Überblick über Extreme Programming, © Christoph Knabe, TFH Berlin, FB VI, 2007-10-07

1. Überblick über Extreme Programming

Extreme Programming (**XP**) entstand als leichtgewichtiger, iterativer Softwareentwicklungsprozess aus Kritik am überregulierten Phasenmodell. Siehe p. 12 Aufwandskurven in http://www.dpunkt.de/leseproben/2278/Kapitel_1.pdf

Grundlegende Werte: Einfachheit, Feedback, Kommunikation, Mut. Alle folgenden 12 XP-Techniken wirken im Zusammenspiel.

2. Kundenbezogene Techniken

1.1 Kunde vor Ort

Kunde (Anwender und Auftraggeber) geben fachliche Anforderungen in ihrer Sprache vor, können jederzeit dazu befragt werden.

1.2 Kurze Releasezyklen

Durch häufige Releases (alle 1-3 Wochen) mit sofortigem Einsatz
- schneller Nutzen für den Kunden
- schnelles Feedback vom Kunden für die Entwicklung.

1.3 Planungsspiel (Iterationsplanung)

Kunde formuliert Anforderungen an das nächste Release auf Story-Cards,
Entwickler schätzen deren Aufwand, **Kunde** priorisiert diese dann im Lichte der Kosten.

1.4 Metapher

Kernideen des Systems durch wenige, klare Metaphern beschreiben, wirken als Vereinheitlichung für Systementwicklung. Bsp.: Alles als Karteikarte oder Webauftritt als Baum

3. Entwicklungsbezogene Techniken

1.5 Testen

Test First: Für Komponente zuerst reproduzierbaren Testfall (z.B. mit JUnit) entwickeln, dann diese soweit ausprogrammieren, bis er fehlerfrei durchläuft.
Akzeptanztests: Durch Kunden für Release-Abnahme definieren, auch möglichst reproduzierbar.

1.6 Einfaches Design

Möglichst einfache, verständliche Struktur der Software, keine „Technologie auf Vorrat".

1.7 Refactoring

Wenn ungünstige Softwarestruktur bemerkt wird, diese unter Beibehaltung der Funktionalität beseitigen. Danach alle Komponententests durchlaufen lassen!
Ziele: Redundanzfreiheit => Optimale Änderbarkeit.

1.8 Fortlaufende Integration

Integration von Änderungen sehr häufig (mindestens täglich) jeweils nach Durchlaufen aller Komponententests. Dadurch frühe Erkennung von Integrationskonflikten und frühe Rückkopplung von anderen Entwicklern.

4. Gruppenbezogene Techniken

1.9 Programmieren in Paaren

Stets 2 Entwickler vor 1 Rechner, wechselnde Tastatur, Partner
=> Hohe SW-Qualität, -Verständlichkeit, Wissensverbreitung.

1.10 Gemeinsame Verantwortlichkeit

Der gesamte Code gehört dem Team. Jedes Paar soll jede Möglichkeit zur Codeverbesserung jederzeit wahrnehmen. Das ist kein Recht sondern eine Pflicht.
=> Keine Wartezeiten, hohe Projektsicherheit.

1.11 Programmierstandards

Nötig, damit „Programmieren in Paaren" und „Gemeinsame Verantwortlichkeit" funktionieren.

1.12 40-Stunden-Woche

XP ist sehr intensiv, Überstunden wirken dabei kontraproduktiv.
Durch iteratives Vorgehen entfällt der Termindruck zu Projektende.

Literatur:

Wolf, Rook, Lippert: eXtreme Programming. Eine Einführung mit Empfehlungen und Erfahrungen aus der Praxis, 2. Aufl., dpunkt.verlag, Heidelberg, 2005