



EXTREME PROGRAMMING

ACM/GI-Regionalgruppe Bremen, 12.6.2001

Tammo Freese
OFFIS, Oldenburg

freese@acm.org
<http://www.tammofreese.de>

Frank Westphal
unabhängiger Berater

westphal@acm.org
<http://www.frankwestphal.de>

GLIEDERUNG

- Übersicht
- Planen
- Demo: Planungsspiel
- Programmieren
- Demo: Programmiererepisode
- Zusammenfassung

AGILE PROZESSE

- Menschen und Zusammenarbeit vor Prozessen und Werkzeugen
- Funktionierende Software vor umfassender Dokumentation
- Zusammenarbeit mit dem Kunden vor vertraglicher Verhandlung
- Reaktion auf Veränderung vor Einhaltung eines Plans

EXTREME PROGRAMMING

XP ist ein Software-Entwicklungsprozess für kleine Teams. In vier Worten:

- human
- leichtgewichtig
- agil
- adaptiv

WERTESYSTEM

Im Kern beruht XP auf vier Werten:

- Kommunikation
- Einfachheit
- Feedback
- Mut

XP erfordert Disziplin und Prinzipientreue

ÖKONOMISCHES MODELL

- Langsamer Geld ausgeben
- Schneller Einnahmen erzielen
- Produktive Lebensdauer des Projekts verlängern

- Abkehr vom Festpreisprojekt

PLANEN

- Kunde vor Ort
- User Stories
- Funktionale Abnahmetests
- Release-/Iterationsplanung
- Trennung technischer und geschäftlicher Verantwortung

USER STORIES, KUNDE, ABNAHMETESTS

Anforderungsdefinition ist ein Dialog,
kein Dokument

- Anforderungen erzählt als User Stories
 - passen jeweils auf eine Karteikarte
 - werden zur rechten Zeit im Dialog verfeinert
 - validiert durch Abnahmetests des Kunden
- Wenn wir unseren Kunden nicht direkt verfügbar haben, müssen wir raten

RELEASE-/ ITERATIONSPLANUNG

- Kurze Iterationen (1..3 Wochen)
- Kurze Releasezyklen (1..3 Monate)
- Vier Variablen:
Zeit, Umfang, Ressourcen, Qualität
- Trennung technischer und geschäftlicher Verantwortung
 - Kunde trifft geschäftliche Entscheidungen
 - Entwickler treffen technische Entscheidungen

PLANUNGSSPIEL

- Spielelemente: Stories
- Spieler: Kunde (K), Entwicklung (E)
- Spielzüge:
 1. Storycard schreiben (K)
 2. Schwierigkeit der User Story einschätzen (E)
 3. Stories nach Geschäftswert sortieren (K)
 4. Teamgeschwindigkeit bestimmen (E)
 5. Umfang für die Version festlegen (K)

PROGRAMMIEREN

- Programmieren in Paaren
- Komponententests
- Fortlaufende Integration
- Einfaches Design
- Refaktorisieren
- Gemeinsame Verantwortlichkeit
- Metapher

PROGRAMMIEREN IN PAAREN

Keine Zeile Code wird geschrieben,
ohne daß zwei Paar Augen auf den
Bildschirm gerichtet sind

- Partner wechseln sich an der Tastatur ab, Programmierpartner rotieren
- Höhere Codequalität
- Größere Produktivität (mehr Spaß!)
- Bessere Wissensverbreitung

KOMPONENTENTESTS

Eine Funktion existiert solange nicht, bis sie durch einen automatisierten Test abgeprüft wird

- Tests werden gesammelt, gepflegt und müssen bei jeder Integration zu 100% laufen
- Selbsttestender Code mittels Testing Framework xUnit

FORTLAUFENDE INTEGRATION

- Mehrmalige tägliche Integration
- Zu jedem Zeitpunkt integriert maximal ein Programmierduo, garantiert durch
 - Integrationsrechner
 - Integrationstoken
 - Versionsverwaltung

EINFACHES DESIGN

Sobald wir etwas tun,
das wir später vielleicht benötigen,
beschäftigen wir uns nicht mehr mit dem,
was wir tatsächlich benötigen

- "What's the simplest thing
that could possibly work"
- "You ain't gonna need it"

REFAKTORISIEREN (1)

"Nobody gets things right first time"

- Evolutionäre Designstrategie
- Verbessern des Designs von Code,
 - nachdem er geschrieben wurde
 - ohne dessen Verhalten zu ändern
- Werkzeug: Tests und Refactoringbrowser
- Zwei Hüte: Refaktorisieren, neue Funktion

REFAKTORISIEREN (2)

"Listen to the code"

- Code, der schlecht riecht:
 - Lange Funktionen
 - Duplizierte Logik
 - Kommentare
 - switch-Ketten, verschachtelte if-then-else
 - Code, der die Intention des Programmierers nicht ausdrückt

GEMEINSAME VERANTWORTLICHKEIT

Der Code gehört dem Team

- Natürliches Arbeiten mit Objekten führt uns durch viele Klassen – Lassen wir uns einfach vom Code führen
- Jedes Paar soll jede Möglichkeit zur Codeverbesserung jederzeit wahrnehmen
- Das Team folgt gemeinsamen Programmierstandards

METAPHER

- System von Namen
- Verbessert unser Verständnis des Systems
- Leitet das Team in der Architektur (Generative Metapher)
- Beispiele:
 - Geschichte von fünf Klassen
 - eShop (Warenkorb, zur Kasse gehen)
 - Lines/Bins/Parts/Stations (Datenflußmaschine)

PROGRAMMIEREPISEDE

1. Wir überlegen uns eine Funktion
2. Wir schreiben einen Test dafür
3. Wir refaktorisieren den Code in Form
4. Wir implementieren die Funktion
5. Wir lassen die Test Suite laufen
6. Wir refaktorisieren unseren Code
7. Wir sind fertig, wenn alle Tests erfüllt sind

ZUSAMMENFASSUNG

- Softwareentwicklung soll Spaß machen
- Tut sie das nicht, ist der Prozess defekt
- Es muss nicht XP sein
- Oft kann es nicht XP sein
- Menschen müssen ihren eigenen Prozess akzeptieren
- Maßschneidern Sie Ihren Prozess und verbessern sie ihn kontinuierlich

WAS PROGRAMMIERER AN XP BEGEISTERT

- Sie können Qualität abliefern
- Sie erfahren, was der Kunde in welcher Reihenfolge entwickelt haben möchte
- Sie bekommen Unterstützung von Kollegen und Kunden, wenn sie sie benötigen
- Sie dürfen ihre eigenen Abschätzungen machen und sie aktualisieren
- Sie können Verantwortlichkeit übernehmen und bekommen sie nicht zugewiesen

WAS KUNDEN AN XP BEGEISTERT

- Sie erfahren, was bis wann und zu welchen Kosten erreicht werden kann
- Sie können aus jedem Personentag den maximalen Wert ziehen
- Sie sehen den Projektfortschritt
- Sie dürfen ihre Meinung ändern
- Sie erfahren von Planänderungen so früh wie möglich und können jederzeit aussteigen

INFORMATIONEN

- Kent Beck: Extreme Programming
- Martin Fowler: Refactoring
- www.extremeprogramming.org
- www.xprogramming.com
- xp.c2.com
- www.yahogroups.com/group/xp-forum