Forschungszentrum Telekommunikation Wien
[Telekommunications Research Center Vienna]

CAMPARI-Trainings-Workshop
Overview of Extreme Programming

Martina Umlauft
umlauft@ftw.at

Zielgruppe und Vorkenntnisse

- Praktikanten CAMPARI und Interessierte
- Programmiererfahrung
- Erste Projekterfahrung

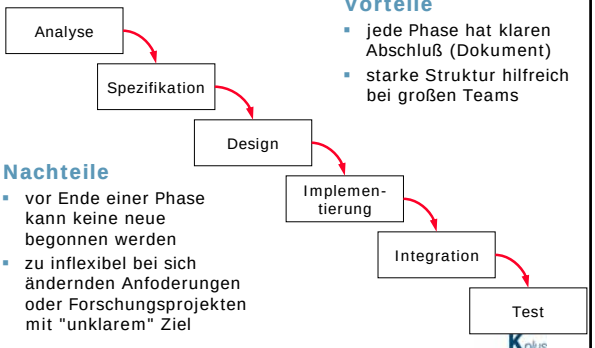
Overview

- Zielgruppe & Vorkenntnisse
- Traditionelle Methoden
- Extreme Programming
  - Planning
  - Designing
  - Coding
  - Testing

Traditionelle Methoden - Geschichtliche Einordnung

- **60er/frühe 70er:** "Cowboy Programming"
- **1968 Dijkstra:** "Goto Statement considered harmful"
- **80er:** einige Regeln - gut handhabbar
- **danach:**
  - Regelwerk überfrachtet
  - zu hoher Dokumentationsaufwand bei kleinen Teams
  - schlecht für sich ändernde Anforderungen
- **späte 90er:** neue Lightweight-Methoden, zb. Extreme Programming

Traditionelle Methoden - Wasserfall-Modell



Vorteile
- jede Phase hat klaren Abschluß (Dokument)
- starke Struktur hilfreich bei großen Teams

Nachteile
- vor Ende einer Phase kann keine neue begonnen werden
- zu inflexibel bei sich ändernden Anfoderungen oder Forschungsprojekten mit "unklarem" Ziel

Traditionelle Methoden - Probleme

- **47%** aller Projekte < 1 PJ
- **69%** aller Projekte 1 PJ - 10 PJ
- **89%** aller Projekte > 10 PJ

brauchen **> 25% Mehraufwand** als ursprünglich geschätzt

Aktuelles Beispiel:
deutsche Autobahnmaut "Tollcollect"

Overview

- Zielgruppe & Vorkenntnisse
- Traditionelle Methoden
- Extreme Programming
  - Planning
  - Designing
  - Coding
  - Testing

Extreme Programming - XP / 1

Lightweight Methodology:

- wenige Regeln
- leicht zu folgen

Für Projekte:

- mit sich schnell ändernden Anforderungen
- "new challenge"
- kleine Teams 2-10 Personen

- **Customer**, Management & Developer = **1 Team**

Extreme Programming - XP / 2

Idee:

Software, die **simpel & elegant** ist, ist ***mehr wert*** als Software, die complex & hard zu warten ist.

Typ. Projekt:

- 20x mehr Personalkosten als HW-kosten

- mit kryptischen Tricks HW-Kosten sparen bringt weniger als durch sauberes Programmieren Personalkosten zu sparen

Extreme Programming - XP / 3

Ziel:

- durch viele kleine Releases flexibel für wechselnde Anforderungen
- immer ein lauffähiges Produkt - schon während der Entwicklung
- bei Verzögerungen kann der Funktionsumfang zur Not gekürzt werden - keine "alles oder nichts" Situation
- keine Überlastung kleiner Teams mit starren Regeln oder unnötiger Dokumentation

XP Rules & Practices / 1

- Planning
- Designing
- Coding
- Testing

- Extreme Programming ist **keine** Rückkehr zum "Cowboy-Programmieren"!

XP Rules & Practices / 2

Planning:

- User stories are written
- Release planning creates the schedule
- Make frequent small releases.
- The Project Velocity is measured.
- The project is divided into iterations.
- Iteration planning starts each iteration.
- Move people around.
- A stand-up meeting starts each day
- Fix XP when it breaks.

XP Rules & Practices / 3

Designing:

- Simplicity.
- Choose a system metaphor.
- Use CRC cards for design sessions.
- Create spike solutions to reduce risk.
- No functionality is added early.
- Refactor whenever and wherever possible.

XP Rules & Practices / 4

Coding:

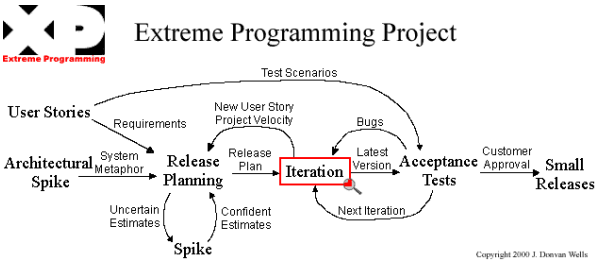
- The customer is always available.
- Code must be written to agreed standards.
- Code the unit test first.
- All code is pair programmed.
- Only one pair integrates code at a time.
- Integrate often.
- Use collective code ownership.
- Leave optimization till last.
- No overtime.

XP Rules & Practices / 5

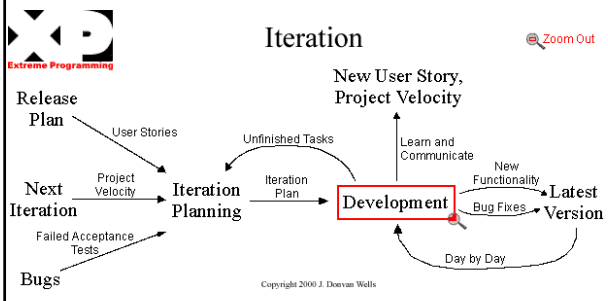
Testing:

- All code must have unit tests.
- All code must pass all unit tests before it can be released.
- When a bug is found tests are created.
- Acceptance tests are run often and the score is published.

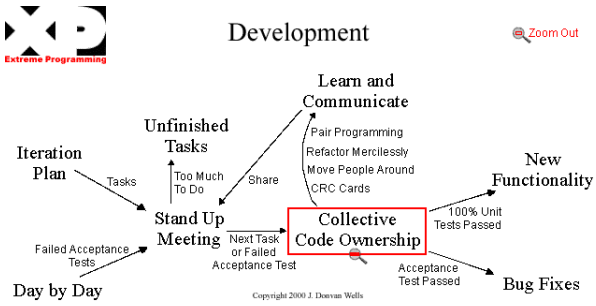
XP Übersicht / 1



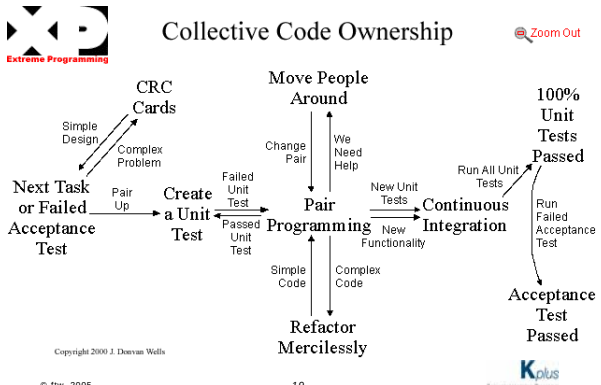
XP Übersicht / 2



XP Übersicht / 3



XP Übersicht / 4



XP - Planning

- User stories are written
- Release planning creates the schedule
- Make frequent small releases.
- The Project Velocity is measured.
- The project is divided into iterations.
- Iteration planning starts each iteration.
- Move people around.
- A stand-up meeting starts each day
- Fix XP when it breaks.

XP - User Stories

- Vom Kunden geschrieben
- ~1 Absatz in natürlicher Sprache
- zur Zeitabschätzung
- für Acceptance Tests
- 1 Story: 1-3 Wochen "ideal time"
- 60-100 Stories -> Release Plan

XP - Release Planning

- Release Plan = overall project layout
- => Iteration Plan für jede Iteration

Release Plan Meeting:

- Dev. Team: Aufwandsabschätzung
- Kunde: Prioritäten für Stories
- Stories auf Kärtchen
- gemeinsam Sets für Releases ausarbeiten
- Schätzungen:
  - wieviel Stories vor Tag X oder
  - wie lang für N Stories
  - "yesterday's weather"
- alle paar Iterationen ein RP-Meeting

XP - Iterations

- Vor jeder Iteration Planung im Detail
- ~1 Dutzend Iterations von 1-3 Wochen Länge
- Kunde wählt Stories aus
- Ausarbeitung Tasks mit 1-3 "ideal days" Länge
- Developer "sign up for" Tasks und schätzen "real time"
- Wenn sich Project Velocity (Projektgeschwindigkeit) stark ändert Release Planung neu verhandeln

XP - Designing

- Simplicity.
- Choose a system metaphor.
- Use CRC cards for design sessions.
- Create spike solutions to reduce risk.
- No functionality is added early.
- Refactor whenever and wherever possible.

XP - CRC Cards

- **C**lass, **R**esponsibility, and **C**ollaboration
- kleine Kärtchen
- jede Karte ein Objekt mit Responsibilities und Collaboration Classes
- CRC Session: Programmablauf anhand der Karten erzählen => "Simulation"

XP - Spike Solution

- Bei techn. Problemen / Unklarheiten
- eine einfache "**Wegwerfimplementierung**" um ein spezielles Problem besser zu verstehen

=> bessere Aufwandsabschätzungen, vermindertes Risiko

XP - Coding

- The customer is always available.
- Code must be written to agreed standards.
- Code the unit test first.
- All code is pair programmed.
- Only one pair integrates code at a time.
- Integrate often.
- Use collective code ownership.
- Leave optimization till last.
- No overtime.

XP - Customer

- ist Teil des Teams
- schreibt User Stories
- legt Prioritäten fest
- gibt Feedback zu Releases
- steht für Detailfragen zur Verfügung
- hilft beim Testen (zb. Bereitstellen v. Testdaten)

XP - Code Unit Tests First

- Vor eigentlichem Programm!
- Hilft, Requirements zu verstehen
- Hilft beim Denken
- Wenn alle Tests laufen ist das Programm korrekt / fertig

Ablauf:

- Test für kleinen Teilbereich
- Code für diesen Teil
- nächsten Test erstellen
- Code
- ...

XP - Pair Programming

- 2 Leute an einem Computer sind nahezu gleich effizient wie 2 einzelne, manchmal effizienter(!)
- Qualität des Programms besser!

- Lerneffekt
- Motivation
- "4 Augen sehen mehr als 2"
- ...

XP - Integration

- Nur ein Paar committed gleichzeitig ins Repository
- dadurch immer konsistente Version
- Oft Integrieren vermeidet wochenlange Integration am Ende des Projekts!
- Mind. jede Woche, besser jeden Tag!

XP Testing

- All code must have unit tests.
- All code must pass all unit tests before it can be released.
- When a bug is found tests are created.
- Acceptance tests are run often and the score is published.

XP Unit Tests

- Use Unit Test Framework
- Tests vor eigentl. Code schreiben
- Tests ins Repository einchecken

Ermöglicht:

- Collective Code Ownership
- Refactoring
- Häufige Integration

- Spart im Endeffekt Zeit, auch wenn am Anfang aufwändig

XP Acceptance Tests

- Black Box System Test
- erzeugt aus User Stories
- Ergebnis für jeden im Team zugänglich
- sichert Kundenzufriedenheit!

Lessons Learned

- Pair Programming: nie 2 neue zusammen
- Experiment Univ. Utah:
  1. Aufgabe 60% mehr Personenstunden
  2. Aufgabe 20% mehr
  3. Aufgabe 10% mehr
- dh: einzelner 10h, Paar 5:15
- ABER: bessere Qualität
  (15% bessere Ergebnisse bei der Abgabe)

References

- **www.extremeprogramming.org**
- **The Mythical Man Month**, *Essays on Software Engineering*, Frederick P. Brooks, Jr., Addison-Wesley, ISBN 0-201-00650-2

Folgetermine

- **Fr, 26. Aug, 9:00-12:00**
  9:00-10:30: Ed Schofield, Python
  10:30-12:00: Martina Umlauft, Doxygen

The End

Danke für die Aufmerksamkeit!