

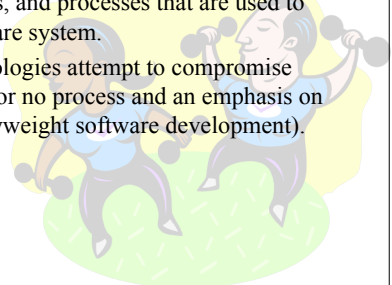
## Extreme Programming

*An Introduction*



## Agile or Lightweight Methodologies

- A software development methodology is a set of rules, practices, and processes that are used to create a software system.
- Agile methodologies attempt to compromise between little or no process and an emphasis on process (heavyweight software development).



## Agile or Lightweight Methodologies

- Agile methods are adaptive rather than predictive.
- Agile methods are people-oriented rather than process-oriented.
- Some form of self-adaptation may also be a part of an agile method - regular review of the process itself.



## Extreme Programming: Definition

- XP is a software development process that is organized around small releases and constant monitoring of the project's status.
- There is an emphasis on simple design based on OO principles and the use of pair programming.
- Pairs of programmers refactor, test, and write code.
- Customers are involved continuously via user stories and acceptance tests.



## XP Principles



- XP was created in response to the fact that requirements are always changing.
  - Addresses the problem of project risk.
- Designed to work with small sized teams of developers.
- XP has four major values
  - Communication
  - Feedback
  - Simplicity
  - Courage

## XP and Testing



- Strong emphasis on testing.
- Testing is the foundation of development.
- Tests are written at the same time as the production code is being written.
- In fact, tests may be written before the code is written.

## XP and Refactoring



- XP is an evolutionary design process that relies on the refactoring of a simple base system with every iteration.
- XP combines discipline with adaptivity.

## XP Activities



- Planning
- Design
- Coding
- Testing

## Planning



- Customer provides user stories.
  - Similar to use cases.
  - Used to describe the whole project.
  - Feature lists can be produced from the stories.
- Division of project into iterations.
  - The project is brought to completion in planned iterations.
  - Easier to incorporate user changes.

## Design Rules



- Keep it simple.
- Chose a system metaphor.
- When in doubt, make a spike.
- Do not add extra features early.
- Make it better by refactoring.

## Coding



- Write to agreed standards.
- Pair programming.
- Leave optimization until last.
- No overtime.

## Testing



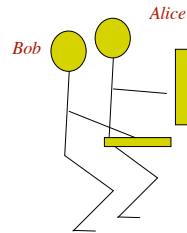
- Test first, code second.
- Use unit testing.
- Debugging involves creating more tests.
- Customers should define acceptance tests.

## Pair Programming

- Pair programming is a programming technique where two people program on one computer.
- One person is typing the code in and the other person is commenting and making suggestions.
- It can be thought of as a dialogue between two people as they simultaneously analyse, design, test, and code.



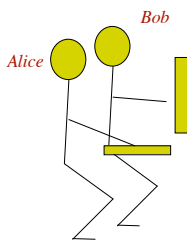
## Pairs in Action



- Bob and Alice have decided to work together on a module.
- Bob starts typing in the code and Alice points out an improvement.



## Pairs in Action



- After 30 minutes, Bob decides that he would like to take a break from typing.
- Alice switches seats with Bob and starts typing in code and Bob comments on the direction that they should go in next.



## Why Pairs?

- PP encourages communication.
  - Information diffuses through the entire team if the pairings are changed periodically.
  - The junior member of a pair can learn from the experienced senior member.



## Why Pairs?



- PP increases productivity.
  - Code quality is higher.
  - Encourages refactoring.
  - Encourages people to stick to XP principles such as continuous testing.
  - Allows reflective thinking while analysis, design and coding are being conducted.
  - Enhances the software development process.