Extreme Programming

Einleitung

Extreme Programming, im Folgenden kurz *XP* genannt, ist eine kompakte Methode zur Softwareentwicklung in kleinen bis mittelgroßen Teams von zwei bis zehn Personen.

Der Name leitet sich von der extremen Umsetzung von allgemein als vernünftig eingestuften Prinzipien ab:

Prinzip	XP-Umsetzung
Code Reviews	Programmierung in Paaren.
Testen	Andauerndes Testen von Code (Komponententests vom Programmierer, Funktionstests vom Kunden).
Design	Alltägliche Überarbeitung (Refactoring).
Einfachheit	Auswählen jenes Systems, das das einfachste Design aufweist und die geforderte Funktionalität bietet.
Architektur	Andauernde Definition und Verbesserung, Einführung einer Metapher.
Integrationstests	Laufende Integration des Codes.
Kurze Iterationen	Planungsspiel (Minuten und Stunden statt Tage und Wochen).

Das Problem

Das Risiko

Bei jedem Softwareprojekt besteht das Risiko, daß die Entwicklung die Erwartungen der Geschäftsseite nicht erfüllen kann. Im Folgenden sind einige Risikobeispiele zusammen mit der XP-Strategie, dieses zu minimieren, angegeben:

- **Terminverzögerungen**

  XP verwendet Release Zyklen von maximal einigen Monaten. Innerhalb einer Release finden ein- bis dreiwöchige Iterationen von geforderter Funktionalität statt. Die Planung erfolgt in ein bis drei Tagesschritten. Zuerst wird die Funktionalität mit der höchsten Priorität implementiert.

- **Projektabbruch**

  Der Kunde wird aufgefordert, die kleinste, geschäftlich sinnvolle Variante zu wählen.

- **Unrentabilität des gelieferten Systems**

  XP fordert, daß das System laufend getestet wird und sich immer in einem erstklassigen Zustand befindet.

- **Fehlerrate**

  Es werden fortlaufend Tests vom Programmierer (Komponententests) und vom Kunden (Funktionstests) durchgeführt.

- **Die Software löst nicht das ursprüngliche Problem**

  Der Kunde ist Mitglied des Teams.

- **Das Geschäftsziel ändert sich währen der Entwicklung**

  Kurze Release Zyklen und kurzfristige Planung machen Änderungen leicht möglich.

- **Personalwechsel**

  Jeder Programmierer muss für die Aufwandsschätzung und die Fertigstellung der eigenen Aufgaben die Verantwortung übernehmen. Durch die Förderung der Kommunikation im Team wird die Integration neuer Teammitglieder erleichtert.

- **Falsche Funktionsfülle**

  Die Aufgaben mit höchster Priorität werden zuerst ausgeführt.

Die Ökonomie

Der Wert eines Softwareproduktes hängt ab von

- den ein- und ausgehende Cashflows.
- der Projektlebensdauer.
- den Zinssätzen.

Um die Faktoren möglichst optimal beeinflussen zu können, bietet XP

- genaue, häufige Feedbacks über den Arbeitsfortschritt.
- viele Möglichkeiten, die Anforderungen abzuändern.
- eine kleine Anfangsinvestition.
- die Möglichkeit einer raschen Entwicklung.
- einen Berücksichtigung der Unsicherheit hinsichtlich des möglichen Wertes einer Option (weglassen kann billiger sein als auf Verdacht einbauen !).

Die vier Variablen

Ein Softwareprojekt wird hauptsächlich durch folgende vier Variablen beeinflußt:

- **Kosten**
- **Zeit**
- **Qualität**
- **Umfang**

Das Management sollte drei Variablen festlegen, das Entwicklungsteam eines. Die Konzentration des Teams sollte sich dabei auf den Umfang richten.

Die Kosten von Änderungen

In der traditionellen SW-Entwicklung steigen die Kosten von Änderungen mit der Zeit exponentiell an. Das System wird Anpassungen gegenüber träge.

In XP steigen die Kosten mit der Zeit langsam an. Damit müssen wichtige Entscheidungen nicht am Projektanfang (oft auf Verdacht) getroffen werden, sondern können auf einen späteren Zeitpunkt verschoben werden.

Damit dies realisiert werden kann, sind folgende Vorraussetzungen günstig:

- der Einsatz einer schnittstellenorientierten, objektorientierten Technologie.
- die Entwicklung eines einfachen Designs.
- die Durchführung automatisierter Tests.
- eine Übung im Ändern des Designs.

Die Grundwerte

- Kommunikation

  - durch Programmierung in Paaren.
  - durch die Durchführung von Komponententests.
  - durch die Durchführung von Aufwandsschätzungen.

- Einfachheit

  Alle Projektmitglieder müssen den Code und das Design verstehen. Daher sind Änderungen dann leicht durchführbar.

- Feedback

  - durch die Durchführung von Komponententests.
  - durch die Durchführung von Funktionstests.
  - durch die Aufwandsschätzung von Storycards.
  - durch eine frühe Systemintegration.

- Mut

  - zu Architektur und Design Änderungen.
  - den Code zu verwerfen.

Die Grundprinzipien

- Unmittelbares Feedback
- Anstreben von Einfachheit

  Man soll nicht zuviel für die Zukunft planen und wieder verwendbare Module entwickeln sondern die unmittelbar anliegende Aufgabe gut erledigen.

- Inkrementelle Veränderungen

  Man soll kleine, aber wirkungsvolle Änderungen anstreben.

- Wollen von Veränderungen
- Qualitätsarbeit liefern

Die Arbeitsschritte

- Programmieren

  Das Programmieren ist die zentrale Tätigkeit, da der Quellcode auch das Design ausdrückt.
  Über den Quellcode wird die Kommunikation abgewickelt.
  Über den Quellcode wird gelernt, wie sich Ideen umsetzen lassen.

- Testen

  Der Test gilt als Ressource und als Verantwortung.
  Jeder Test, welcher fehlschlagen könnte, muss auch geschrieben werden.
  Es sollen automatisierte Tests entwickelt werden.
  Das Testen gibt Vertrauen in den Code.

- Zuhören

  Der Kunde erläutert, worin seiner Meinung nach das zu lösende Geschäftsproblem besteht.

- Designentwurf

  Die Logik des Systems muss organisiert werden.
  Gutes Design zeichnet sich dadurch aus, dass

  - Änderungen in einem Teil nicht weitere Änderungen nach sich ziehen.
  - jedes Logikelement an nur einer Stelle vorkommt.
  - die Logik in der Nähe der zu bearbeitenden Daten liegt.
  - Erweiterungen Veränderungen an nur einer Stelle nach sich ziehen.
  - das System nicht komplex ist.

Die Lösung

Überblick

Die Lösung besteht in der Synergie aus einfachen Verfahren, die ihre Stärke erst in der Kombination miteinander ausspielen.

- Das Planungsspiel

  Geschäftliche und technische Überlegungen werden in einem fortwährenden Dialog zwischen Möglichem und Erwünschtem diskutiert.

  Die Geschäftsseite muss entscheiden über
  - den Umfang.
  - die Prioritäten.
  - die Zusammensetzung der Versionen.
  - den Liefertermin.

  Die Entwickler müssen durchführen
  - Aufwandsschätzungen.
  - Abschätzungen von Konsequenzen.
  - die Steuerung des Prozesses.
  - eine genaue Terminplanung.

- Kurze Release Zyklen

- Entwickeln einer Metapher

  Das System soll aus einer Vogelperspektive als in sich stimmige Einheit betrachtet werden.

- Einfaches Design

  Ein Design gilt dann als einfach, wenn es
  - alle Tests besteht.
  - keinen Redundanzen enthält.
  - alles offen legt, was die Programmierer beabsichtigten.
  - die geringste, mögliche Anzahl an Klassen und Methoden besitzt.

- Testen

  Tests werden für Produktionsmethoden geschrieben, welche möglicherweise nicht funktionieren.

- Refactoring

  Folgende Fragen sollten immer gestellt werden:
  Kann das vorhandene Programm geändert werden, um eine Funktion leichter hinzuzufügen?
  Gibt es eine Möglichkeit, das Programm zu vereinfachen und gleichzeitig alle Tests positiv zu bestehen?

- **Programmieren in Paaren**

  Ein Programmierer konzentriert sich auf die Implementierung der aktuellen Methode.
  Der andere denkt strategischer:
  - Kann der gewählte Ansatz funktionieren?
  - Gibt es noch Testfälle, welche noch nicht funktionieren?
  - Gibt es eine Möglichkeit, das System so zu vereinfachen, dass sich das Problem von selbst löst?

- **Gemeinsame Verantwortlichkeit**

  Der gesamte Code fällt in die Verantwortlichkeit aller Entwickler.
  Als Konsequenz daraus kann jeder im Team jederzeit etwas am Code ändern.

- **Fortlaufende Integration**

  Jeder Satz von Änderungen (üblicherweise einige Stunden Entwicklungsarbeit) wird sofort in das System integriert und getestet.

- **Keine Überstunden**

  Die Programmierer sollen Spaß an der Arbeit haben und nicht überlastet werden.

- **Kunde vor Ort**

  Ein Kunde ist Teil des Entwicklungsteams.

- **Festlegen von Programmierstandards**

Die Planungsstrategie

Das Planungsspiel

Das Planungsspiel ist eine Abstraktion zweier Teilnehmer: der Geschäftsseite und der Entwicklung.
Das Ziel ist die Maximierung des Wertes der produzierten Software.
Die Strategie zielt darauf ab, die wertvollste Funktionalität möglichst schnell in Betrieb zu nehmen.
Die Spielelemente sind so genannte **Storycards**, auf denen die Geschäftsseite ein Leistungsmerkmal beschreibt.

Das Spiel selbst umfasst drei wesentliche Spielzüge:

- **Die Erforschung**

  Dabei gilt es herauszufinden, was das alte System geleistet hat und das neue System leisten soll.
  Der Kunde schreibt auf Storycards die gewünschten Leistungsmerkmale nieder, schätzt sie nach Priorität ein und teilt sie entsprechend auf.

- **Die Verpflichtung**

  Dabei wird entschieden, welche möglichen Anforderungen als nächstes in Angriff genommen werden.
  Die Storycards werden nach Wert und nach Risiko sortiert. Dann wird von den Entwicklern die Geschwindigkeit und dann vom Kunden der Umfang für die Versionen festgelegt.

- **Die Steuerung**

  Dabei gilt es die Entwicklung zu steuern, während die Realität den Plan formt.
  Die Entwickler planen eine Iteration in der Dauer von ein bis drei Wochen, der Kunde wählt das wichtigste Element aus.
  Eventuell muss eine Plankorrektur erfolgen, ein neues Leistungsmerkmal hinzugenommen werden und eine Neueinschätzung erfolgen.

Die Iterationsplanung

Innerhalb einer Iteration werden die gleichen Regeln wie beim Planungsspiel verwendet , wobei nur mehr die Entwickler beteiligt sind.

- **Die Erforschung**

  Dabei gilt es die Leistungsmerkmale einer Storycard auf bestimmte Aufgaben aufzuteilen und diese in so genannten **Taskcards** festzuhalten

- **Die Verpflichtung**

  Dabei übernehmen die Programmierer die Verantwortung für die Erledigung bestimmter Aufgaben. Sie schätzen die Aufgabe und ihren Belastungsfaktor ein.

- **Die Steuerung**

  Dabei wird die Aufgabe implementiert.
  Jeder Programmierer sucht sich einen Partner, schreibt zunächst die erforderlichen Testfälle und codiert dann die Aufgabe.
  Der Arbeitsfortschritt wird protokolliert, eventuell erfolgt eine Plankorrektur.
  Nach der Integration in das Gesamtsystem wird das Leistungsmerkmal mittels eines Funktionstests verifiziert.