EXTREME PROGRAMMING(XP)

By Ganesh Sambasivam
Ganesh.Sambasivam@isoftplc.com

Extreme Programming (XP) is a discipline of software development based on values of simplicity, communication & feedback. It works by bringing the whole team together in the presence of simple practices, with enough feedback to enable the team to Check where they are and to tune the practices to their unique situation.

XP recognizes that the end goal of a development project is to produce quality, production code that can be executed and maintained. Anything in a project that does not directly support this goal is questioned and discarded if appropriate.
XP takes 12 software development "best practices," and applies them to the extreme.
Every contributor to the Project is a part of the 'TEAM' and the Team interacts with the 'Customer' daily.

XP Core Practices:

1. The Planning Game: Business and development cooperate to produce the maximum business value as rapidly as possible. The planning game happens at various scales, but the basic rules are the same:

- Business comes up with a list of desired features for the system. Each feature is written out as a User Story, which gives the feature a name, and describes in broad strokes what is required. User stories are typically written on 4x6 cards.
- Development estimates how much effort each story will take, and how much effort the team can produce in a given time interval.
- Business then decides which stories to implement in what order, as well as when and how often to produce a production release of the system.

2. Small Releases: XP teams practice small releases in two important ways: First, the team releases running, tested software, delivering business value chosen by the Customer, every iteration. The Customer can use this software for any purpose, whether evaluation or even release to end users.The most important aspect is that the software is visible, and given to the customer, at the end of every iteration.
Second, XP teams release to their end users frequently as well. XP Web projects release as often as daily, in house projects monthly or more frequently.

3. Simple Design: XP uses the simplest possible design that gets the job done. The requirements will change tomorrow, so only do what's needed to meet today's requirements.

Design in XP is not a one-time thing but an all-the-time thing. There are design steps in release planning and iteration planning, plus teams engage in quick design sessions and design revisions through refactoring, through the course of the entire project.

4. Metaphor : Extreme Programming teams develop a common vision of how the program works, which we call the "metaphor". At its best, the metaphor is a simple evocative description of how the program works.

XP teams use a common system of names to be sure that everyone understands how the system works and where to look to find the functionality you're looking for, or to find the right place to put the functionality you're about to add.

5. Continuous Testing: XP teams focus on validation of the software at all times. Programmers develop software by writing tests first, and then code that fulfills the requirements reflected in the tests. Customers provide acceptance tests that enable them to be certain that the features they need are provided.

6. Refactoring: XP Team Refactor out any duplicate code generated in a coding session. Refactoring is simplified due to extensive use of automated test cases.

7. Pair Programming: All production code is written by two programmers sitting at one machine. This practice ensures that all code is reviewed as it is written and results in better Design,testing and better code.

Some programmers object to pair programming without ever trying it. It does take some practice to do well, and you need to do it well for a few weeks to see the results. Ninety percent of programmers who learn pair programming prefer it, so it is recommended to all teams.

Pairing, in addition to providing better code and tests, also serves to communicate knowledge throughout the team.

8. Collective Code Ownership: No single person "owns" a module. Any developer is expected to be able to work on any part of the codebase at any time.

9. Continuous Integration: All changes are integrated into the codebase at least daily. The unit tests have to run 100% both before and after integration.

Infrequent integration leads to serious problems on a software project. First of all, although integration is critical to shipping good working code, the team is not practiced at it, and often it is delegated to people who are not familiar with the whole system. Problems creep in at integration time that are not detected by any of the testing that takes place on an unintegrated system. Also weak integration process leads to long code freezes. Code freezes mean that you have long time periods when the programmers could be working on important shippable features, but that those features must be held back.

10. 40-Hour Work Week: Programmers go home on time. In crunch mode, up to one week of overtime is allowed. But multiple consecutive weeks of overtime are treated as a sign that something is very wrong with the process and/or schedule.

11. On-site Customer: Development team has continuous access to the customer who will actually be using the system. For initiatives with lots of customers, a customer representative(i.e Product Manager) will be designated for Development team access.

12. Coding Standards: Everyone codes to the same standards. The specifics of the the standard are not important: what is important is that all the he code looks familiar, in support of collective ownership.

XP Values

XP is a values-based methodology. The values are Simplicity, Communication, Feedback and Courage. XP's core values are best summarized in the following statement by Kent Beck: Do more of what works and do less of what doesn't.

Highlights of the four values are itemized below:

Simplicity encourages:

- Delivering the simplest functionality that meets business needs
- Designing the simplest software that supports the needed functionality
- Building for today and not for tomorrow
- Writing code that is easy to read, understand, maintain and modify

Communication is accomplished by:

- Collaborative workspaces
- Co-location of development and business space
- Paired development
- Frequently changing pair partners
- Frequently changing assignments
- Public status displays
- Short standup meetings
- Unit tests, demos and oral communication, not documentation

Feedback is provided by:

- Aggressive iterative and incremental releases
- Frequent releases to end users
- Co-location with end users
- Automated unit tests
- Automated functional tests
- urage is required to:
- Do the right thing in the face of opposition
- Do the practices required to succeed

CONCLUSION:

Extreme Programming is not a complete template for the entire delivery organization. Rather, XP is a set of best practices for managing the development team and its interface to the customer. As a process it gives the team the ability to grow, change and adapt as they encounter different applications and business needs. And more than any other process we have encountered Extreme Programming has the power to transform the entire delivery organization, not just the development team.