EXTREME PROGRAMMING

Studientag 01853

Moderne Programmiertechniken und -methoden

ANSATZ

- im Mittelpunkt steht der Programmierer
- Methode: schnelles Feedback, schnelle Fehlerbehandlung
- Ziel: qualitativ hochwertige Software, die im Zeit- und Kostenrahmen liegt
  - Verantwortung liegt bei Programmierer und Kunden(!)

TEST FIRST

- Annahme: Programmierer ist blind für eigene Fehler
- Tests werden zuerst geschrieben, dann das Programm
  - Code kann einfach von Dritten geändert werden
  - Regressionstests sind schon vorhanden

TEST FIRST II

- Funktionstests
- testen das Gesamtsystem
- ersetzen Anforderungsspezifikation
- Verantwortung liegt beim Kunden

PAIR PROGRAMMING

- Ansatz: ständiger Review-Prozess zur frühen Fehlererkennung
- nach Diskussion schreibt ein Programmierer den Code, der andere schaut nur zu und hinterfragt den Quellcode
- ständiger Wechsel des Partners

PAIR PROGRAMMING II

- Vorwurf: Ressourcenverschwendung, Gegenargumente:
  - kein wissenschaftlicher Nachweis
  - erhöht Code-Qualität und zahl sich so am Ende des Projekts aus
  - Wissen über Code auf mehrere Köpfe verteilt
  - gegenseitiges Lernen steigert Langfristig die Produktivität

ÄNDERUNGEN

- klassisches Wasserfallmodell
  - Änderungen erfolgen später und sind deshalb teurer
- XP
  - Annahme: Änderungen sollen billig und jederzeit möglich sein

➡ These: langfristige Planung lohnt nicht

ARBEITSWEISE

- Code ist eher einfach denn umfassend
- Code dokumentiert sich selbst
- Rückkopplung: neue Funktionen werden nach jeder Iteration dem Kunden vorgelegt

ARBEITSWEISE II

- Anforderungen (Systemfunktionen) jeweils auf Karteikarten notiert
- werden von den Programmierer-Paaren abgearbeitet
- ersetzen den klassischen Projektplan
- Programmierer schätzen den Aufwand pro Karte

ARBEITSWEISE III

- Ende des Projekts nach fixer Anzahl an Releases
- jedes Release besteht aus mehreren Iterationen
- jede Iteration wird in 1-4 Wochen entwickelt
- Arbeitspensum je Iteration nach Können der Programmierer festgelegt

RESOURCEN

- klassischer Ansatz: Projektgröße und Zeitplan bestimmen die Teamgröße
- XP: Teamgröße ist fix, Zeit (Anzahl der Releases) nach Projektgröße
  - Verspätung eines Releases wird durch funktionale Einschränkungen des nächsten Releases begegnet
- Kunde sollte für Rückfragen vor Ort sein

COLLECTIVE CODE OWNERSHIP

- Änderungen an einem Modul werden von verschiedenen Programmierern durchgeführt
  - klassischer Autor entfällt
  - komplexer Code wird durch einfachen ersetzt
  - Fehler werden vom Entdecker und nicht vom Verursacher behoben
- erfordert Formatierungs- und Namenskonventionen

PROZESS

- XP ist iterativer Prozess
- Ansatz: „If you‘re going to fail, fail early“
- kurze Iterationen zeigen Fortschritt und schützen vor Verschiebung von Aufgaben in Richtung Projektende
- ggf. schon frühzeitiger Prouktiv-Einsatz einer Iteration
  - kann zu Änderung der Prioritäten einzelner Funktionalitäten führen

VORAUSSETZUNGEN

- mind. 2 Programmierer
- max. 10 Programmierer, sonst zu viel Overhead durch Kommunikation
- Einsatz moderner Entwicklungstools (Test-Framework, Refactoring-Tools)
- laut Schöpfern für OOP

VORAUSSETZUNGEN II

- keine Altlasten
- Bereitschaft, viel nicht produktiven Code (Tests) zu schreiben
- zu Projektbeginn unklare Anforderungen
- experimentierfreudiger Kunde, der stark involviert werden kann

WERKZEUGE

- Editoren / IDEs mit Syntax-Vervollständigung
- Refactoring-Tools
- (D)VCS (Subversion, Git)
- Build-Systeme, die automatisiert Iterationen übersetzen
- schon gut in vorhandenen Tools integriert, also geringe Kosten in Infrastruktur

ZUSAMMENFASSUNG

1. Prinzip der kleinen Schritte (iterativer Prozess)
2. Prinzip der Einfachheit (Design während der Implementierung)
3. Prinzip der gemeinsamen Verantwortung
4. Prinzip des sprechenden Codes

ZUSAMMENFASSUNG II

5. Prinzip des Kunden vor Ort

6. Prinzip keine Überstunden

- Arbeitspensum = Zeit x Produktivität

7. Prinzip der Rückkopplung

- Unit-Tests, Integrationstests, Iterationen, Release)

AGILE PROZESSE

- Problem des XP: alles ist dynamisch, bis auf den Prozess selbst

- agile Prozesse regen Projektverantwortliche zum reflektieren über Prozessmodelle an