

Agile Softwareprozess-Modelle

Steffen Pingel

Regionale Fachgruppe IT-Projektmanagement

2003-07-03

Was bedeutet Agil?

- Beweglich, Lebhaft, Wendig
- Andere Bezeichnung: Leichtgewichtiger Prozess
- Manifesto for Agile Software Development (2001, Kent Beck et al)
 - ★ **Individuals and interactions** over processes and tools
 - ★ **Working software** over comprehensive documentation
 - ★ **Customer collaboration** over contract negotiation
 - ★ **Responding to change** over following a plan
- Bekannte Modelle und Bezeichnungen
 - ★ XP (eXtreme Programming)
 - ★ Adaptive Software Development
 - ★ Feature-Driven Development
 - ★ Pragmatic Programming
 - ★ Scrum

Was zeichnet agile Entwicklung aus?

- Anforderungen ändern sich
 - ★ Klasische Entwicklung: Hohe Kosten von späten Änderungen
 - ★ Agile Entwicklung: Gleichbleibende Kosten von späten Änderungen

XP Grundlagen (1)

- Die vier Werte
 - ★ Kommunikation (communication)
 - ★ Einfachheit (simplicity: The simplest thing that could possible work)
 - ★ Feedback
 - ★ Mut (courage)
- Die Grundprinzipien
 - ★ Unmittelbares Feedback (rapid feedback)
 - ★ Einfachheit anstreben (assume simplicity)
 - ★ Inkrementelle Veränderungen (incremental change)
 - ★ Veränderungen wollen (embracing change)
 - ★ Qualitätsarbeit (quality work)

XP Grundlagen (2)

- Weitere Prinzipien
 - ★ Lernen lehren (teach learning)
 - ★ Geringe Anfangsinvestition (small initial investment)
 - ★ Auf Sieg spielen (play to win)
 - ★ Gezielte Experimente (concrete experiments)
 - ★ Offene, ehrliche Kommunikation (open, honest communication)
 - ★ Die Intuition der Mitarbeiter für sich nutzen, nicht dagegen arbeiten (work with people's instincts, not against them)
 - ★ Verantwortung übernehmen (accepted responsibility)
 - ★ An örtliche Gegebenheiten anpassen (local adoption)
 - ★ Mit leichtem Gepäck reisen (travel light): wenig, einfach, wertvoll
 - ★ Ehrliches Messen (honest measurement)

XP Techniken

- Die Methoden
 - ★ Das Planungsspiel (planning game)
 - ★ Kurze Releasezyklen (short releases)
 - ★ Metapher (metaphor)
 - ★ Einfaches Design (simple design)
 - ★ Testen (testing)
 - ★ Refactoring
 - ★ Programmieren in Paaren (pair programming)
 - ★ Gemeinsame Verantwortlichkeit (collective ownership)
 - ★ Fortlaufende Integration (continuous integration)
 - ★ 40-Stunden-Woche (40 hour week)
 - ★ Kunde vor Ort (on-site-customer)
 - ★ Programmierstandards (coding standards)

Das Planungsspiel

- Story-Cards
 - ★ Möglichst knapp (DIN A5 oder A6 Karten)
 - ★ Kurze Beschreibung des Features
 - ★ Notizen zur geplanten Implementierung
 - ★ Geplanter Aufwand
- Der Kunde priorisiert die Story-Cards
- Wertvollste Funktionalität zuerst
- Vergleich des geplanten Aufwands mit realem Aufwand
- Story-Cards als Fortschrittsmass öffentlich sichtbar aufhängen

Kurze Releasezyklen

- Ein Release kann aus mehreren Iterationen bestehen
- Akzeptanztest zu jedem Release
- Das Release sollte für den produktiven Einsatz geeignet sein und eingesetzt werden

Testen

- Der Kunde führt Akzeptanztests durch und schreibt basierend auf den Story-Cards Funktionstests
- Die Programmierer schreiben die Komponententests selbst
 - ★ Test-Driven Development
 - ★ Test-First
- Ein Feature ist fertig implementiert, wenn einem keine Tests mehr einfallen
- Es gibt keine Features für die es keinen Test gibt
- Schlechte Tests können blenden

Refactoring

- Jedes Refactoring kann und muss in kleine Schritte zerlegt werden

Gemeinsame Verantwortlichkeit

- Minimierung des Truck-Faktors

Fortlaufende Integration

- Optimistisches Sperren vs. Pessimistisches Sperren

Programmierstandards

- Abweichungen von der Richtlinie nur in begründeten Ausnahmefällen

Fazit

- Das Prozessmodell ist nur ein Faktor für den Erfolg eines Projektes
- Jedes Modell muss für das konkrete Projekt angepasst werden
- Man darf Prozessmodelle auch mischen
- Mehr Informationen
 - ★ <http://www.agilealliance.org>

Vielen Dank für die Aufmerksamkeit.