Extremes Programmieren

Übersicht, Demonstration, Erfahrungen

ACM/GI–Regionalgruppe Hamburg, 16.3.2001

Frank Westphal
unabhängiger Berater

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

Tammo Freese
OFFIS, Oldenburg

freese@acm.org

Gliederung

- Motivation
- Programmieren
- Demo: Programmierepisode
- Planen
- Demo: Planungsspiel
- Erfahrungen

Motivation

- Softwareerstellung im Dialog
- Risiken in Softwareprojekten
- Agile Prozesse
- Wertesystem
- Ökonomisches Modell

Softwareerstellung im Dialog

- Industrieller Fertigungsprozess für die Softwareerstellung ungeeignet
- Softwareerstellung als Dialog zwischen Kunden und Entwicklung
- Neue Paradigmen: Cluetrain Manifesto
- Besinnung auf das Wesentliche
- Wie würden wir programmieren, wenn wir genug Zeit hätten?

Kommt ein Aufstand der Programmierer?

- Wir wollen Qualität abliefern
- Wir wollen wissen, was der Kunde in welcher Reihenfolge entwickelt haben möchte
- Wir wollen Unterstützung von Kollegen und Kunden bekommen, wenn wir sie brauchen
- Wir wollen unsere eigenen Abschätzungen machen und sie aktualisieren dürfen
- Wir wollen Verantwortlichkeit übernehmen und sie nicht zugewiesen bekommen

Kommt ein Aufstand der Kunden?

- Kunden wollen wissen, was bis wann und zu welchen Kosten erreicht werden kann
- Kunden wollen aus jedem Personentag den maximalen Wert ziehen
- Kunden wollen den Projektfortschritt sehen
- Kunden wollen ihre Meinung ändern können
- Kunden wollen Planänderungen so früh wie möglich erfahren, Kunden sollten jederzeit aussteigen können
- Kunden wollen nicht belogen werden

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

Extreme Programming ist ein
leichtgewichtiger, hochdisziplinierter
Software–Entwicklungsprozess
für kleine Teams,
die während der Entwicklung mit vagen
und sich rasch ändernden Anforderungen
konfrontiert werden.

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

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)

Programmierepisode

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

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)

Erfahrungen

- Zufriedene Kunden
- Teamgeist, Spaß zurückentdeckt
- Langlebige Software
- XP ist nicht für jeden
- XP ist nicht einfach

Fazit

- 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

Informationen

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

XPeditionstraining

XP–Workshop

19.–20. April

Gastwerk Hotel

www.XPeditionstraining.de