Notes for Extreme Programming(XP)

XP teams work in a series of fixed iteration cycles. Iterations typically last 1, 2 or 3 weeks each depending on the team.

# Process of XP:

1.Planning meeting at the beginning of every Iteration:

a) Planning meeting between development teams and customers.

customers’ requirements/desired features should be discussed(usually as user stories).

b) Divided user stories into individual engineering tasks. Developers then sign up for specific tasks, and estimate tasks(See The Planning Game below).

c) No one is allowed to sign up more tasks than what they did in last iteration.

2.During the iteration:

a) Implement the engineering tasks by following the core practices of XP listed below.

3.At the end of iteration:

a) Deliver a working system to customer. It does not need to be complete, but all the implemented functions should work completely and bug-free.

# 12 Core Practices of XP:

1.The Planning Game:

a) Business provides a list of desired features as User Stories. User Stories contain name, requirements and usually written in 4 x 6 cards.

b) Development estimates about how much effort each story take and how much effort the group can produce in a given period.

c) Business decides the priorities of user stories and timeframe and frequencies of production releases

2.Small Releases:

a) Start with smallest but useful feature set. Release early and often, every time add a few new features.

3.System Metaphor:

a) Provides an easy way to remember name convention, so that everyone(customers, developers, managers) can understand how system works.

4.Simple Design:

a) Use simplest possible and necessary design since requirements changes frequently.

5.Continuous Testing:

a) Test before Coding. Test-Driven Development.

b) Unit tests and acceptance tests are two basic tests in XP.

6.Refactoring:

a) Refactor codes to ensure the clarity and integrity.

b) complete tests are required so that refactor would not break the function of codes.

7.Pair Programming:

a) Two programmers write codes together so that code is reviewed during writing.

b) Programmers should switch their role frequently.

8.Collective Code Ownership:

a) Any developer in the team should be able to work on any parts of the program at any time.

9.Continuous Integration:

a) All changes are integrated into the codebase at least daily. The tests have to run 100% both before and after integration.

10.40-Hour Work Week:

a) Programmers go home on time, up to one week overtime is allowed, however, multiple consecutive weeks overtime implies something wrong with the process.

11. On-site Customer:

a) Development team should have assess to their customer. A customer proxy is usually used in commercial software development.

12. Coding Standrads:

a) Every team member should follow the same coding standards. Ideally, no one should be able to tell the difference between codes from different team members.

# Extreme Roles:

1.Tracker:

a) Leadership role.Track the progress of program, if it is going off the track, take actions like setting meeting with clients.

b) Do not responsible for writing code, only negotiation.

2.Manager:

a) Leadership role. Responsible for schedules meetings(like Iteration plan). Ensure everyone follow the meeting process, take records of meeting for future reporting and passes to tracker.

b) Responsible for some sundries, like fill the personnel actions and pay for the pizza.

c) Do not tell team members what to do, when to be done. These should be discussed in meetings.

3.Customer:

a) Give requirements/desired features usually as user stories.

b) Specifies acceptance tests(aka functional tests).

c) Decides the priorities of user stories.

4. Programmer:

a) Estimates stories, defines EngineeringTasks from stories, estimates how long stories and tasks will take, implements stories and UnitTests.

5.Coach:

a) Watches people and helps with anything to make project stay extreme.

6.Doomsayer:

a) Predict potential risks and problems.

7.Tester:

a) Implement and run integration test/acceptance test.

b) Not responsible for unit tests.