

Extreme Programming

Eine Einführung

Thilo Utke
Mr.Utke@gmx.net

Das ist XP

- Agile Entwicklungsmethode/Disziplin
 - Leicht
 - Flexibel
 - Effizient
 - Kalkulierbar
 - Vergnüglich

Das ist XP

- Viel Feedback
- Inkrementelle/Evolutionäre Planung
- Flexible/Adaptive Implementierung von Funktionen
- Ständige/Automatisierte Tests
- Vertrauen
- Evolutionäres Design
- **So programmieren als ob man genug Zeit hätte**

Dafür eignet sich XP

- Teams von 2 bis 10 Entwicklern
- Projekte mit unklaren oder sich stark ändernden Anforderungen

Das macht XP aus

- Wendet best practices in extremer Weise an!
 - Code Reviews: **Pair Programming**
 - Kommunikation: **Pair Programming, On Site Customer**
 - Kurze Iterationen: **User Stories, Planning Game**
 - Testen: **Unit Tests, Functional Tests**
 - Integrationstests: **Continuous Integration**
 - Einfachheit: **Keep it Simple**
 - Design: **Refactoring**
 - Vertrauen: **Collective Code Ownership**

Pair Programming

- 2 Programmierer (Driver/Partner) + 1 Computer
- Driver implementiert den Code und erläutert
- Partner assistiert, hilft aus
- Rollen sollten wechseln
- Vorteile: Wissenstransfer, Besserer Code, Schnellerer Fortschritt, Motivation

On Site Customer

- Der Kunde ist vor Ort vertreten
- Erstellt oder spezifiziert Funktionstests
- Hilft beim Festlegen der Prioritäten
- Steht für Rückfragen bereit
- Vorteile: besseres Produkt für den Kunden, bessere Anforderungskontrolle

User Stories

- Beschreiben einzelner geschäftsrelevanter Funktionen
- Programmierer schätzen den Zeitaufwand für eine Story (max. 4 Wochen)
- Kunde ordnet den Funktionen Prioritäten zu
- Programmierer wählen aus dem vom Kunden gebildeten Pool aus
- Aus diesen Informationen setzt sich das nächste Release zusammen
- Vorteile: bessere Zeitplanung, max. Business Value pro Release, flexibel

Unit Tests

- Testen aller Einzelfunktionen, die kaputt gehen können
- Tests laufen automatisch und wiederholbar
- Test first
- Nur bei 100 %igem Bestehen geht der Code in das Release
- Vorteile: schnelle Fehlerfindung, keine Regressionen, Motivation, zusätzliche Dokumentation

Functional Tests

- Tests die zeigen, dass das Gesamtsystem wie vom Kunden gewünscht funktioniert.
- Tests die Probleme bei dem Zusammenspiel der Komponenten aufzeigen.
- Tests die automatisch und wiederholbar laufen
- Vorteile: schnellere Fehlerfindung, bessere Motivation, Fortschrittsmessung

Continuous Integration

- Build und Tests laufen automatisch
- Macht den Integrationsprozess zum Teil der täglichen Arbeit
- Dokumentation und Feedback über den Integrationsprozess
- Vorteile: Hilft den Fortschritt zu bewerten, Motivation, Integrationsprobleme lassen sich schneller finden

Keep it Simple

- Programmieren was gebraucht wird
- So Programmieren, dass es andere verstehen.
- Minimale Anzahl von Klassen und Methoden
- Vorteile: besseres Design, weniger Zeitaufwand

Refactoring

- Kontinuierliche Änderungen am Code die dessen Funktionsfähigkeit nicht beeinflussen.
- Ziele sind bessere Lesbarkeit, Testbarkeit, Verwendbarkeit, keine Redundanz
- Das Design wächst mit dem System.
- Vorteile: besseres Design

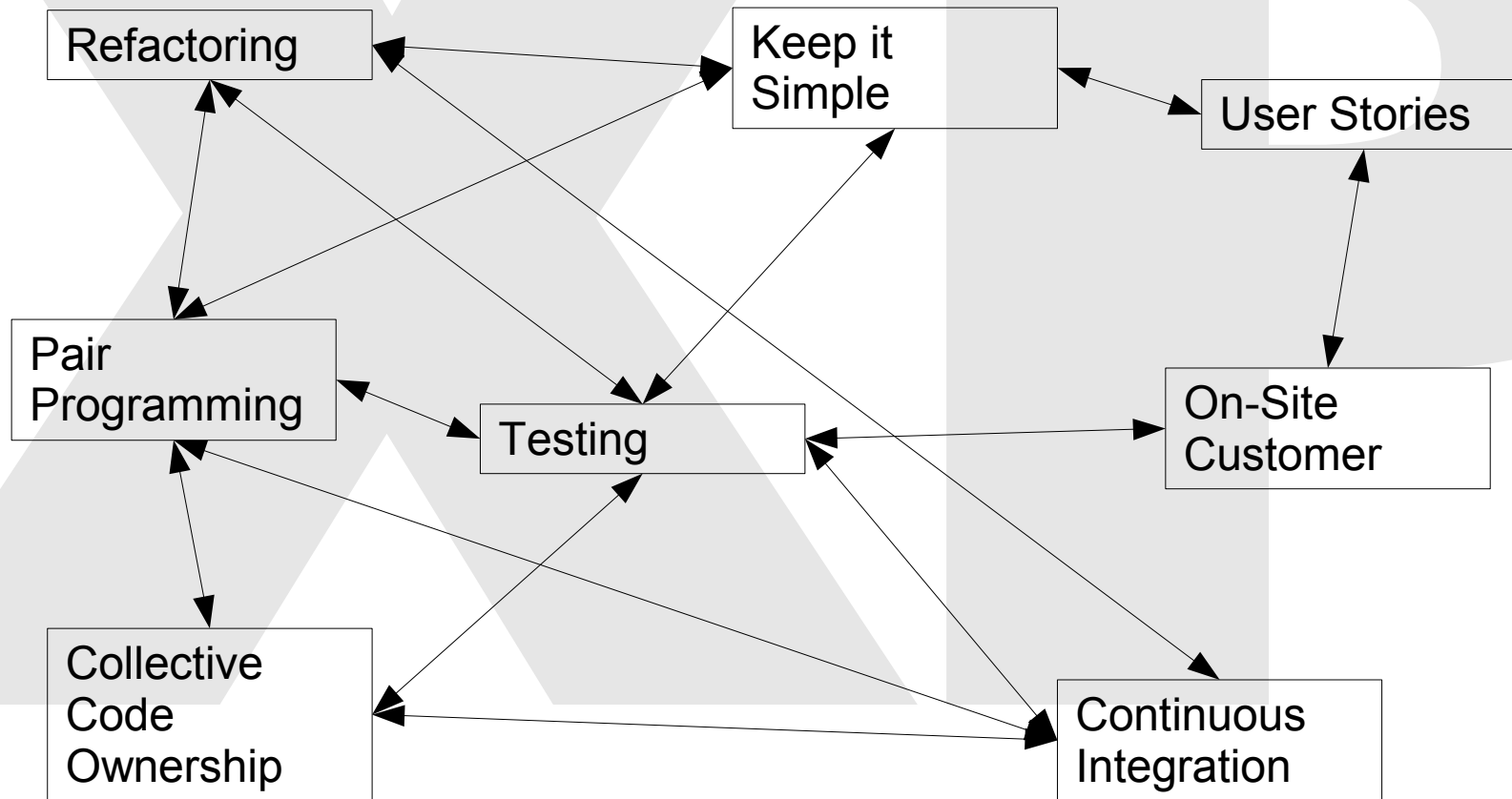
Collective Code Ownership

- Jeder ist für den gesamten Code verantwortlich.
- Jeder kann jederzeit am gesamten Code Änderungen vornehmen
- Vorteile: Keine Wartezeit auf Freigaben, sauberer Code, Wissenstransfer

Wir haben doch keine Zeit!

- Ziel ist die 40 Stunden Woche
- Releases werden eingehalten, notfalls mit weniger Funktionen
- Durch das Feedback bei XP lässt sich gut feststellen, wo die optimale Arbeitsgeschwindigkeit liegt.
- Vorteile: Besser konstant optimale Leistung statt kurze Höchstleistung und dann Fehler.
Programmierer sind auch nur Menschen.

XP Gegenseitigkeiten



Das braucht XP

- Gute Tool-Unterstützung
- Gute Arbeitsumgebung
- Gutes Management
- Motiviertes Team

Buchempfehlungen

Standardwerk:

Extreme Programming (176 S.)

von Kent Beck

Praxiseinsatz:

Extreme Programming Installed (265 S.)

Von Ron Jeffries, Ann Anderson, Chet
Hendrickson