Extreme Programming (XP)

1 Plan-driven vs Agile

Definition: Plan Driven

Well established means for transferring knowledge about how to structure and design systems

- Tend to put emphasis upon following procedures, and making use of notations to create models of the system
- Work well for those classes of problem that lend themselves to being modelled using widely-available forms

Definition: Agile

More flexible and more centred around ”concepts” and ”social models”

- They still require discipline in their application, and risk degenerating to a ”code and fix” form without this
- There is often still a need to determine the type of system architecture required, and what its main elements should be

2 Evolution of agile forms

- Waterfall model - descriptor of the over-arching process borrowed from a manufacturing context, but in practice each step is not as precise as implied
- Prototyping
- Spiral model - Incremental form
- Rapid application development
- Scrum - process driven approach
- DevOps - operations staff involved in process

3 Extreme Programming

Employs a mix of practices and techniques (rather than procedures) so forms a well established and sell defined example of the agile concept

3.1 Characterising XP

3.1.1 Values

Communication:

- Rich collection of procedures and activities to support this
- Stakeholders include customers, users, developers

Feedback:

- Emphasis is on delivering quality, rather than on speed

Simplicity:

- Every aspect of the system must be justified

Courage:

- Confidence to do risky things and accept change

Respect

- Core underpinning of an effective team approach

3.1.2 Practices

The planning game:

- At the start of each iteration, customers, managers and developers meet to flesh out, estimate and prioritise requirements called user stories, captured on story cards

Small releases:

- Release early and release often. Each release implements more stories and adds useful functionality

Metaphor:

- Basis for the system model, creating classes and methods to achieve the functionality embedded in the stories

Simple design:

- Avoid ”bells and whistles” and ask ”does the customer really need this feature”

Test-first programming:

- Write the tests before writing the code and then test continuously

Refactoring:

- Restructure the code without changing functionality to simplify

Pair programming:

- All code is written by two programmers working at a single machine discussing their work as they go

Continuous integration:

- Code is integrated into the system as often as possible, and after passing unit tests

Collective ownership

- The code is owned by all and they may make changes to it wherever they feel it necessary

On-site customer

- The customer works with the team to answer questions, perform acceptance tests and monitor progress

40 hour weeks

- Iterations should be sized so that overtime is not needed, on the basis that tired programmers make mistakes

Open workspace:

- Developers share workspace and use shared coding standards with clear conventions

4 Design in XP

XP is neither design-led nor plan-driven

Characteristics of design strategy in XP:

- KISS principle
- Use CRC (Class/Responsibilities/Collaboration) cards
- Reduce risk by using ”spike” solutions where appropriate
- Have a metaphor for talking/naming
- Add extra functions for the customer (only)
- Refactor regularly and mercilessly
- Design for test

5 CRC Cards

A ”classic” modelling technique, primarily used to handle requirements elicitation

CRC stands for Class-Responsibility-Collaborators and the use of the term ”class” can be used to describe any design element

Used to document collaborative design decisions and

- Identify components
- Discuss design issues in (customer-developer) teams
- Provide an informal specification of components

6 Coding

The popular emphasis on pair programming can easily obscure other aspects of the XP view of coding:

- The customer is always available
- The code must be written to agreed standards
- The unit tests are coded first
- All production code is developed by pair programming
- Only one pair integrates code at a time
- Integrate often
- Use collective code ownership
- Leave optimisation until last
- Coding should involve no overtime

7 Pair programming

All programming is undertaken by teams of two

- One person (the driver) uses the keyboard, and the other (the observer or navigator) looks at the screen while they discuss what they are doing
- The pairs swap roles on a regular basis
- Provides learning benefit for less experienced programmers

To work it does require:

- Mutual respect between the team
- Getting used to talking while programming

8 Test-First

- All code must have unit tests that are written before any code is produced
- All code must pass unit tests before it is released
- When a bug is found, tests are created
- Acceptance tests are run often and the score from these is published