Extreme Hour
XP-Techniken selbst erleben

Jutta Eckstein, jeckstein@acm.org
Frank Westphal, westphal@acm.org

net.ObjectDays 2001

Extreme Programming

In fünf Worten:

- human
- leichtgewichtig
- agil
- adaptiv
- effizient

„Excuse me, I gotta go program.“

– Kent Beck

Der erste Tag der Iteration

- Der Kunde stellt seine Stories vor
- Aufkommende Fragen im Team klären
- Die Stories in Tasks zerlegen
- Verantwortung für einzelne Tasks übernehmen
- Den Aufwand der Tasks einschätzen
- Stories in die Iteration übernehmen, abhängig von der Teamgeschwindigkeit

Morgens um halb zehn

- Im Team zum Stand Up-Meeting zusammenkommen
- Einen Programmierpartner finden
- Kaffee und Kekse holen
- Die Task mit dem Kunden besprechen
- Kurze Designsession
- Losprogrammieren!

Acht Stunden programmieren

- Kurze Designsession
- Eine Reihe von Tests schreiben
- Die Tests auf einfachste Weise erfüllen
- Den Code in einfachste Form bringen und das Design verbessern
- Zur Integrationsmaschine wechseln und den neuen Code integrieren
- Wir legen eine kurze Pause ein, bevor wir neue Paare bilden

Abends um halb sechs

- Die angefangene Task entweder integrieren oder verwerfen
- Nach acht Stunden intensiver Arbeit sind wir erschöpft und gehen nach Hause.

Der letzte Tag der Iteration

- Die Abnahmetests durchführen
- Das Team präsentiert dem Kunden die neuen Stories.
- Der Kunde nimmt die Ergebnisse ab.
- Die Teamgeschwindigkeit bestimmen
- Den Abnahmetestgraph aktualisieren
- Eine Iterationsretrospektive durchführen
- Wir feiern den Erfolg!

Wertesystem

Im Kern beruht XP auf vier Werten:

- Kommunikation = Communication
- Einfachheit = Simplicity
- Feedback = Feedback
- Mut = Courage

XP benötigt Disziplin und Prinzipientreue.

Techniken

- Planungsspiel
- Kurze Releasezyklen
- Metapher
- Einfaches Design
- Testen
- Refactoring
- Programmieren in Paaren
- Gemeinsame Verantwortlichkeit
- Fortlaufende Integration
- Konstante Geschwindigkeit
- Kunde vor Ort
- Programmierstandards
- Stand Up-Meeting
- Retrospektive

Vision: Eine bessere Kaffeemaschine

- Unsere Firma konzentriert sich auf Kaffeemaschinen für Privatpersonen
- Unser Hauptprodukt soll im oberen Marktsegment angesiedelt sein

- Mit dieser Vision im Hinterkopf: Planung, Entwicklung und Qualitätssicherung unserer ersten Freigabe
- Projektzeit: Eine Stunde!

~~60~~ 70 Minuten Projekt

- 10 Min: Sammeln der User Stories und erste Architekturentwürfe
- 10 Min: Priorisierung und Aufwandsschätzung
- 10 Min: 1. Iterationsplanung
- 10 Min: 1. Iteration
- 10 Min: Refaktorisieren
- 10 Min: 2. Iterationsplanung
- 10 Min: 2. Iteration
- Freigabe!

Zeitangaben sind fix!

Personen und Rollen

- **Coach** sorgt für problemlosen Projektablauf
- **Tracker** achtet auf die Zeiten
- **Kunde** spezifiziert Anforderungen, gibt jedoch keine Schätzungen ab
- **QA** entwickelt Funktionstests
- **Entwickler** schätzen Zeitaufwände und implementieren

- **Regeln:**
  - Was nicht gezeichnet ist, gilt als nicht geliefert!
  - Was nicht geschrieben steht, gilt als nicht angefordert!
  - QA kann erst am Ende der Iteration sehen, was die Entwickler zeichnen

10 Min: User Stories, Architekturentwurf, Testrahmen

- **Kunde** schlägt Stories vor. **Tracker** hilft bei der Spezifikation
- **Entwickler** untersuchen verschiedene Architekturmöglichkeiten. (1 Stift für zwei Entwickler: Pairing!)
- **QA** definiert Testrahmen

10 Min: Schätzung von Aufwand und Priorität

- **Kunde** und **Tracker** ordnen Stories in 3 Stapel:
  - unabdingbar / Marktvorteil / wäre schon prima
  - jeden Stapel für sich nochmals nach Prioritäten sortieren
- **Entwickler** geben Idealminuten für jede Story an, basierend auf ihrer Erfahrung mit den Architekturentwürfen
  - Maximalaufwand pro Story: 5 Idealminuten
  - bei höherer Schätzung: aufteilen, Rückfrage beim Kunden
  - gibt es keine Einigung unter den Entwicklern: der Optimist gewinnt, aber das Risiko muss notiert werden

10 Min: 1. Iterationsplan

- Stories über zwei Iterationen planen
- Einplanen nach Priorität und Risiko
- Entwickler übernehmen Verantwortung für die Aufgaben

10 Min: 1. Iteration

- QA spezifiziert Funktionstest für jede Story
  - QA sieht nicht was die Entwickler zeichnen
- Entwickler zeichnen die Lösungen
  - Soviel wie nötig, so wenig wie möglich
- Kunde und Tracker überlegen sich neue - und ändern bestehende Stories

10 Min: Refaktorisieren

- QA führt die Tests durch, notiert die Fehler
- Coach und Tracker kontrollieren die *Geschwindigkeit*
- Kunde priorisiert Fehler und neue Stories
- Entwickler verbessern das Design der 1. Iteration
  - keine Redundanzen
  - soviel wie nötig, so wenig wie möglich

10 Min: 2. Iterationsplan

- Kunde kommt mit neuen Stories auf
- Entwickler schätzen Aufwände in Idealminuten
- Planung der 2. Iteration:
  - Priorität und Risiko zuerst
  - Berücksichtigen der tatsächlichen Geschwindigkeit

10 Min: 2. Iteration

- QA spezifiziert Funktionstest für jede Story
  - QA sieht nicht was die Entwickler zeichnen
- Entwickler zeichnen die Lösungen
  - Soviel wie nötig, so wenig wie möglich
- Kunde und Tracker überlegen sich neue - und ändern bestehende Stories

FREIGABE!

- QA führt Tests durch. Können wir ausliefern?
- Brauchen wir eine weitere Iteration?
- Können wir Stories neu einplanen, die nicht in diese Freigabe eingegangen sind?

Referenzen

- Extreme Hour auf dem Wiki: http://c2.com/cgi/wiki?ExtremeHour
- Bericht über Planning Game auf der OOPSLA'00: http://csis.pace.edu/~bergin/xp/planninggame.html
- Planning Extreme Programming. Kent Beck, Martin Fowler. Addison Wesley 2000.
- Extreme Programming Explained. Kent Beck. Addison Wesley 2000.