase the success rate of software projects.

The rest of research is organized into six sections. Section 2 provides a background about XP values, practices, phases, activities, and methods of project tracing in agile projects. Section 3 presents the proposed approach. Section 4 presents how to apply the proposed approach to real projects and a discussion of the results. Section 5 presents the conclusion of the study. The last section includes some related ideas that can be adopted in future.

2. Background

XP approach introduces best practices for a good quality software product at small scale[8]. sXP approach is based on four main values that are achieved through twelve main practices. The development activities related to a software project using XP approach can be viewed or understood in the form of life cycle phases. This section consists of three parts; the first part presents XP values and practices, the second part focuses on XP phases and activities, then the third part focuses on the project management approaches in agile projects.

2.1 XP Values and Practices

XP approach is based on four core values: simplicity, communication, feedback and courage. Figure 1 illustrates the four core values of XP[9]. Simplicity is one of the values supported explicitly by XP[10]. XP team members are encouraged to start with the simplest solution that needs less time to finish than a complex one. Simple design increases the speed of software development while still retaining an emphasis on working software[7]. Then, extra functionality can be added later. Continuous communication helps XP team members and users to own a unified view of the requirements that reduces the possibilities of ambiguities and misunderstandings of the requirements. In XP projects, good performance can be achieved using story cards to collect requirements, wall boards, and shared workspaces to maximize face-to-face communication[11].

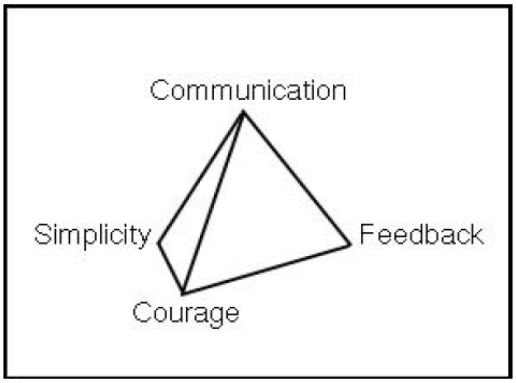

Figure 1. XP values[9].

XP team members should have methods and techniques for getting information about their performance in the project. Feedback may include many dimensions such as: the system, customer, and team. Feedback from the system and the team aims to provide project leaders with quick indicators of the project's progress to take corrective or supportive actions. In addition, feedback from customer includes the functional and acceptance tests. XP approach encourages teams to make important decisions that support XP practices. Courage enables developers to feel comfortable with refactoring their code when necessary[12].

XP approach is suitable for small to medium-sized teams developing software based on vague or rapidly changing requirements, it has twelve main practices: Planning Game, Simple Design, Small Releases, Testing, Metaphor, Refactoring, Pair Programming, Collective Code Ownership, Continuous Integration, On-site Customer, 40-hour Week, and Coding Standards. Software companies are progressively adopting development practices associated with XP approach[6, 7]. There are mutual relationships among XP practices. Therefore, any XP practice doesn't stand well on its own. They require the other practices to keep them in balance[7]. XP practices can be briefly explained in the following:

- The planning game is a collaborative process that is achieved by all XP team members to cre-

ate, estimate, and prioritize requirements for the next release[13]. The first release provides the selected functionalities and many releases could be necessary done before all functionalities are completed[14].

- The right design of the software at any point of time is: the one that runs all the tests, has no duplicated logic, states every intention important to the developers, and has a minimum number of classes and methods[7]. XP approach encourages teams to use the simplest possible design that will satisfy the current needs[15].
- The processes in XP project are divided into a sequence of small iterations where each one implements new features testable and accepted by the customer[16]. XP speed up the software delivery using short releases of 3-4 weeks.
- In traditional software development methods, testing is a phase of development that is carried out after the main coding effort[17]. In XP approach, tests must be created prior to coding. Programmers write unit tests whereas customers write functional tests.
- The system metaphor is a shared view that expresses the overall method in which the system wills operate[4]. It is an effective method for all team members to visualize the project.
- Refactoring includes restructuring the system without changing its behavior to remove duplication, improve flexibility, or simplify the work to save time of development and increase quality[10,17].
- Pair Programming means that two developers work together to achieve a programming task using one machine. It is useful to: perform an immediate peer review of code, reduce the time required for task completion, train junior developers, deal complex tasks.
- Collective Code Ownership means that after testing the code and adding it into the code base, the code can be modified by any team member[18].
- Continuous Integration means that the code related to each story is integrated into the evolving system as soon as it is accepted[4]. XP team members integrate and build the system multiple times per day to achieve the target scope.
- On-site Customer means that a customer works with XP team members to provide all the information required identifying and defining requirements, performing acceptance tests, and refining the structure and features of the system[6].
- 40-hour Week means that XP team members should not work more than 40 hours in the week. Therefore, requirements of each iteration should be selected carefully by an efficient way to make team members work without any need of overtime.
- Coding Standards means that XP team members must adopt an agreed set of coding rules that makes the understanding of code easier and facilitates the process of producing a consistent code.

2.2 XP Phases

XP life cycle includes six phases: exploration, planning, iterations to release, production, maintenance, and death. Figure 2 illustrates the phases of XP life cycle that will be explained in the following paragraphs[19]. Table 1 summarizes all XP phases and the activities required to achieve each phase.

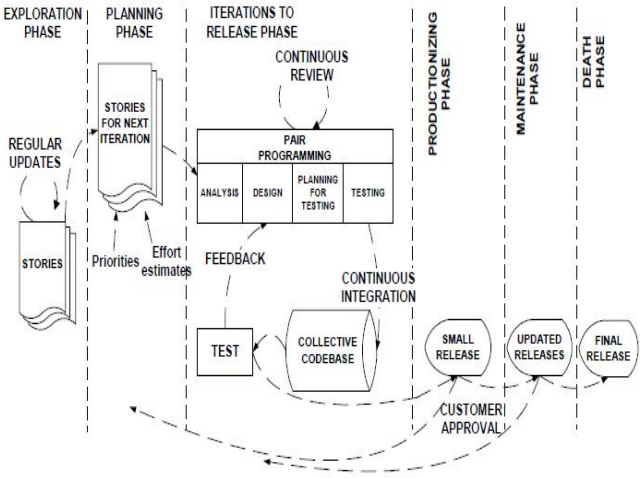

Figure 2. XP life cycle[19].

During the Exploration phase, the customers write the story cards to be included in the first release[1]. A user story is a software system requirement formulated as one or two sentences in the everyday or business language of the customers. This phase can be achieved through the following activities:

1. Ensuring that a preliminary background of the project is obtained (motivation, assumptions, constraints, addressed technology, acceptance criteria).

2. Clarifying the purpose of the story cards as a tool for collecting the requirements.
3. Clarifying the standards of writing the story cards (consistent, clear, testable, and integrated).
4. Writing the story cards by customers.
5. Ensuring that developers understand all story cards.
6. Analyzing and validating the story cards.

Table 1. Summary of XP phases and activities

Phase	Activities
Phase (1): The Exploration Phase	Ensuring that a preliminary background of the project is obtained (motivation, assumptions, constraints, addressed technology, acceptance criteria). Clarifying the purpose of the story cards as a tool for collecting the requirements. Clarifying the standards of writing the story cards (consistent, clear, testable, and integrated). Writing the story cards by customers. Ensuring that developers understand all story cards. Analyzing and validating the story cards.
Phase (2): The Planning Phase	Determining the priority order of the stories by the customers. Selecting the features that must be included in the current release. Negotiating the features of the current release. Estimating the effort and time that are required for each story. Proposing a schedule for the current release. Discussing the proposed schedule of the first release to reach to a final one.
Phase (3): Iterations to Release Phase	Breaking down the schedule to a number of iterations where every iteration takes one to four weeks. Selecting the smallest set of most valuable stories that make sense together and can be included in an iteration. Reviewing the functionality of all iterations. Selecting the iteration to be implemented. Writing the unit tests for the selected iteration. Writing the code for the selected iteration. Testing and integrating the selected iteration. Ensuring that all iterations scheduled are completed.
Phase (4): Productionizing Phase	Ensuring extra testing of the functionality and performance are done (system testing, load testing, installation testing). Determining new changes needed to be included in the current release. Implementing and testing the new changes. Documenting the postponed ideas and suggestions to implement them during maintenance phase or in next releases. Delivering the current running release to the customers.
Phase (5): Maintenance Phase	Documenting and analyzing the circumstances that led to bugs. Editing programs to fix bugs. Performing unit, system, and regression testing of the edited programs. Documenting and analyzing the causes of the system crash. Determining corrective instructions to prevent the system crash (terminate the on-line session, reinitialize the system, recover lost or corrupted databases, fix network problems, and fix hardware problems). Performing additional user training. Determining enhancement ideas and requests. Taking decisions about the enhancement ideas and requests that must be implemented in this phase or moved to the next releases.
Phase (6): The Death Phase	Ensuring that all predefined stories has been implemented. Finalizing all project documentation. Evaluating the quality of the current release and the related parts of the system. Determining the learned lessons from the project. Studying the feasibility of continuing the running of the release and the system.

During the Planning phase, the customers prioritize the stories and select the features to be in the first small release[12,20]. This phase can be achieved through the following activities:

1. Determining the priority order of the stories by the customers.
2. Selecting the features that must be included in the current release.

3. Negotiating the features of the current release.
4. Estimating the effort and time that are required for each story.
5. Proposing a schedule for the current release.
6. Discussing the proposed schedule of the first release to reach to a final one.

Iterations to Release phase indicates that several iterations of the software development are included before reaching the first release. The schedule is broken down to a number of iterations that will each take one to four weeks to be achieved[19]. XP promotes the concept of "small releases"[4]. This phase can be achieved through the following activities:

1. Breaking down the schedule to a number of iterations where every iteration takes one to four weeks.
2. Selecting the smallest set of most valuable stories that make sense together and can be included in iteration.
3. Reviewing the functionality of all iterations.
4. Selecting the iteration to be implemented.
5. Writing the unit tests for the selected iteration.
6. Writing the code for the selected iteration.
7. Testing and integrating the selected iteration.
8. Ensuring that all iterations scheduled are completed.

During product ionizing phase, there are more testing and checking of the functionality and performance of the system. This phase can be achieved through the following activities:

1. Ensuring extra testing of the functionality and performance are done (system testing, load testing, installation testing).
2. Determining new changes needed to be included in the current release.
3. Implementing and testing the new changes.
4. Documenting the postponed ideas and suggestions to implement them during maintenance phase or in next releases.
5. Delivering the current running release to the customers.

The Maintenance phase includes the efforts of customer support. Development stays in this phase until the system satisfies all customers' requirements[21]. This phase can be achieved through the following activities:

1. Documenting and analyzing the circumstances that led to bugs.
2. Editing programs to fix bugs.
3. Performing unit, system, and regression testing of the edited programs.
4. Documenting and analyzing the causes of the system crash.
5. Determining corrective instructions to prevent the system crash (terminate the on-line session, reinitialize the system, recover lost or corrupted databases, fix network problems, and fix hardware problems).
6. Performing additional user training.
7. Determining enhancement ideas and requests.
8. Taking decisions about the enhancement ideas and requests that must be implemented in this phase or moved to the next releases.

During the Death phase, the software development process has been finished and there is no change to architecture, design or code will be made. Death may occur if the system is not delivering the desired outcomes, or if it becomes non-feasible for further development. This phase can be achieved through the following activities:

1. Ensuring that all predefined stories has been implemented.
2. Finalizing all project documentation.
3. Evaluating the quality of the current release and the related parts of the system.
4. Determining the learned lessons from the project.
5. Studying the feasibility of continuing the running of the release and the system.

2.3 Methods of Project Tracing in Agile Projects

There are many attempts, studies, and researches conducted in the domain of tracing the progress of agile software projects, especially XP approach.

In[18], the researchers consider an optimal control model of extreme programming where user and developer efforts (the control variables) are optimally chosen during the development period to maximize net system value (system value minus user and developer effort). The researchers incorporate both demand side (user value) and supply side (developer and user effort) considerations in software development.

In[22], the researcher presents an approach of evaluating the degree to which various XP practices are implemented. For each XP practice, the researcher adapts Goal Quality Metric (GQM) method to elaborate its goal, questions, and metrics. The calculation of the elaborated metrics provides an efficient indicator regarding the degree of the implementation of XP practices. This approach doesn't focus on XP life cycle phases and activities.

In[23], the researchers present an improved version of XP to be applied to large scale projects with hundreds of software developers. The basic idea was to apply the "hierarchical approach", as a management principle of reorganizing companies, to organize XP project. The researchers discuss how the elements of the hierarchical approach can improve XP and how to scale up XP to very large projects.

In[24], the researchers investigate how teams adopt and use agile practices to help in moving to agile. They identified two methods for adopting agile in an organization, the big bang and gradual adoption. The researchers studied teams which adopted some or all agile practices. These teams committed to continuous assessment and improvement of their ways of working. The researchers concluded that teams prefer adapting agile-based, team-oriented practices suited to their particular needs over technical practices and defined methodologies.

In[25], the researchers present the effect of applying a disciplined project management method "PRINCE2" on the flexibility and agility of agile methods. So, the researchers resort to the findings of a real life project, where a team of eleven developers was able to deliver high quality software within budget and time limitations.

In[26], the researchers present a proposed method to estimate the optimal size, effort, and cost of XP projects. The researchers utilize the data of completed real projects. A tool is developed which takes the finished project as input and produce the output of the size, effort and cost of the project, which is more transparent and trustworthy to the client.

In[27], the researchers focused on team productivity in XP projects and provide a productivity model dedicated to this approach. The proposed model is developed based on the most significant features affecting team productivity. This model evaluated and gained enough acceptance. The researchers also show the most effective XP practices that increase team productivity in XP projects.

In[28], the researchers present that the adoption of the human-centered methodology "Design Thinking" (DT) leads to creativity and innovation. The researchers integrate DT practices into XP approach to improve the quality of software products for the end-users and enable businesses to achieve creativity and innovation. The proposed integrated framework presents the various DT practices (empathy, define, persona, DT user stories) that were adopted into XP phases.

In[29], the researchers introduce a proposed effort estimation model for agile software projects. The proposed model is divided into three parts; the first part is dedicated to estimating the project velocity, the second part dedicated to estimating the story size, and the third part is dedicated to calculating the contingency allowance using COCOMO II factors. The results evaluated via Magnitude of Relative Error (MRE) and Prediction Level (PRED) metrics. The results showed that the accuracy of the proposed model is superior to agile story point model. This study focused only on one activity in agile life cycle phases.

Most of the previous attempts in the domain of tracing the progress of XP projects are not enough because they don't introduce a comprehensive guide that includes all the phases and activities of achieving XP project. Therefore, this study tries to solve this problem by introducing a proposed approach that can be used to trace the progress of XP projects.

3. The Proposed Approach

Depending on a combination of simple statistical techniques and the phases and activities of XP projects, the study introduces a proposed approach that can be used to trace the progress of XP projects. The proposed approach depends on the following main steps:

1. Performing the project phase.
2. Evaluating the actual achievement of the project phase.
3. Comparing the actual achievement of the phase with the accepted level.

3.1 Performing the Project Phase

This step means that XP team members perform the current project phase using the predefined activities that were listed in Table 1. XP team members review the steps required for each phase. One of the team members must ensure these steps are clear to all the team members. In addition, she/he must trace the progress of the project to

help in redirecting the team to the right way of performance.

3.2 Step (2): Evaluating the Actual Achievement of the Project Phase

This step means that XP team members evaluate the actual achievement of the project phase. This phase depends on two statistical techniques which are the rating scale and the average. The activities of each phase are organized in the form of a checklist with 5-rating scale which expresses the level of actual achievement of each step. The 5-rating scale includes the possible responses: "Very Good", "Good", "Neutral", "Poor", and "Very Poor" which corresponding to 5,4,3,2, and 1 respectively. The usage of the rating scale is an attempt to convert the qualitative answers to quantitative values.

Table 2 illustrates a representation of the Exploration Phase using the rating scale with some actual answers. The actual achievement (AC) of each phase can be measured by calculating the average of the activities' answers as shown in equation 1[30].

$$AC = \frac{S}{N} \quad (1)$$

From Equation 1, S represents the sum of quantitative answers and N represents the number of activities (answers). Upon the answers in Table 2:

N = 6

S = 4 + 5 + 3 + 4 + 3 + 1 = 20

AC= S/N= 20/6 = 3.33 out of 5

The actual achievement of the Exploration Phase = 3.33 out of 5 = 66.6 %

3.3 Comparing the Actual Achievement of the Phase with the Accepted Level

This step means that XP team members compare the actual achievement of the phase (AC) with the accepted level (AL). The accepted level of achievement is determined by the software organization and XP team members prior to starting of the work on the projects. Sometimes this level determined differently depending on the domain of the project. The comparison has two cases:

- If AC < AL , then take corrective actions to correct mistakes in the performance of the phase and perform step 2 to reevaluating the updated achievement of the current phase
- Else (the actual achievement is greater than or equal to the accepted level) move to the next phase and perform step 1

Upon the accepted level of achievement is 80% and the results of the previous calculations, the researcher found that:

Table 2. The exploration phase and rating scale

Activities	Very Good (5)	Good (4)	Neutral (3)	Poor (2)	Very Poor (1)
Ensuring that a preliminary background of the project is obtained (motivation, assumptions, constraints, addressed technology, acceptance criteria).		√			
Clarifying the purpose of the story cards as a tool for collecting the requirements.	√				
Clarifying the standards of writing the story cards (consistent, clear, testable, and integrated).			√		
Writing the story cards by customers.		√			
Ensuring that developers understand all story cards.			√		
Analyzing and validating the story cards.					√

- The actual achievement of the Exploration Phase = 3.33 out of 5 = 66.6 %
- The accepted level of achievement is 80%
- The actual achievement is less than the accepted level. Therefore, take corrective actions to correct mistakes in the performance of the phase. Then, perform step 2 to reevaluating the updated achievement of the current phase

4. Applying the Proposed Approach and a Discussion

The proposed approach was applied on two XP projects. The first project was in the domain of higher educational institutions. The second project was in the domain of construction.

4.1 Project (1)

The first project was a graduation project in a higher educational institute. The basic data about the project are summarized in Table 3. The results of achievement and tracing this project can be summarized in Table 4. The column "Deviation" is the difference between the actual achievement and the accepted level of achievement. The negative values present a shortage in performance while the positive values present a good performance. Figure 3 illustrates a graphical representation of the actual achievement for all phases in project (1).

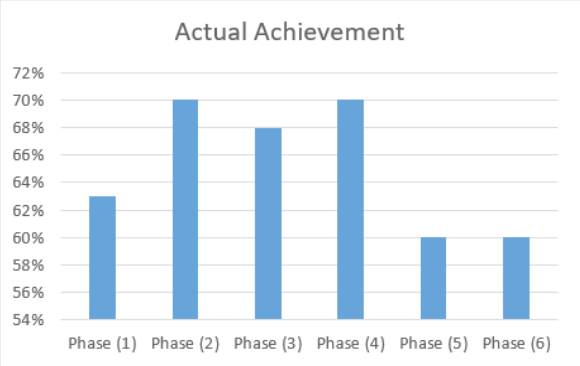

Figure 3. The actual achievements of project (1).

4.2 Project (2)

The second project was a software application in the construction domain. The basic data about the project are summarized in Table 5.The results of achievement and tracing this project can be summarized in Table 6. Figure 4 illustrates a graphical representation of the actual achievement for all phases in project (2).

Table 3. The basic data of project (1)

Data Item	Description
Scope	To develop a software application to manage the training department in AAI institute.
Duration	7 months (3 months in the first term and 4 months in the second term)
Cost	There is no funding for this project because it was a graduation project.
Team size	5 students
Accepted level of achievement for each phase	70 %
Number of iterations	4 iterations each one was one month duration

Table 4. The results of achievement of project (1)

Phase	Actual Achievement	Deviation =(Actual-Accepted)
Phase (1): Exploration	63%	-7%
Phase (2): Planning	70%	0%
Phase (3): Iterations to Release		
Iteration (1)	**55%**	**-15%**
Iteration (2)	**70%**	**0%**
Iteration (3)	**72%**	**+2%**
Iteration (4)	**75%**	**+5%**
The average of phase (3)	**68%**	**-2%**
Phase (4): Production	70%	0%
Phase (5): Maintenance	60%	-10%
Phase (6): Death	60%	-10%

4.3 Discussion

The actual achievement of the phases of project (1) increased after the first phase because of the knowledge gained by the team members. In the same project, the actual achievement of last two phases was less than the accepted level. Because of the main focus of the team members was low after phase (3) which is the main phase of implementation. The results agree with the nature of the

graduation projects. The actual achievement of the phases of project (2) increased after the first phase and continued in the same trend. Generally, the actual achievement of the phases related to project (2) is significantly higher than the values of project (1) as shown in Figure 5.

Table 5. The basic data of project (2)

Data Item	Description
Scope	To develop a software application to manage the procurement department in XYZ construction company.
Duration	6 months
Cost	22000 $
Team size	8 (6 Professional and 2 Juniors)
Accepted level of achievement for each phase	85 %
Number of iterations	7 iterations each one was two weeks duration

Table 6. The results of achievement of project (2)

Phase	Actual Achievement	Deviation =(Actual-Accepted)
Phase (1): Exploration	80%	-5%
Phase (2): Planning	82%	-3%
Phase (3): Iterations to Release		
Iteration (1)	**85%**	**0%**
Iteration (2)	**87%**	**+2%**
Iteration (3)	**85%**	**0%**
Iteration (4)	**90%**	**+5%**
Iteration (5)	**84%**	**-1%**
Iteration (6)	**85%**	**0%**
Iteration (7)	**90%**	**+5%**
The average of phase (3)	**86.6%**	**+1.6%**
Phase (4): Production	88%	+3%
Phase (5): Maintenance	95%	+10%
Phase (6): Death	92%	+7%

The software organizations or XP teams can adopt the proposed approach for tracing the progress of the projects and revealing any weaknesses or problems to solve them at a suitable time. Therefore, the proposed approach can be used to identify the performance gap which is the difference between the actual achievement and the accepted level of achievement. Consequently, the team members can highlight the factors that lead to this gap[31].When the team members become more familiar with the proposed approach, the adoption will be more useful. The accumulated experience of the team members in applying XP practices and activities enhance their maturity across the time.

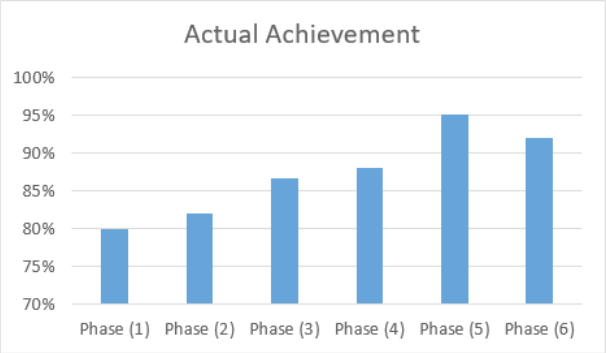

Figure 4. The actual achievements of project (2).

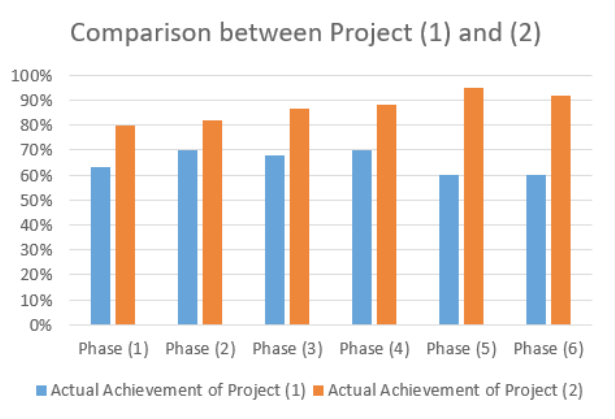

Figure 5. Comparison between project (1) and project (2).

5. Conclusion

Tracing the progress of software projects is one of the most crucial success factors of these projects. This study aimed to utilize the XP life cycle phases and activities and simple statistical techniques to propose an approach for tracing the progress of XP software projects. The proposed approach starts by performing the current project phase using the predefined activities then evaluating the actual achievement of the current phase. Then, comparing the actual achievement of the phase with the accepted level.

Software organizations can use the proposed approach to track, evaluate, control, and enhance the performance of XP software projects to increase the success rate of software projects. In addition, the proposed approach can serve as a base for building a software tool to trace the progress of agile software projects.

6. Future Work

In the domain of the progress of XP projects and tracing implementation of its phases and activities, there are many issues that can be targeted in the future:

- Utilizing the proposed approach for building a software tool to trace the progress of agile software projects.
- Expanding the work to include other agile methods such as Scrum.
- Extracting the metrics that can be used for measuring the performance of each phase in XP projects.
- Expanding the work to include human factors related to XP team that have the critical effect on the success of XP projects.

7. References

1. Yang Y, Bosheng Z. Evaluating Extreme Programming Effect through System Dynamics Modeling. Proceedings of International Conference on Computational Intelligence and Software Engineering (CiSE). 2009; 1–4. https://doi.org/10.1109/CISE.2009.5365556.
2. Flora H, Chande S. A Systematic Study on Agile Software Development Methodologies and Practices. Int. j. comput. sci. inf. technol. 2014; 5(3): 3626–3637.
3. Cohen D, Lindvall M, Costa P. An Introduction to Agile Methods. Advances in computers. 2004; 62: 20–22. https://doi.org/10.1016/S0065-2458(03)62001-2.
4. Agile Software Construction [internet]. https://www.springer.com/gp/book/9781852339449. Date accessed: 2006.
5. Best Practices for Implementing Agile Methods: A Guide for Department of Defence Software Developers [internet]. http://www.businessofgovernment.org/report/best-practices-implementing-agile-methods-guide-department-defense-software-developers. Date accessed: 2008.
6. Scaling Software Agility: Best Practices for Large Enterprises [internet]. https://www.amazon.com/Scaling-Software-Agility-Practices-Enterprises/dp/0321458192. Date accessed: 08/03/2007.
7. Extreme Programming Explained: Embrace Change. https://www.amazon.in/Extreme-Programming-Explained-Embrace-Change/dp/0321278658. Date accessed: 16/11/2004.
8. Nortier B, Von Leipzig K, Schutte C. The Development of a Software Development Framework by Combining Traditional & Agile Methods to Address Modern Challenges. Proceedings of ISEM 2011. 2011; 1–18.
9. Extreme Programming in Perl [internet]. https://www.extremeperl.org/f/extremeperl.pdf. Date accessed: 2004.
10. Sharp H, Robinson H. Collaboration and co-ordination in mature eXtreme programming teams. INT J HUM-COMPUT ST. 2008; 66(7): 506–518. https://doi.org/10.1016/j.ijhcs.2007.10.004.
11. Karlheinz K, Sabine Z. Just Enough Structure at the Edge of Chaos: Agile Information System Development in Practice. Proceedings of 9th International Conference: XP2008. 2008; 137–146. https://doi.org/10.1007/978-3-540-68255-4_14.
12. Wallace N, Bailey P, Ashworth N. Managing XP with Multiple or Remote Customers. Proceedings of 3rdInternational Conference on eXtreme Programming and Agile Processes in Software Engineering: XP2002. 2002; 134–137.
13. Williams L, Kessler R, Cunningham W, Jeffries R. Strengthening the Case for Pair Programming. IEEE Software. 2000; 17: 19–25. https://doi.org/10.1109/52.854064.
14. Angioni M, Carboni D, Pinna S, Sanna R, Serra N, Soro A. Integrating XP Project Management in Development Environments. J SYST ARCHITECT. 2006; 52(11): 619–626. https://doi.org/10.1016/j.sysarc.2006.06.006.
15. Agile Software Development: Evaluating the Methods for Your Organization [internet]. https://www.mobt3ath.com/uplode/book/book-48346.pdf. Date accessed: 2005.
16. Concas G, DiFrancesco M, Marchesi M, Quaresima R, Pinna S. An Agile Development Process and Its Assessment Using Quantitative Object-Oriented Metrics. Proceedings of 9thInternational Conference: XP2008. 2008; 83–93. https://doi.org/10.1007/978-3-540-68255-4_9.
17. Rittenbruch M, McEwan G, Ward N, Mansfield T, Bartenstein D. Extreme Participation: Moving Extreme Programming Towards Participatory Design. Proceedings of the Participatory Design Conference.2002; 29–41.
18. KumarS, Susarla A, Mookerjee V. Coordinating User-Developer Efforts in Extreme Programming: A Control-Theoretic Approach. Proceedings of the 15thAnnual Workshop on Information Technologies and Systems (WITS). 2005; 189–194. https://doi.org/10.2139/ssrn.885435.
19. Agile Software Development Methods: Review and Analysis [internet]. https://arxiv.org/ftp/arxiv/papers/1709/1709.08439.pdf. Date accessed: 2002.
20. Hildenbrand T, Geisser M, Kude T, Bruch D, Acker T. Agile Methodologies for Distributed Collaborative

Development of Enterprise Applications. Proceedings of the International Conference on Complex, Intelligent and Software Intensive Systems. 2008; 540–545. https://doi.org/10.1109/CISIS.2008.105.

21. Usha K, Poonguzhali N, Kavitha E. A Quantitative Approach for Evaluating the Effectiveness of Refactoring in Software Development Process. Proceedings of the International Conference on Methods and Models in Computer Science. 2009; 1–7. https://doi.org/10.1109/ICM2CS.2009.5397935.
22. Darwish N. Towards an Approach for Evaluating the Implementation of Extreme Programming Practices. International Journal of Intelligent Computing and Information Sciences (IJICIS). 2013; 13(3): 55–67.
23. Rumpe B, Scholz P. A Manager's View on Large Scale XP Projects. Proceedings of 3rdInternational Conference on Extreme Programming and Flexible Processes in Software Engineering: XP2002. 2002; 158–159.
24. Brendan J, James N, Craig A. Agile Practices in Practice: Towards a Theory of Agile Adoption and Process Evolution. Proceedings of 20thInternational Conference on Agile Processes in Software Engineering and Extreme Programming: XP2019. 2019; 3–18. https://doi.org/10.1007/978-3-030-19034-7_1.
25. Alnoukari M, Alzoabi Z, Elshiek A. Introducing Discipline to XP: Applying PRINCE2 on XP Projects. Proceedings of IADIS Multi Conference on Computer Science and Information Systems (MCCMIS), Portugal, 2009; 51–58. https://doi.org/10.1109/ICTTA.2008.4530347.
26. Karunakaran E, Sreenath N. A Method to Effort Estimation for XP Projects in Clients Perspective. Int. J. Appl. Eng. 2015; 10(7): 18529–18550.
27. Tavakoli F, Gandomani T. A Novel Team Productivity Model for XP Teams. Journal of Cases on Information Technology. 2018; 20(4): 93–109. https://doi.org/10.4018/JCIT.2018100106.
28. Sohaib O, Solanki H, Dhaliwa N, Hussain W, Asif M. Integrating Design Thinking into eXtreme Programming. J AMB INTEL HUM COMP. 2019; 10(6): 2485–2492. https://doi.org/10.1007/s12652-018-0932-y.
29. Raslan A, Darwish N, Hefny H. A Proposed Contingency Model for Effort Estimation of Agile Projects. JOKULL Journal. 2017; 67(10): 86–95.
30. Arithmetic Mean (Average) – GMAT Math Study Guide [internet]. http://www.platinumgmat.com/gmat_study_guide/statistics_mean. Date accessed: 2019.
31. How Gap Analysis Can Improve Your Project Management [internet]. https://www.projectmanager.com/blog/gap-analysis-project-management. Date accessed: 08/01/2019.