CHAPTER 7
EFFECTIVENESS OF XP BASED MAINTENANCE

7.1 INTRODUCTION

Software maintenance is a continuous process of enhancing operational life of software. The maintenance process of software is dissimilar from software development and requires its own process model. The existing models of software maintenance such as, quick-fix, Boehm, Osborne, iterative-enhancement, and full-reuse are derivatives of the traditional models of development. These models have certain limitations in resolving the problems of maintenance. Alternatively, extreme programming practices such as, test driven development, refactoring, pair programming, continuous integration, small releases, and collective ownership help to resolve the problems of unstructured code, team morale, poor visibility of the project, lack of communication, and lack of proper test suites. The limitations of existing maintenance process models over other models and XP based maintenance approach is already discussed in the previous chapters. Hence, it can be deduced that a process model based on extreme programming practices can be an effective solution to software maintenance problems.

The iterative maintenance life cycle using XP is a dedicated process model proposed for software maintenance and is described in Chapter 2. Main phases of proposed iterative process model are identification and categorization, planning, analysis, design revision, change implementation, acceptance testing, and release. The proposed model incorporates XP practices in different activities of its phases to resolve the aforesaid problems of maintenance. This process model applies changes in an iterative manner using RC stories and old software as input during its phases to produce a modified product. The process, phases, activities and artifacts in existing models of software
maintenance are different from iterative maintenance life cycle using XP. Therefore, the comparative analysis of iterative maintenance life cycle using XP with traditional process models is required to analyze the effectiveness of proposed model.

The comparison is performed on the basis of various parameters such as, origin, requirement artifacts used, role of customer, planning, number of iterations, average duration of iterations, estimation techniques used, design improvement practices used, programming style used, code ownership, build time, code review practices used, testing strategy applied, way of regression testing performed, amount of documentation, phase execution, briefing style used, and application of process model.

The proposed model of maintenance uses development practices of XP for different activities of change implementation phase and management practices being applied throughout the process of maintenance. Organization of the chapter is as follows. Section 7.2 discusses the parameters used in the comparative study. The comparison and effectiveness of the proposed model is discussed in Section 7.3. The overall observation through case studies and experiments to apply XP in maintenance is discussed in Section 7.4. Lastly, Section 7.5 describes the summary of presented research.

7.2 PARAMETERS FOR COMPARISON

The maintenance process is applied in existing system and at the same time, it considers the organizational, technological, and other aspects for performing maintenance. The parameters used for comparing the iterative maintenance life cycle using XP with other maintenance models are discussed below.

Requirement artifacts: It shows the requirement artifacts used in problem reporting and change requirements elicitation.
Role of customer: Customer involvement is important aspects in XP as well as maintenance process. This parameter shows the involvement level of customers in the maintenance process.

Planning: Planning is an important activity of maintenance process. This parameter indicates the scope of planning such as, iteration planning containing limited change requirements, or a planning for entire set of requirements.

Number of iterations: It is the number of cycle considered to complete the list of change requirements.

Average duration of iterations: It shows the average duration of iteration. It indicates the average time for delivery of working software integrated with the requested changes.

Estimation techniques: It describes the effort estimation techniques used at different stages of maintenance process.

Design improvement practices: It indicates the use of design improvement practices (for example, refactoring), and their rate of recurrence during maintenance.

Programming style: Maintenance is a tedious job and requires programming style to make it pleasurable. This parameter indicates agreeable style of programming (for example, pair programming).

Code ownership: It indicates whole or part experience and awareness of system among maintenance team. This experience may be of whole system or module-wise.

Build time: It is the time consumed for integration and can be considered as hourly, daily, weekly or monthly.
**Code review practices:** It indicates the use of code review techniques such as, peer-review, walkthrough, or inspection. These practices review the code for any errors or mistakes.

**Testing strategy:** It shows the types of testing approach used at different levels of maintenance and movement of testing, i.e., test-first or test-last.

**Execution of regression testing:** It reflects the way to verify the impact of modifications on other part of the same system.

**Amount of documentation:** It is the extent of standard documentation housekeeping during the maintenance process.

**Phase execution:** It shows the execution flow of phases in maintenance process viz. sequential, parallel, or hybrid.

**Briefing style:** It indicates the use of briefing style used during maintenance process.

**Application of process model:** It indicates the application area of process model; for example, system developed using object-oriented or procedure-oriented technology.

### 7.3 COMPARISON OF MAINTENANCE PROCESS MODELS

The proposed process model is compared with existing process models of maintenance on the basis of parameters discussed in the previous section. The effectiveness of its different characteristics is discussed in subsequent paragraphs and it is shown in Table 7.1. The existing models of software maintenance are derived from the traditional models of software development. For example, origin of the quick-fix model is code-and-fix model and other ad-hoc development methodologies, Boehm model of maintenance is derived from spiral
Table 7.1: Comparative Analysis of Iterative Maintenance Life cycle using XP with other Maintenance Models

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Existing Maintenance Models</th>
<th>Iterative Maintenance Life Cycle using XP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirement artifacts</td>
<td>Software problem reports, software change request form</td>
<td>RC story</td>
</tr>
<tr>
<td>Role of customer</td>
<td>Requirement submission and validation</td>
<td>Requirement submission, prioritization, validation</td>
</tr>
<tr>
<td>Planning</td>
<td>For whole set of change requirements</td>
<td>Iterative planning according to the business needs of customer</td>
</tr>
<tr>
<td>Number of iterations</td>
<td>Single or multiple</td>
<td>Multiple</td>
</tr>
<tr>
<td>Average duration of iterations</td>
<td>One to six months</td>
<td>Two-weeks</td>
</tr>
<tr>
<td>Estimation techniques</td>
<td>ACT model, FP model, COCOMO 2.0 reuse model</td>
<td>Planning poker, SMEEM</td>
</tr>
<tr>
<td>Design improvement practices</td>
<td>If required</td>
<td>Continuous process through refactoring, pair programming, and TDD</td>
</tr>
<tr>
<td>Programming style</td>
<td>individually programming</td>
<td>Pair programming</td>
</tr>
<tr>
<td>Code ownership</td>
<td>Module wise</td>
<td>Collective code ownership</td>
</tr>
<tr>
<td>Build time</td>
<td>Continuous integration, i.e., daily basis</td>
<td></td>
</tr>
<tr>
<td>---------------------------------</td>
<td>------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>Code review practices</td>
<td>Continuous process through pair programming</td>
<td></td>
</tr>
<tr>
<td>Testing strategy applied</td>
<td>Unit testing, integration test and system test, regression testing, acceptance testing</td>
<td></td>
</tr>
<tr>
<td>Execution of regression testing</td>
<td>After module completion</td>
<td></td>
</tr>
<tr>
<td>Amount of documentation</td>
<td>Limited</td>
<td></td>
</tr>
<tr>
<td>Phase execution</td>
<td>Initial phase is sequential while remaining phase are parallel</td>
<td></td>
</tr>
<tr>
<td>Briefing style</td>
<td>Daily stand-up meeting at story board</td>
<td></td>
</tr>
<tr>
<td>Application of process model</td>
<td>Object-oriented programming technology</td>
<td></td>
</tr>
</tbody>
</table>

model of development, iterative-enhancement model of maintenance originated from iterative development models and full-reuse model of maintenance derived from the component-based development. IEEE-1219 and ISO-12207 maintenance processes are based on waterfall model and other traditional software development models. These maintenance process models have limitations in handling critical problems such as, unstructured code, team morale, poor visibility, and lack of communication between stakeholders. The proposed iterative
maintenance life cycle using XP is derived from IEEE-1219 process of maintenance and extreme programming practices. This XP based model resolves maintenance issues in a much improved manner. The proposed approach speeds up maintenance process and produces more maintainable code.

Problem reporting and its specification are important aspects of maintenance process and unambiguous change requirement and user acceptance criteria during maintenance reduces risks. Change requirements in existing software maintenance model are elicited and documented using existing artifacts, viz., software problem reports, software change request form etc. The proposed XP based maintenance model uses RC story format for problem reporting during maintenance process. As RC stories are written by end users; hence, they are very helpful for elicitation of unambiguous change requirements. RC story improves customer collaboration and simplifies requirement engineering process of software maintenance. RC story format resolves the problems of poor visibility of a project and lack of communication in maintenance process. RC story format contains acceptance criteria and recommended solutions. Acceptance criteria in RC story is helpful during acceptance testing to ensure that the components and the system as a whole provides expected results and recommended solution section supports for problem reproduction.

Involvement of customer is crucial in the success of development as well as during maintenance process. The existing process models involve customers during problem reporting and acceptance testing whereas, proposed XP based maintenance model recommends their availability at the maintenance site for the problem reproduction, feedback, and acceptance testing. Customer submits RC story with the acceptance criteria and recommended solutions. Customer also takes the business decisions by participating in release planning with the developers. Involvement of customer improves understanding of system, and thereby improves the productivity of maintenance team.
Planning is an important activity in maintenance process. Existing models of maintenance plan for whole set of change requirements or a part of it. The proposed XP based model of maintenance applies two levels of planning; namely, release planning and iteration planning. Release planning improves the involvement of customer to take business decisions. Iteration planning responds towards changes in requirements during maintenance.

Number of iterations used in maintenance process model to complete the list of change requirements is important to obtain the feedback that reduces risks. Existing models of maintenance have considered single iteration to complete all the change requirements that are submitted or can considered multiple iterations to complete it. The proposed XP based model of maintenance completes all change requirements in multiple iterations. In XP based model, initial iteration emphasizes on corrective maintenance and preventative maintenance in terms of code quality improvement. After initial iteration, perfective maintenance is performed according to the priority provided by the customer. Quality improvement habit of proposed model reduces future maintenance effort.

The duration of iterations indicate the average time for delivery of working software with the requested changes. The existing models of maintenance consumes single iteration to complete all change requirements submitted or can have multiple iterations to complete with average duration of maintenance being one to six months. The proposed XP based model of maintenance completes all change requirements in multiple iterations with average duration of iteration being two week with reduced risks and provides rapid feedback.

Estimation techniques and their application stage in software process, are very important in the success of software projects. The existing models of software maintenance applies existing models such as, ACT model, FP model, COCOMO 2.0 reuse model etc., for the
estimation of cost, size, and duration. These estimation models consider SLOC, FPs and object points as sizing units for determining maintenance effort. The proposed XP based maintenance model recommends two level estimations. An initial estimation is performed during planning phase using planning poker technique. The second level of estimation is performed at the end of analysis phase using SMEEM. For accurate estimation, SMEEM model incorporates value adjustment factors for estimation of size and effort of a maintenance project. The algorithmic approach of SMEEM uses story points to calculate the volume of maintenance. SMEEM provides task and release level for estimation of different volume of maintenance. The task-level model of SMEEM estimates the effort of implementing each maintenance task, which is considered in the form of an RC story. This model deals with small effort estimates. The release-level model of SMEEM estimates the effort of a planned set of RC stories or a planned release. Through estimation using planning poker and SMEEM, the iterative maintenance life cycle using XP enfolds the nature of maintenance.

Approaches of coding in existing models of software maintenance provide very limited scope for design improvement practices. Literature shows that 5% of total cost is spent in preventative maintenance [144]. The proposed XP based maintenance model provides scope for design improvement practices such as, refactoring, pair programming, and TDD. The code change approach that incorporates all these design improvement practices improves the maintainability of software product.

A change in the existing code, which is written by others is very common and lengthy job during maintenance of a software product. The existing models of maintenance support for solitary programming. The proposed XP based maintenance model highly recommends pair programming. Pair programming encourages exploration of more
alternative solutions through continuous review and makes maintenance tasks more enjoyable.

Staff turnover is one of the major problems of software maintenance. In the existing models of maintenance, maintainers work on a particular module without bothering about other modules of the same system. The proposed XP based maintenance model highly recommends the collective code ownership and pair programming; and hence, maintainers gain the knowledge and experiences of whole system through team and partner switching. Problems, which are raised during maintenance due to staff turnover, are resolved by this type of working culture.

In the existing models of maintenance, integration is performed on weekly basis. The proposed XP based maintenance model highly recommends continuous integration on daily-basis. This supports instant feedback and reduces testing effort after maintenance. Code review is an important aspect in maintenance of a software product. In existing models of maintenance, code review is performed after completion of a maintenance task. The proposed XP based maintenance model supports continuous review through pair programming.

The suitable testing strategy plays a vital role in the success of a maintenance project. Existing models of maintenance apply different type of testing at different stages of maintenance such as, unit testing, integration testing, system testing, regression testing, and productive system testing. The proposed XP based maintenance model applies different testing strategy. Test-first coding starts in change implementation phase. Change implementation phase also contains continuous integration and testing. Acceptance testing is applied on the basis of RC story acceptance criteria. Test-first coding provides several advantages in maintenance process such as, instant feedback while working on legacy code, confidence and courage while performing error-prone modifications, improved code readability, and
faster impact analysis before any modifications. Continuous integration and testing practices of XP resolves compatibility issues and interfacing problems of maintenance and provides a smoke testing environment to detect errors in early stages.

Regression testing is applied in existing models of maintenance to verify the impact of modifications. The proposed XP based maintenance model applies continuous integration and testing to verify the impact of modifications. Continuous integration and testing resolves compatibility issues and interfacing problems of maintenance providing instant feedback to the maintenance team.

Existing models of maintenance uses huge amount of documentation during the maintenance process. For example, IEEE-1219 itself recommends heavy documentation by the use of 41 different IEEE standards for software maintenance. The proposed XP based maintenance model recommends very limited documentation during the maintenance. The proposed XP based maintenance model emphasizes on code maintainability, satisfaction of customers and maintainers.

The proposed XP based maintenance model recommends hybrid approach in which identification and planning phases are executed in sequential order while remaining phases can be executed in parallel according to the needs. Existing models of maintenance use sequential phase execution. Existing models of maintenance follow normal meeting for briefing whereas, the proposed XP based maintenance model recommends daily stand-up meeting for briefing. This briefing style is used regarding the tasks for the day, technical issues and for information sharing among the team members. Existing models of maintenance can be used for application developed in any programming language. But the proposed XP based maintenance model is suitable for those applications, which are developed using object-oriented technologies.
7.4 EMPIRICAL ANALYSIS

Data of overall observations collected from experiments performed in Chapter 6 are summarized here to analyze the effectiveness of proposed XP based process model. In this experiment, we have applied the proposed XP based maintenance approach and traditional maintenance approach in maintenance of five different applications. Applications were Exam Control-Room Management (ECM), Student Feedback System (SFS), Library Circulation System (LCS), Student Information System (SIS), and Campus Search Engine (CSE). Observations with respect to code quality metrics, maintenance effort, and programmers’ confidence, which were analyzed during maintenance, are presented in Table 6.5 to Table 6.19. Here, statistics of observations are presented.

As the maintainability of software product depends upon product attributes such as, complexity, coupling and cohesion; therefore, data under observation is grouped into these three categories. In this representation, complexity attribute is presented as the addition of metrics such as, VG, WMC, DIT, NOC, NBD, and RFC. In the same way, coupling attribute is the addition of metrics such as, CBO, CA, CE, and RMI. Cohesion is represented as an addition of LCOM and WMC metrics values. Productivity of maintenance teams is represented using number of hours used in completing RC stories. Confidence of maintainers in the code, which is modified by them, is represented in the scale of 1 to 5.

Observations with respect to complexity metrics, which are considered during maintenance of all five projects, are presented in Figure 7.1. In case of ECM, SFS and CSE projects, XP based maintenance approach (XPM) produces less complex code in terms of VG, WMC, DIT, NOC, NBD and RFC as compared to traditional maintenance approach (TM) whereas, in case of LCS and SIS, TM produces less complex code. These values of complexity metrics
indicate that the XP based maintenance approach produces less complex code, which increases maintainability of code as compared to traditional approach of maintenance. Observation with respect to coupling metrics, which are considered during maintenance of all five projects, is presented in Figure 7.2, which are compared with XPM and TM. In all five projects, XPM produces low coupling code in terms of CBO, CA, CE, and RMI as compared to TM. These values of coupling metrics indicate that the XP based maintenance approach produces low coupling code and hence, increases maintainability of code as compared to traditional approach of maintenance.

Observations with respect to cohesion metrics, which are considered during maintenance for all five projects, is presented in Figure 7.3. It compares XPM and TM. In case of ECM, SFS, LCS and CSE projects, XPM produces lower values of LCOM and WMC as compared to TM whereas, in SIS project, TM produces lower values of LCOM and WMC as compared to XPM. These values of cohesion metrics indicate that the XP based maintenance approach produces high cohesive code that increases maintainability of code than traditional approach of maintenance.

Also, it is observed that in all five maintenance projects, XPM teams modified the systems in less time duration as shown in Figure 7.4. This significant difference shows that productivity of XPM team is higher than TM. All five teams perceived that the XP based maintenance approach are more confident about the code and also reported higher confidence in future changes to their products as shown in Figure 7.5. For example, in SFS project, average value of maintainability for XPM team is 3.4 and value of TM is 2.8.

7.5 SUMMARY

There are various process models for maintenance, which are derived from traditional models of software developments such as,
Figure 7.1: Complexity of Code

Figure 7.2: Coupling of Code
Figure 7.3: Cohesion of Code

Figure 7.4: Effort in Hours
quick-fix, Boehm, Osborne, iterative-enhancement, full-reuse etc. Each of these models has certain limitations to handle maintenance activity over XP based maintenance. The comparative analysis of XP based model is performed with traditional process models on the basis of various parameters such as, requirement artifacts, role of customers, planning, number of iterations, average duration of iterations, estimation techniques, design improvement practices, programming style, code ownership, build time, code review practices, testing strategy, execution of regression testing, amount of documentation, phase execution, briefing style, application of process model etc.

The comparative analysis of XP based model with traditional process models shows that XP based model is the better approach than other models because it elicits maintenance needs using RC story that provides unambiguous change requirements with acceptance criteria. The RC story improves customer collaboration and resolves the problems of poor visibility of the project, lack of communication in
maintenance process. The code change approach of XP based model improves the productivity of maintenance team. Iterative planning of XP based model improves the involvement of customer and responds to changes during maintenance. Two level estimation, i.e., planning poker at the initial level and SMEEM at the end of analysis phase, supports during maintenance planning. SMEEM provides more realistic effort estimation results. XP practices based code change approach improves the maintainability of software products through design improvement practices. Continuous integration and testing verify the impact of modifications that provides instant feedback. Besides that, the proposed process model is an extreme programming based approach; it includes documentation for future maintenance activity. The iterative maintenance life cycle using XP consists of all features of a process model that can resolve problems of maintenance.