A Short Introduction to Extreme Programming

Extreme Programming (XP) is a modern software development process model that puts programming and the programmer at the centre of the development process.

The goal of XP is to produce high quality software by keeping the fun in programming.

Unfortunately, the name "extreme programming" has some negative connotations (like "extreme sports") suggesting high−risk software development by wild−eyed kids. The opposite is true.

Three Key Ideas of XP

- Pair Programming
- Design Simplicity
- Automated Testing

XP encompasses other business-oriented ideas, but these three can be applied in any almost setting.

Pair Programming

The basic rule of pair programming:

Computer code is never written or modified unless two people are sitting side by side in front of one computer.

Programming becomes a dialogue. Only one person types, but both people analyze, design, program and test.

Pairing is not a long−term commitment. After one task is done, the pair may split or stay together for the next task.

Pair Programming in Practice

Design simplicity ensures that a short−term pairing can be productive.

Writing test code together provides an opportunity for the pair to reach a common understanding of the task.

An established coding standard helps to avoid trivial quarrels.

Collective Ownership

Pair programming fosters collective ownership of the code.

Any programmer can change any part of the program at any time. Every programmer has an understanding of every part of the program at every time.

System integration takes place as often as possible to identify conflicting changes. A working version of the program always exists.

Design Simplicity

The basic rule of design: *Build the simplest thing that works.*

Simple designs lead to code that can be easily modified or replaced as requirements evolve.

"Embrace change" is the motto of XP.

Design Simplicity in Practice

Pair programming forces an immediate "simplicity check" as design ideas are explained to the second programmer.

Automated testing gives programmers confidence that design changes have been implemented correctly.

An overall system metaphor guides the design process.

Refactoring

A simple design has no duplicated code or logic.

Refactoring is the process of simplifying code to avoid or eliminate duplication.

For example, two classes that implement similar features might be combined into one class or might inherit from a common superclass.

Automated Testing

The basic rule of testing: *Write the tests first.*

Automated tests become an integral component of the program, which can be asked to self−test at any time.

System integration takes place as often as possible. Automated testing ensures that a working version of the program always exists.

Automated Testing in Practice

Design simplicity makes it easy to write tests.

Each programmer thinks of ways to test their partner's ideas.

A successful test provides the gratification of making it work that programmers crave.

XP in Practice

- Pair Programming
- Design Simplicity
- Automated Testing

The keys ideas support each other. The strengths of one compensate for the weaknesses of the others.

The use of XP must be a choice not a requirement.