

Extreme Programming: Überblick



Inhalt

- ◆ Prinzipien
- ◆ Rollen
- ◆ Planung
- ◆ Implementierung
- ◆ Praktiken
- ◆ weitere Vorgehensweisen
- ◆ Grenzen



Prinzipien (Auswahl)

- ♦ strukturiertes Vorgehen – kein Chaos!
- ♦ Teamarbeit
- ♦ Offenheit
- ♦ stetige Kommunikation
- ♦ gleichmäßige Auslastung, keine Überstunden
- ♦ “gelebtes Risikomanagement”



Rollen

- ▶ Product Owner: z.B. Produktmanagement, Marketing, Kunde – hat Verantwortung für das Produkt
- ▶ Kunde: kann identisch mit Product Owner sein – Auftraggeber
- ▶ Entwickler: Architekt, Designer, Programmierer, Tester, etc.
- ▶ Projektmanager: z.B. Product Owner oder Entwickler – Teamleitung
- ▶ User: Nutzer des Produkts



Planung (1)

- ◆ Release-Planung
- ◆ Anzahl und Dauer (1-4 Wochen) der Iterationen definieren
- ◆ Story Cards erstellen (Kunde)
- ◆ daraus Anforderungen
- ◆ Alternative zu Story Cards: CRC-Cards
 - ◆ CRC = Class Responsibility Collaboration
 - ◆ beschreibt Akteur und dessen Verantwortlichkeiten und Interaktionen mit anderen Akteuren



Planung (2)

- ◆ Nutzen- und Risikoabschätzung der Story Cards
- ◆ Priorisierung:
 - ◆ Prio 1: hohes Risiko, hoher Wert
 - ◆ Prio 2: niedriges Risiko, hoher Wert
 - ◆ Prio 3: niedriges Risiko, niedriger Wert
 - ◆ Prio 4: hohes Risiko, niedriger Wert
- ◆ Implementierung der Prio 4 vermeiden



Planung (3)

- ♦ Analyse der Kundenwünsche
 - ♦ Must-haves
 - ♦ lineare Kundenzufriedenheit
 - ♦ Exciters (es geht auch ohne, aber mit erzeugt Begeisterung)
- ♦ Aufwandsabschätzung: Story Points – relativer Aufwand der Stories (Entwickler)
- ♦ Ergebnis der Planung:
Zuweisung von Story Cards zu Iterationen



Implementierung einer Iteration (1)

- ♦ Iterations-Planung:
 - ♦ Zerlegung der User Stories in Tasks
 - ♦ Task: techn. Arbeitspaket (Stundenbereich)
- ♦ Verteilung der ersten Tasks im Team
- ♦ Implementierung der Tasks
- ♦ Task beendet: neuen Task zuweisen und implementieren



Implementierung einer Iteration (2)

- ▶ Story Card ist beendet, wenn alle zugehörigen Tasks beendet sind
- ▶ Story Card nicht beendet zum Release-Termin:
in nächste Iteration verschieben



Praktiken (1)

- ◆ Pair Programming
- ◆ Kollektives Eigentum (Code und Wissen)
- ◆ Permanente Integration
 - ◆ erfordert automatisches Build-System
- ◆ Test Driven Development
- ◆ Einbeziehung des Kunden, On-Site Customer
- ◆ Ständige Refactorings
 - ◆ Keep the software soft!



Praktiken (2)

- ◆ Keine Überstunden
- ◆ kurze Iterationen (1-4 Wochen), ständige Transparenz
- ◆ Coding Standards
- ◆ Einfaches Design, KISS
 - ◆ “It's easy to have a complicated idea. It's very very hard to have a simple idea.” -- Carver Mead
- ◆ Planing Game
 - ◆ Planungsphase mit Entwicklern und Kunde



Weitere Vorgehensweisen

- ◆ Stand-up Meetings
 - ◆ tägliche, kurze Besprechung
 - ◆ Was wurde erreicht?
 - ◆ Was kommt heute?
 - ◆ Wo gibt es Probleme?
- ◆ neue Paar-Bildung für Pair Programming (nach Bedarf)
- ◆ nach Release: Feedback von Management und Usern
 - ◆ Akzeptanztests, häufiges Feedback
 - ◆ Einfluss auf aktuelle Iteration und ggf. Priorisierung/Planung



Grenzen

- ♦ schwierig einsetzbar in verteilten Umgebungen
 - ♦ verschiedene, konkurrierende Kunden
 - ♦ verteilte Entwicklungsteams
- ♦ anonymer Kunde, Massenmarkt
- ♦ Teamgröße max. 10 Personen
- ♦ Probleme durch Beweislast, dass Pair Programming die Produktivität steigert – Mißtrauen
- ♦ Einbettung in Unternehmenskultur, die XP nicht unterstützt
 - ♦ starre Strukturen, starre Vorgehensweisen, Ignoranz

