extreme Programming

Holger Bohlmann, Urs Reupke

Holger.Bohlmann@it-agile.de
Urs.Reupke@it-agile.de

Perfekt?

it-agile

"In software development, *perfect* is a verb, not an adjective."

Kent Beck, Extreme Programming Explained

Die nächste halbe Stunde

- Warum machen wir heute alle nicht XP?
- Was ist von XP geblieben?
- Herausforderungen und Nutzen von XP-Techniken

XP ?= fail

- ‚Nur' Entwicklungspraktiken
- Kent Beck: Management nicht abgeholt

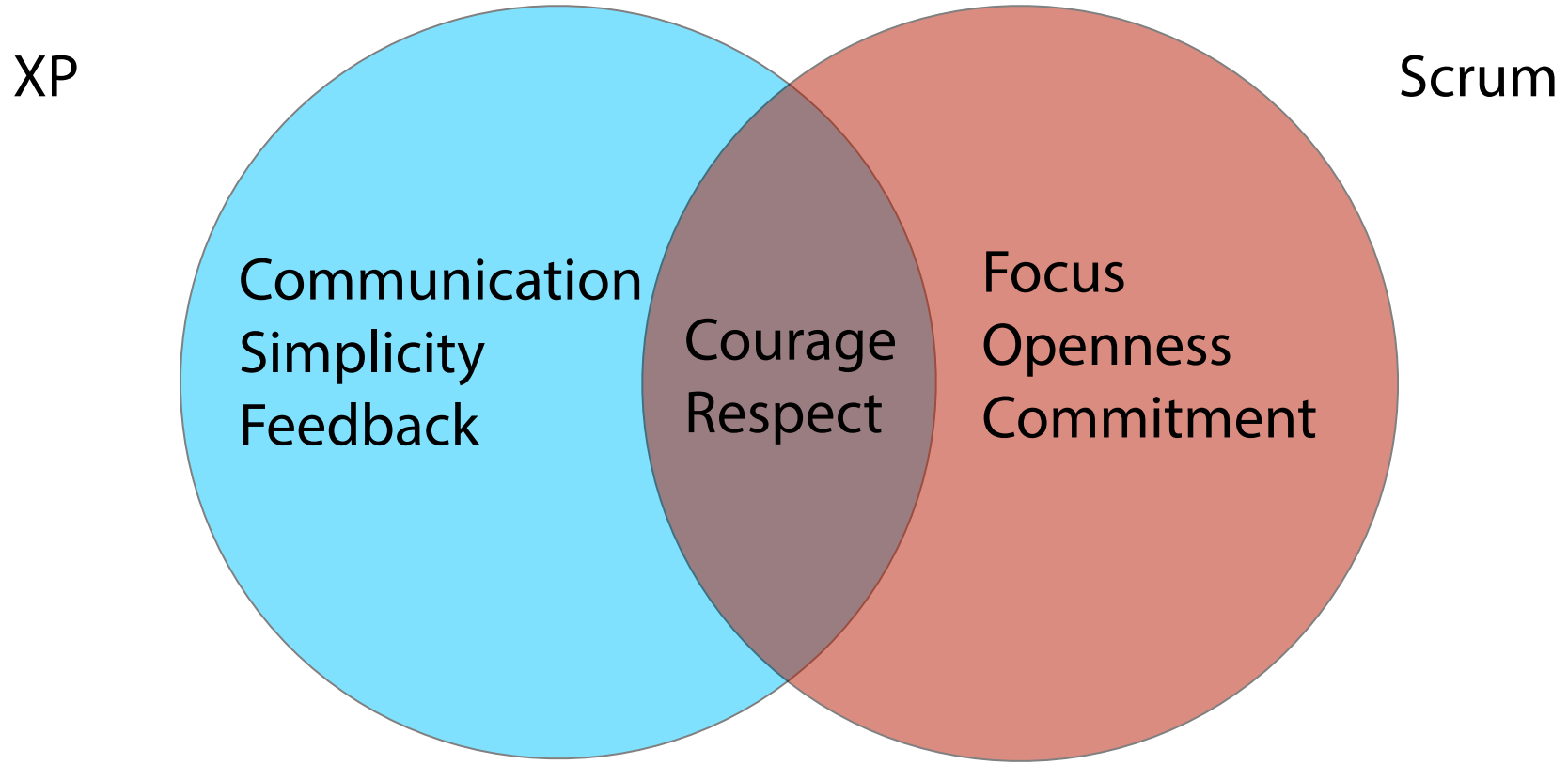
- Radikale Schritte, radikal vertreten
- Gelindert in XP2 - too little, too late?
- War der Name vielleicht abschreckend? Safe vs. Extreme

Ist Scrum genug?

- ‚Einfach' anzuwenden
- Management wird abgeholt
- Fokus auf neue Features & Speed

- Scrum gibt nur einen Rahmen:
  - Wie können wir die Geschwindigkeit halten?
  - Wie machen wir technische Schuld transparent?
  - Wie gestalten wir die Entwicklung?

Gemeinsame Werte



XP ist noch aktuell

- Scrums “CSD”-Fertigkeiten stammen aus XP
- Tools helfen bei Umsetzung, moderne Sprachen ebenso
- XP ist immer mehr in die Köpfe gesickert, Potential ist aber noch vorhanden
- Viele Praktiken sind heute etabliert

Etablierte Techniken

Daily Standup

Fortlaufende Integration (Continuous Integration)

Refactoring

Gemeinsame Verantwortlichkeit (Collective Ownership)

Pair Programming

it-agile

Hindernisse

- Effizienzdenken
- Psychologie
- Arbeitsumgebung
- Anstrengend

Belohnungen

- Wissensverteilung
- Frühes Feedback im Entwicklungsteam
- Besseres Design

Continuous Integration

it-agile

Herausforderungen

- Kleine Schritte
- Branches
- Schnelle automatische Tests
- Automatischer Build ist nicht immer einfach einzuführen

Belohnungen

- Lauffähiges System
- Frühes Feedback für Entwickler / PO / Kunden
- Schnellere Releases

Test First

it-agile

Herausforderungen

- Unterschätzte Debug-Aufwände
- Effizienzdenken
- Kapselung von „Untestbarem“
- “Wie soll ich vorher wissen, was ich testen soll?”

Belohnungen

- Regressionsfehler fallen schneller auf
- Dokumentation für den Rest des Teams
- Mit TDD: Schlankes Design, testfähige Architektur

Nachhaltiges Tempo

it-agile

Herausforderungen

- Nein-Sagen
- Fokus auf die wichtige Arbeit
- Ehrliches Tempo für sich finden

Belohnungen

- Höhere Qualität (Flüchtigkeitsfehler)
- Zufriedene Mitarbeiter
- Waste wird vermieden

Refactoring

it-agile

Herausforderungen

- Bedarf erkennen
- Effizienzdenken
- Fähigkeit für kleinschrittiges Refactoring
- Tool-Support

Belohnungen

- Flache Aufwandskurve
- Code ist leichter zu warten
- Einarbeitung fällt leichter

Whole Team

it-agile

Herausforderungen

- Entwickler arbeiten eng zusammen
- Cross-Funktionales Team aufstellen
- Kunden ins Projekt bekommen

Belohnungen

- Das Team kann alles selbst erledigen
- Nicht nur einzelne Verantwortliche
- Kurze Kommunikationswege

Abschluss

- XP hat neue Kleider, ist aber weiterhin aktuell
- Was damals radikal war, ist heute “Good Practice”

- Probiert die Praktiken aus!
- Wagt kontrollierte Experimente!
- Übernehmt was funktioniert!
- Verbessert euch kontinuierlich!
- Und überfordert euch nicht!

Fazit

it-agile

XP ist nicht perfekt, aber es hilft uns,
unsere Entwicklung zu perfektionieren.

agile Softwareentwicklung Build-Prozess Eclipse Way
Crystal Einführung eXtreme Programming Feature Driven
Development Kanban Lean Management OSGi Pair-Programming
Refactoring Retrospektiven Schätzverfahren Scrum
testgetriebene Entwicklung Überblick Unit-Tests