A Closer Look at Extreme Programming (XP) with an Onsite-Offshore Model to Develop Software Projects Using XP Methodology

Ponmurugarajan S. Thiyagarajan and Sachal Verma

Tata Consultancy Services
Rajan.st@tcs.com
Sachal.Verma@tcs.com

Abstract. The business world of today is highly competitive. Business users demand IT organizations to adapt quickly to changes and provide on-time, cost-effective solutions. This compels companies to look closely at their software development processes, to improve them and remain cost-effective. Offshoring is a well-known cost-effective solution for projects that follow waterfall and other traditional software development life cycles (SDLC). Waterfall SDLC may not be ideal when requirements are changing rapidly. Achieving rapidness in software development along with offshoring will enable companies to provide quick and cost effective IT solutions. To manage rapidly changing requirements, a large telecommunications company moved out of traditional waterfall model and adopted Extreme programming (XP) software development methodology. This paper discusses, in detail, a Telecommunication software project case study along with the customized XP onsite-offshore model that was successfully used in developing the project. This paper also share the lessons learnt from this XP onsite-offshore model.

Keywords: Extreme Programming, XP, Offshore, Model, Telecommunication.

1 Introduction

In today's world, business changes are very rapid due to intense competition and frequent introduction of new products to the market. These changes to business requirements trigger changes to traceable IT requirements. Managing rapidly changing business and IT requirements and developing durable and adaptable software may be challenging and costly. Traditional models like waterfall software development methodology may not even fit or be efficient in such situations. A move towards agile software development techniques may be necessary. Agile software development techniques, like Extreme Programming or XP (Kent Beck, 2000), can adjust to rapidly changing requirements and help refactor the software accordingly. Extreme Programming an agile software development methodology, is a predictable way to develop high quality software with minimal risk in the short term and the long term, which the customers will like. A reason for this liking is customer presence and close contact with the team through out the tenure of the project.

K. Berkling et al. (Eds.): SEAFOOD 2008, LNBIP 16, pp. 166–180, 2009.

Offshoring has now become a common option for many organizations to develop and maintain their cost effective software. Extreme Programming, developed by Kent Beck, is born out of the desire to apply best in class software practices in a disciplined and reproducible way. To perform disciplined software development using XP methodology involving onsite and offshore teams may require suitable customization of XP methodology. There are efforts by researchers (Samantha Butler et. al., 2003), but not complete study, to investigate the effectiveness of global software development using XP.

This paper is mainly organized as follows: Section 2 provides an overview of XP and compares it with other traditional models. Section 3 details a case study with an onsite-offshore model developed and successfully completed for a large Telecommunication company based in the United States. Here, the authors also compare traditional and customized onsite-offshore XP practices. Lessons learnt from this project are also listed. Section 4 describes conclusions.

2 Overview of XP

XP is a deliberate and disciplined approach of software development. It enables users to get reliable, working software quickly and continue development at a rapid, confident, and predictable pace with ever-increasing quality. It is designed to suite environments where requirements are rapidly changing and scope is unclear. It emphasizes on customer satisfaction and teamwork. The following sub-sections discuss some of the important XP concepts.

2.1 XP Values

Key XP values (Chromatic, 2003) are *simplicity, communication, testing and courage*. XP requires communication to be simple with involvement of onsite business customer during all software development phases. XP encourages good communication so that business people do not promise the unachievable and technical people do not achieve the unwarranted. It involves starting with simple solutions so that more complex functionalities shall be added later. Requirements are tweaked into simple and clear user stories for better understanding. Feedback is another key value of XP. Feedback from customers, teams and systems are ensured in all phases of development. Unit tests ensure feedback from systems. Regular presence of customers ensures continuous feedback from them. Daily stand-up calls and pair-programming ensures continuous feedback from teams. Feedback from customers is obtained at all phases to perform necessary refactoring. Courage is another key XP value. XP lets customer drive the project courageously. XP based software development is in small and regular cycles involving frequent evaluations. It practices pair-programming and encourages team to sit together so that everybody could see what each one is capable of doing.

2.2 XP and Traditional Models Compared

Traditional way to software development usually have these:

i. Months of meetings with customers before and during the project startup phase
ii. Generate requirements, specification documents, use cases etc.

iii. Customers negotiate a release date
iv. Developers design, code and test
v. Customer performs User Acceptance Testing (UAT)
vi. Often pieces and requirements are missed
vii. Often whole process takes longer than expected

In an XP way,

i. Customer interacts with XP team regularly during development
ii. Customer writes requirements which are broken down into clear and crisp user stories
iii. Developer estimate user stories
iv. Stories are grouped into releases comprising of various iterations
v. Customer provides feedback during development and reviews outputs at the end of each iteration
vi. Customer writes acceptance tests
vii. Developers test, code and refactor
viii. Customer controls team's direction

Following diagram (Figure 1) compares traditional methods like Waterfall and Iterative development with XP. You can see that XP takes small and simple steps to achieve the target whereas Waterfall and Iterative development progresses in relatively large steps and in phases. In a waterfall model, any change or issue discovered at a later point of the project will badly impact project completion and cost.

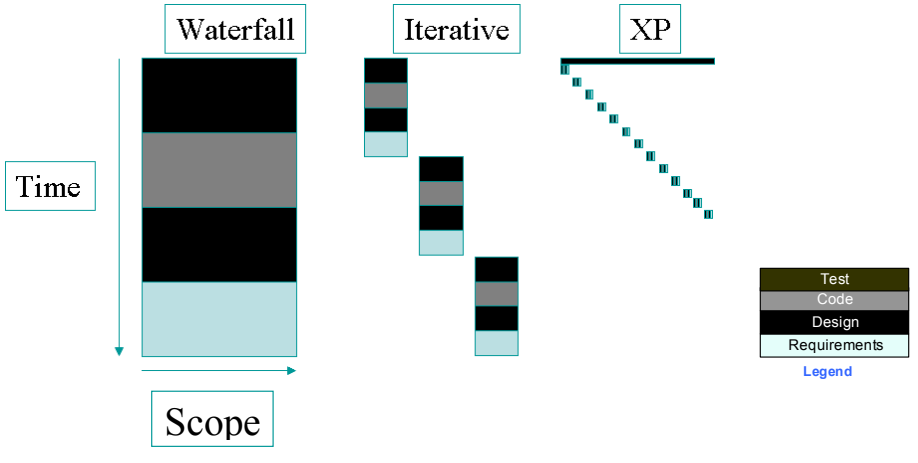

Fig. 1. XP and Traditional Methods – Compared

The following diagram (Figure 2) compare the development phases in Waterfall mapped to XP. You may note that a release of software in XP methodology consist of several iterations.

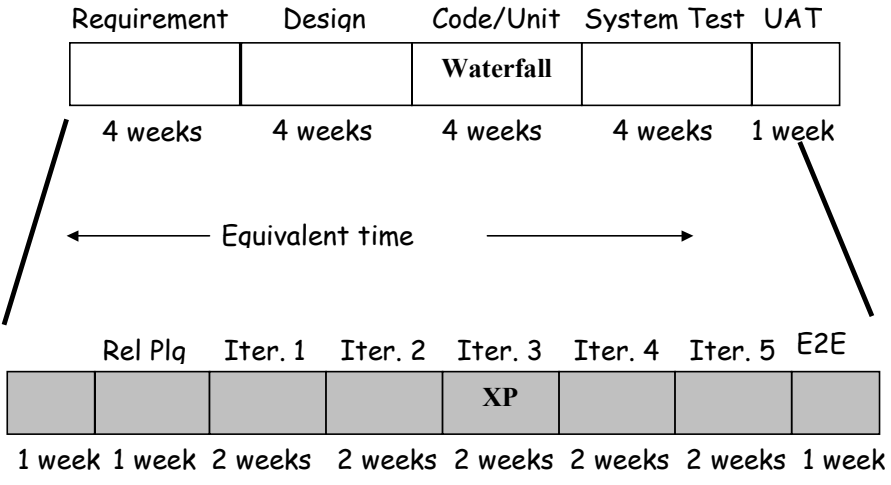

Fig. 2. XP and Waterfall - Compare Releases

2.3 Typical XP Process

The following diagram (Figure 3) illustrates the End-to-End XP Process (Donovan Wells, 1999).

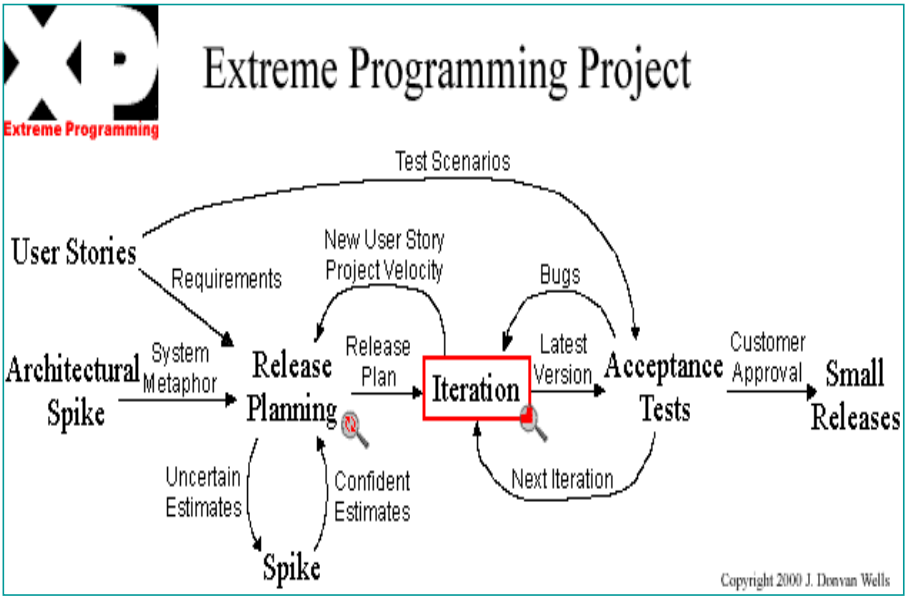

Fig. 3. XP Process

An XP project consists of a series of releases, approximately 1 to 3 months long, each providing some business value. A release consists of number of iterations that are approximately 1 to 3 weeks long. Each iteration consists of stories, which, in turn, are made up of tasks. A task is executed in test and functional code. During release planning, the customer chooses the stories the customer wants in that release. Stories are selected for each iteration based on the customer preference and available user velocity. Stories are spiked or split based on its complexity. Developers estimate effort for each user story. Velocity is a measure of capacity or the effort each team member can put for the iteration. Task assignments are based on available velocity of the team member. Unit tests are written to completely test the user story. Coding and testing take place as pair programming. Once coding and testing is completed, automated integrations tests are run to ensure correctness and completeness of the development planned for that iteration. At the end of each iteration, customer performs user acceptance tests and once signed-off is implemented in production through a release.

In the next section, the authors explain a case study and discuss a successfully implemented onsite-offshore model for an XP based development project for a Telecommunications company.

3 A Case Study

3.1 The Telco Project

A major Telecommunications provider wanted to consolidate its three legacy customer account management (CAM) applications into a single regional Java based application. Due to mergers and acquisitions, which happened years before, this telecommunication provider had to maintain 3 different legacy applications to perform customer account management functions. These 3 legacy applications ran in mainframe environment. Software enhancement and maintenance of these legacy applications were costly and time consuming, and, demanded consolidation.

The team members of these legacy mainframe applications were located in various locations involving onsite and offshore. Onsite teams were located in Denver, Seattle and Omaha whereas offshore team was in Chennai, India. Onsite teams comprised of business subject matter experts and technical leads which included Telco's employees and onsite consultants. Onsite consultants, from a contracting company, reporting to the onsite Project Manager, interacted and shared work with their offshore consultants (of the same contracting company). Offshore teams primarily consisted of a Project Manager, programmers and testers.

This was the time, when the Telco made the decision to move out of Waterfall SDLC to XP based software development. With well-known benefits of XP (Gittins et. al., 2001 and 2002), management was committed to propagate and adhere to the new XP methodology and use it for the new CAM project. To avail the best benefits of XP, the team should follow its defined, deliberate and disciplined process.

The proposed consolidated CAM application was designed to be based on J2EE architecture. The new Java based application was planned to be developed using an open source based integrated development environment (IDE), Eclipse (Eclipse Platform Review, 2003). Following is the comparison (Table 1) of the different technologies involved with legacy and new CAM applications.

Table 1. Technologies Compared

	Legacy CAM applications	**New Consolidated CAM application (XP)**
Operating System	z/OS mainframe, MVS	Linux
Programming Languages	COBOL, Assembler, JCL	Java
Configuration management	Endevor	PVCS Dimensions
Database	DB2, IMS-DB	Oracle
Other Software	Viasoft, Rexx	J2EE Technologies, Hibernate, TIBCO

3.2 Onsite-Offshore Model

3.2.1 Challenges

XP is ideal for teams working at a single location that can ensure face to face communication. With the current team distribution (3 onsite location and 1 offshore location), it was impossible to bring the team in one location. These factors triggered customization of the XP methodology and practices for the project and the need to enhance the existing onsite-offshore model. The main challenges were to manage teams located in several geographical locations and working in different time zones. Adding to these challenges, lack of Java skilled programmers was another issue. Telco's legacy applications SMEs (Subject Matter Experts) were experts in mainframe and did not possess necessary Java/J2EE skills.

3.2.2 Planning and Solutions

Offshore consultant team was ready to ramp up trained and skilled Java programmers. Telco's legacy SMEs were trained in Java technologies for a month's time period before the project startup. This, to some extent, solved the issue of shortage of Java skilled programmers. Next challenge was to address the team distribution at onsite and offshore. Onsite teams were scattered at various US locations - Denver, Omaha and Seattle. Offshore team was located in Chennai. XP process, ideally, demand teams to sitting face to face at the same location (as shown below in Figure 4). Following diagram is an illustration of a typical XP '*pod',* XP development area.

Pods were created or modified, as necessary, at all 3 onsite locations so that respective teams can ensure face-to-face communication at that location. Team members have to perform XP development work only from their pods. Pods were equipped with telephones, personal computers with web conferencing facilities and whiteboards to ensure effective communication as demanded by XP. Now that infrastructure and other set-up are planned and ready, let us discuss the onsite-offshore XP model to develop the software project.

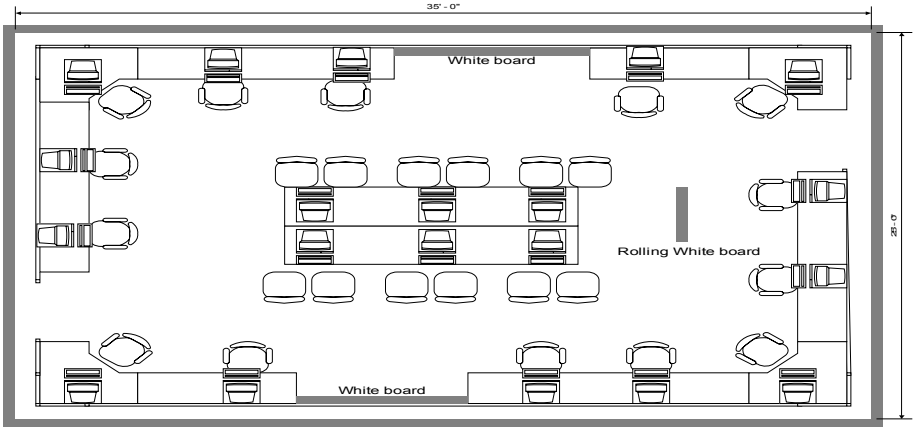

Fig. 4. XP Pod

3.2.3 The Model

The following diagram (Figure 5) represents the onsite-offshore model customized to work in XP methodology.

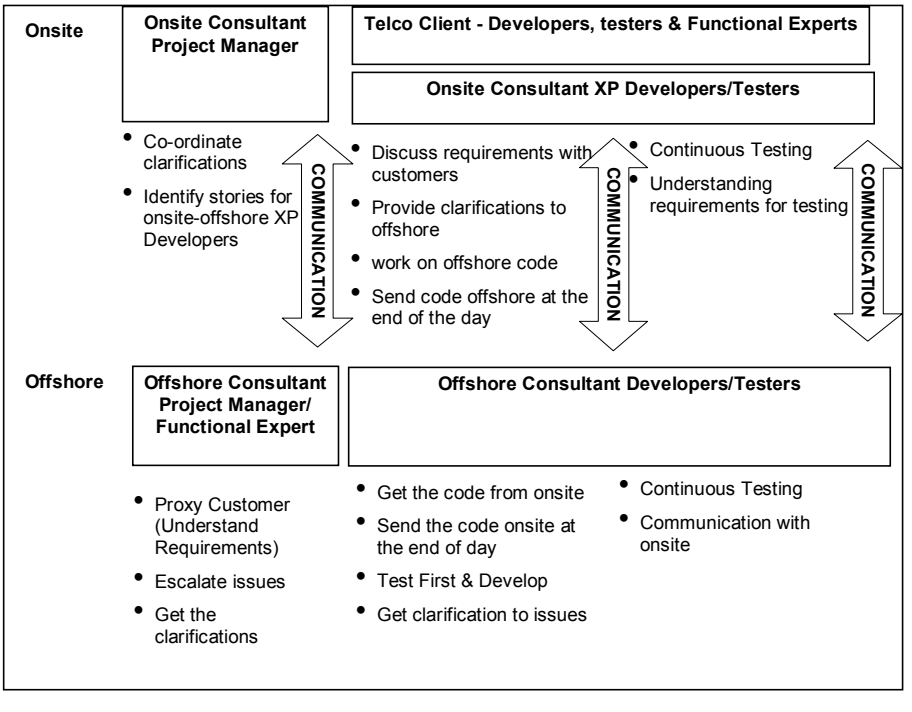

Fig. 5. Onsite-Offshore XP Model

Onsite and Offshore Project Managers: Effective Communication is ensured between onsite and offshore teams. For this, the onsite consultant Project Manager (PM) interacts with offshore consultant PM on a daily basis. Onsite PM provide necessary clarifications on user stories and work with the offshore PM to identify the stories for onsite and offshore developers. Onsite and Offshore PMs participate in daily stand-up calls and identify the user stories based on the velocity of onsite and offshore consultant teams.

Proxy Customer: Offshore PM acts as a proxy customer, during the offshore day, for offshore team members. The offshore PM, the proxy customer, provides clarifications to questions raised by offshore team. Questions unanswerable by the offshore PM are discussed in the next day stand-up call with onsite.

The following diagram (Figure 6) explains the XP based software development workflow.

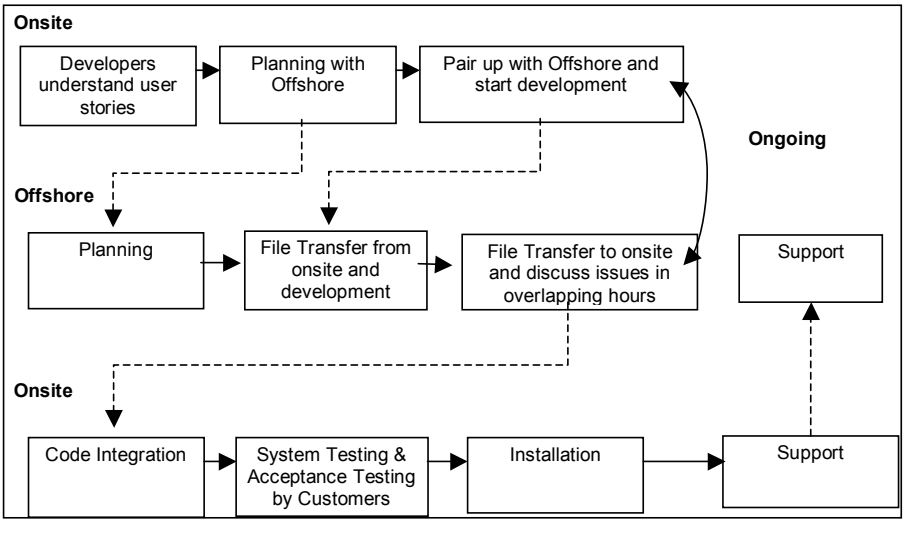

Fig. 6. Onsite-Offshore workflow

Offshore Representation: Offshore consultants work with their corresponding onsite team. To avoid confusion and reduce ambiguity, onsite consultant teams regularly discuss during onsite meetings about the user story tasks, for both onsite and offshore teams, to be performed that day. They discuss the questions and concerns rose, if any, by offshore team and obtain necessary clarifications from customer and functional experts.

Overlapping Work Hours: Onsite team then pairs up with offshore team during the planned overlapping work hours between onsite and offshore. Overlapping hours are

defined well ahead. These overlap hours can change on a weekly basis based on the availability and requirement of onsite and offshore consultants. After the overlapping work window, onsite team pair up with onsite Telco teams and continue work on user stories.

Pair-Programming: Communication, via email, is sent to offshore asking them to continue with the user story tasks. Offshore team, then, pair up with other offshore team members to continue working on the user story. At the end of their days, onsite and offshore team check-in code into PVCS Dimensions and ensure that they did not break each others code.

Collective Ownership: Onsite and offshore team work together on each user story. During the day, onsite team works on the user story and sends the code to offshore at the end of the day. Both onsite and offshore teams follow test first and develop methodology. They emphasize on continuous integration testing to ensure that a change does not break any other part of the code. If, in case, either onsite or offshore breaks the code at the end of the day, the work done for the day is scrapped and the same details are communicated via emails.

Documentation: XP encourages minimum documentation. With onsite-offshore model, documentation effort is higher than what is normally required in an XP project. For the CAM project, Javadocs were created and this served as a reference document for the code. Automated tests and JUNITs reports served as test documentation. Transition documents were created to train new team members.

3.2.4 Distributed Onsite-Onsite Communication

It was a minor challenge to manage onsite-onsite communication as the teams were located in different geographical locations and time zones within the United States – Omaha, Seattle and Denver. Onsite time zone difference across regions was 1 or 2 hours depending on the location. Team members were asked to adjust their work hours such that varying time zone issue can be minimized. This had a slight impact on pair programming involving pairs from 2 different locations. During necessary situations, the start and end of the day, pairs were formed from single location. Daily stand-up calls were conducted in the morning time when team members from all 3 locations were available.

The following sub-section discusses about the customization of XP practices in this onsite-offshore model context.

3.3 Customized XP Practices

Below sub-sections describe and compare the traditional XP practices with customized onsite-offshore XP practices for CAM project.

3.3.1 Practices That Regulate Planning

Table 2. Compare XP Practices that regulate planning

XP Practices	Traditional XP	Customized Onsite-Offshore XP
Release Planning	The team plans the content of the release. A release comprises of one or more iterations.	The team plans the content of the release that is 90 days in length. A release comprises of one or more iterations with 2 weeks iterations.
Iterations	The team plans and periodically releases software. Iterations consist of completed stories.	The team plans and releases software internally on a 2 weeks cycle. Iterations mainly consist of completed stories. If bugs exist or customer not happy, then incomplete stories will be carried over to future iterations
Small Releases	Releases are implemented as soon as there is enough system functionality to add business value to the customer.	Releases, that are 90 days long, are implemented as soon as there is enough system functionality to add business value to the customer.

3.3.2 Practices That Regulate Social and Technical Relationships in the Technical Team

Table 3. Compare XP Practices that regulate social and technical relationships

XP Practices	Traditional XP	Customized Onsite-Offshore XP
Collective Ownership	Any pair can improve any line of code, anywhere in the system, at any time.	Offshore PM or another representative, participates in the daily stand-up calls for discussions. Pair programming involve combinations of onsite and offshore pairs. Code is checked into a common software configuration management tool, PVCS Dimensions, which can be accessed from onsite as well as offshore.

Table 3. (*Continued*)

Simple Design	The simplest thing that could possible work to make the unit test pass, in the context of an overall system architecture that supports the requirements. Refactor as necessary.	Design is kept simple with agreement between onsite and offshore teams. Design information is accessible at a common folder location and can be accessed through a configuration management tool.
Coding Standards	Developers and testers write all code in accordance to predefined rules that enhance communication through the code	Java coding standards are devised and shared with onsite-offshore teams and the same is ensured during pair programming involving onsite and offshore pairs. A Java based IDE; Eclipse is used by both onsite and offshore teams. Software packages, components conventions and standards are followed across the board.
Refactoring	Design changes, no matter how sweeping, take place in small, safe steps	Pair programming involving onsite and offshore ensure necessary refactoring for simplification of code and design.

3.3.3 Practices That Help to Assure Quality Software

Table 4. Compare XP Practices that assure quality

XP Practices	Traditional XP	Customized Onsite-Offshore XP
Pair Programming	All code is written (including acceptance tests) with two people at one machine	Pair programming involving onsite and offshore team member combinations is achieved through web conferencing (for example, Microsoft Net Meeting) and telephone conferencing. Pair programming also happen at '*pods*' at respective offshore and onsite locations. In few scenarios, triplets of programmers (combinations of onsite and offshore) were

Table 4. (*Continued*)

		present during programming for better understanding of business and technology. Triplets, for example, were formed between an onsite Subject Matter Expert, onsite Technical Lead and offshre programmer. A common overlapping work period is devised and followed for onsite-offshore pair programming.
Test-first Development	New code is written or existing code is refactored only after a unit test has been created and verified to pass/fail	Unit tests are written and stored in PVCS Dimension which could be accessed via the Eclipse IDE by both onsite and offshore teams. JUNITs are written for performing unit tests. Tests are automated to the best possible extent.
Continuous Integration	Code is checked into a central repository and the entire system is checked out and built from scratch AND passes all unit tests 100%. Unit tests automated	A common configuration management tool (PVCS Dimensions) is used for software and document management. This tool is accessible by both onsite an offshore. This helped in version control. Unit tests are also checked into the configuration management tool. Daily builds were ensured at the end of the respective end of the days by the onsite and offshore PMs.
Acceptance Tests	A test is defined by the customer to accept the story	Acceptance tests are only performed by customer with support from onsite team.

3.4 Lessons Learnt

Listed below are some of the key lessons learnt from this project, which were experienced in the customized XP onsite-offshore model based project.

- **Offshore suitability** should be evaluated. Following types of projects are suggested to best suitable:
  - Projects with longer/more iterations. Projects that have longer duration are ideal.
  - Projects to be developed from scratch. Brand new development projects.
  - Projects previously developed projects from offshore. Repetitive or streamline type of projects are best suited to work on the onsite-offshore model.
  - Project with minimal dependencies with other applications. More number of interfacing applications makes the project complex and possibility of missing functions and requirements related to interfaces.
- **Functional Experts** must be identified at offshore; who understand business and can provide clarification to offshore developers. Before project startup, offshore functional experts must travel to onsite and obtain a mandatory to provide knowledge transition about the project. Large XP projects often start with a lockdown session. Functional experts should ensure mandatory participation in these lockdown planning sessions.
- **Communication** must be effective and efficient. Use of Teleconference, WebEx and NetMeeting tools must be encouraged. Must have overlapping hours between Onsite/Offshore teams. Frequent and structured meetings between customers and development teams must be arranged. Onsite coordinators must remain in constant touch with customers for any clarifications, validations and suggestions.
- **Configuration & Change Management** processes must be effective. Should have centralized check-in and checkout along with coordinated code integration between onsite and offshore teams. End of the day checks should be in place to ensure that components which are checked-out are checked back in.
- **Coding Standards** should be clearly defined and followed. As documentation is relatively less in XP, usage of inline comments should be encouraged.
- **Issue Resolution and Escalation** must be done at the earliest possible. Identify issues, clarify them, understand them, and resolve them. Ensure everyone understands the resolutions and preventive actions, if any.
- **Entry & Exit Criteria** for each task must be explicitly defined. Offshore should understand these in order to avoid schedule slippage.
- **Documentation** should be minimal as per project requirements. In an onsite-offshore model, minimum documentation is mandatory. This documentation ensures knowledge transition and help for training new team members.
- **Client Review** must be detailed and thorough - not just "sign-offs". Distance factor (onsite-offshore) should be taken in to account and "sign-offs", most often, determine the exit criteria for an iteration or release.
- **Productivity Increase** can be achieved by having triplets of developers with one onsite and two offshore. This is something innovative and when tried in an onsite-offshore model can be effective. Having combination of a technical expert and a subject matter expert, at offshore, added more value to pair programming. Here, the subject matter expert gets an opportunity to improve his technical skills and vice versa.
- Based on the type of project and needs, **Work Timings** shall be adjusted such that both onsite and offshore teams work during same timings. This can be achieved by making one of the teams work in night shift timings (or equivalent

matching timings). This type of timing adjustments are needed when either subject matter experts or technical experts are not available at onsite or offshore. In these situations, pairs must be formed with 1 onsite and 1 offshore team member. In cases where there work timings can not be matched, overlapping work hours should be ensured, ideally, at the start or the end of a work day.

- At offshore, **Work environment** for XP could not be made exactly like a 'pod', as demanded by XP, due to floor space and infrastructure issues. So, teams had to assemble in a meeting room for stand-up calls. Work spaces were organized, to the best possible extent, such that project team members were located close to one another.
- **Training** new team members in XP project was a challenge. With minimum documentation available, team members had to learn from fellow pairs. Due to this, interestingly, the learning curve of new team members was quick and seemed to be very effective.
- **Managing maintenance projects** in XP mode was another challenge. In reality, many XP practices could not be applied to maintenance projects.
- Usage of **agile Test Tools** like JUNIT, Eclipse etc., are mandatory for executing any XP project in an onsite-offshore model. These tools will help minimize effort overrun and schedule slippages, as XP methodology require significant amount of testing effort.

4 Conclusions

XP is a proven software development methodology to produce high quality software products. There are challenges to customize XP in an onsite-offshore software development situation. This paper is an attempt to share the experiences of a Telco's application consolidation project developed in XP methodology and successfully completed using an onsite-offshore model. This consolidation project was very complex to consolidate 3 legacy systems with unclear understanding of the to-be-developed application. Using traditional models like Waterfall, it could have taken a long duration to complete the project. By adopting XP, there were opportunities for the client to adjust the scope and requirement of the project until a clear understanding of the consolidated application was available. Ability to adjust and proceed further is a good feature of XP and this helped in successful completion of the project within the planned duration of the project.

References

Chromatic: Extreme Programming Pocket Guide. O'Reilly, Sebastopol (2003)

Donovan Wells (1999),
http://www.extremeprogramming.org/map/project.html

Eclipse Platform Technical Overview, Object Technology International, Inc. (2003),
http://www.eclipse.org/

Gittins, R.G., Hope, S., Williams, I.: Qualitative Studies of XP in a Medium Sized, Business. UPGRADE The European Online Magazine for the IT Professional III(2) (2002),
http://www.upgrade-cepis.org

Gittins, R.: Qualitative Studies of XP in a Medium Sized Business. In: Proceedings of the 2nd International Conference on Extreme Programming and Flexible Processes in Software Engineering, Sardinia, Italy, pp. 20–23 (2001)

Beck, K.: Extreme Programming Explained: Embrace Change. Addison-Wesley Longman Publishing Co., Boston (2000)

Butler, S.J., Hope, S.: Evaluating Effectiveness of Global Software Development Using the eXtreme Programming Development Framework (XPDF). In: ICSE 2003, Global Software Development Workshop. IEEE, Los Alamitos (2003)