

Testgetriebene Entwicklung (Sneed 2004)

Kennzeichen testgetriebener Entwicklung:

Testfälle ...

- werden früh aus Anwendungsfällen abgeleitet
- dienen als Baseline
- treiben den Entwurf
- treiben die Kodierung

Test-Teams treiben die Entwicklung, statt von Entwicklern getrieben zu werden

Anwendungs- und Testfälle

- Anwendungsfälle beschreiben Anforderungen
- sind jedoch oft nicht detailliert genug, um erwartetes Verhalten vollständig zu spezifizieren
- Testfälle können Anwendungsfälle hier komplementieren
- zu jedem Anwendungsfall sollte es mindestens einen Testfall geben
- Testfälle können zur Kommunikation zwischen Entwickler und Kunden/Anwender dienen

Testfälle dienen als Baseline:

- definieren die Vorbedingungen einer Produktfunktion
- spezifizieren die Argumente einer Produktfunktion
- spezifizieren das Verhalten einer Produktfunktion (die Nachbedingungen)

Testfälle treiben den Entwurf:

- Testfall ist ein Pfad durch die Software-Architektur
- Testfälle verknüpfen Anforderungsspezifikation und Architekturkomponenten
- Testfälle können zur Studie der zu erwartenden Performanz und anderer Systemattribute zur Entwurfszeit verwendet werden

Testfälle treiben die Kodierung:

- Vor Implementierung einer Klasse werden erst Testtreiber entwickelt
- Testtreiber enthält mindestens einen Test für jede nichttriviale Methode
- Testfall hilft, die richtige Schnittstelle zu definieren
- Testfall zwingt Entwickler, über das zu erwartende Resultat nachzudenken
- Testfälle spezifizieren die Methoden zumindest partiell

Testgetriebene Entwicklung (Sneed 2004)

Tester treiben die Entwicklung:

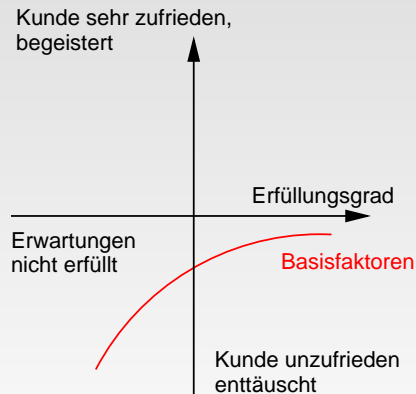
- Tester sind verantwortlich für die Auslieferung des Produkts
- Tester legen Kriterien für die Auslieferbarkeit fest
- Entwickler sind Lieferanten für die Tester
- Softwareentwicklung ist eine Versorgungskette; das Ende zieht, anstatt gedrückt zu werden

- bewusstes Wissen (20-30%)
 - Wissen, über das man sich im Klaren ist oder das in seiner vollen Bedeutung klar erkannt wird
- unbewusstes Wissen ($\leq 40\%$)
 - Wissen, das sich dem Bewusstsein im Moment nicht darbietet, aber dennoch handlungsbestimmend ist, und potenziell aufgerufen werden kann
- unterbewusstes Wissen
 - unbekannte Wünsche, die erst von außen herangetragen werden müssen, um als Anforderungen erkannt zu werden

Basisfaktoren

- Minimalanforderungen
- Mangel führt zu massiver Unzufriedenheit
- mehr als Zufriedenheit ist nicht möglich

Kano-Modell



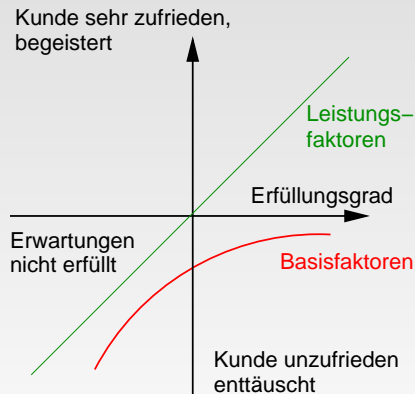
Basisfaktoren

- Minimalanforderungen
- Mangel führt zu massiver Unzufriedenheit
- mehr als Zufriedenheit ist nicht möglich

Leistungsfaktoren

- bewusst verlangte Sonderausstattung
- bei Erfüllung: Kundenzufriedenheit
- sonst: Unzufriedenheit

Kano-Modell



Basisfaktoren

- Minimalanforderungen
- Mangel führt zu massiver Unzufriedenheit
- mehr als Zufriedenheit ist nicht möglich

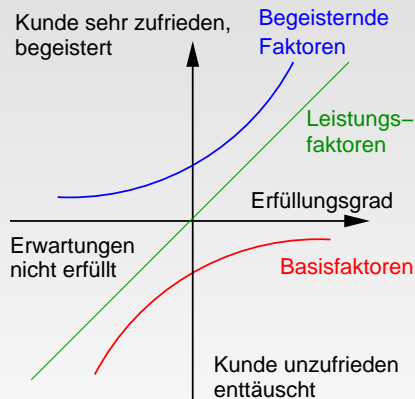
Leistungsfaktoren

- bewusst verlangte Sonderausstattung
- bei Erfüllung: Kundenzufriedenheit
- sonst: Unzufriedenheit

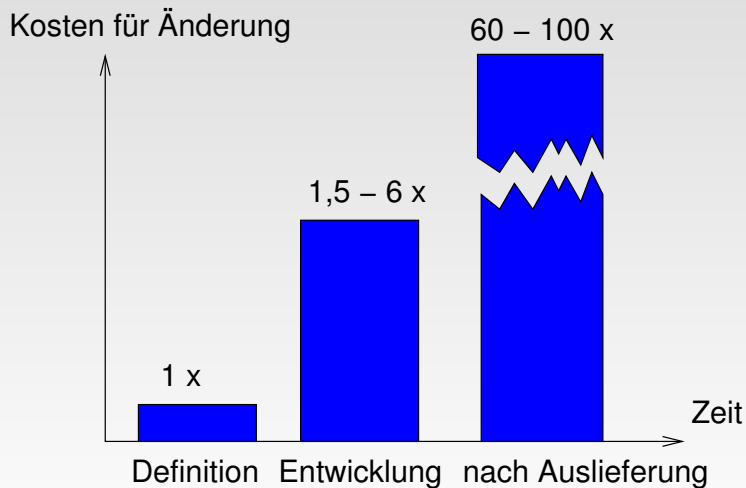
Begeisternde Faktoren

- unbewusste Wünsche, nützliche/angenehme Überraschungen
- steigern Zufriedenheit überproportional

Kano-Modell

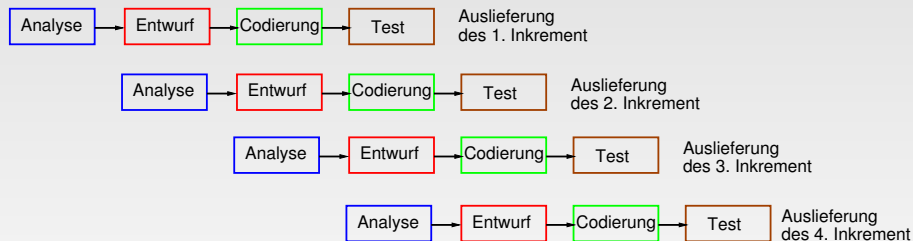


Kosten für Änderungen



Pressman (1997)

Inkrementelles Modell von Basili und Turner (1975)



Iterationen:

- ① grundlegende Funktionalität
 - Datei-Management, Editor, Textausgabe
- ② erweiterte Funktionalität
 - Style-Files, Bearbeitung mathematischer Formeln, Einbinden von Graphiken
- ③ zusätzliche Funktionalität
 - Rechtschreibprüfung, Grammatiküberprüfung, Überarbeitungsmodus
- ④ ergänzende Funktionalität
 - Tabellenkalkulation, Geschäftsgraphiken, E-Mail, Web-Browser, Scanner-Anbindung, Flipper

Inkrementelles Modell von Basili und Turner (1975)

Eigenschaften:

- Wartung wird als Erstellung einer neuen Version des bestehenden Produkts betrachtet.

Inkrementelles Modell von Basili und Turner (1975)

Eigenschaften:

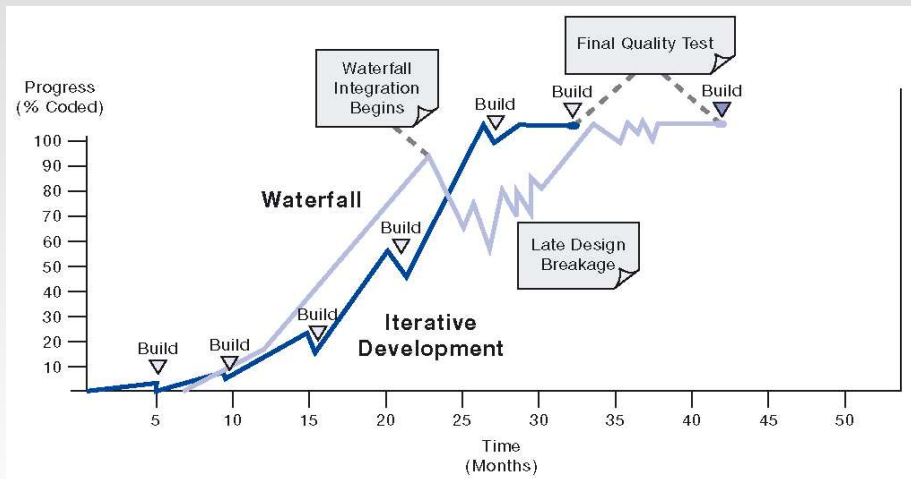
- Wartung wird als Erstellung einer neuen Version des bestehenden Produkts betrachtet.
- + Entwicklung erfolgt stufenweise
 - brauchbare Teillösungen in kurzen Abständen
- + Lernen durch Entwicklung und Verwendung des Systems.
- + Gut geeignet, wenn Kunde Anforderungen noch nicht vollständig überblickt oder formulieren kann.
 - „I know it when I see it“

Inkrementelles Modell von Basili und Turner (1975)

Eigenschaften:

- Wartung wird als Erstellung einer neuen Version des bestehenden Produkts betrachtet.
- + Entwicklung erfolgt stufenweise
 - brauchbare Teillösungen in kurzen Abständen
- + Lernen durch Entwicklung und Verwendung des Systems.
- + Gut geeignet, wenn Kunde Anforderungen noch nicht vollständig überblickt oder formulieren kann.
 - „I know it when I see it“
- Kernanforderungen und Architekturvision müssen vorhanden sein.
- Entwicklung ist durch existierenden Code eingeschränkt.

Vergleich inkrementelles Modell und Wasserfallmodell



Spiralmodell von Boehm (1988)

Mehrere Iterationen der folgenden Schritte:

- ① **Bestimmung der Ziele** und Produkte des Durchlaufs;
Berücksichtigung von Alternativen (z.B. Entwurfsvarianten) und Restriktionen (z.B. Zeitplan)
- ② **Bewertung der Risiken** für alle Alternativen; Entwicklung von Lösungsstrategien zur Beseitigung der Ursachen
- ③ **Arbeitsschritte durchführen**, um Produkt zu erstellen
- ④ Review der Ergebnisse und **Planung** der nächsten Iteration

(Generische) Risiken:

- Ist das Konzept schlüssig? Kann es aufgehen?
- Was sind die genauen Anforderungen?
- Wie sieht ein geeigneter Entwurf aus?



Bewertung Spiralmodell

- Meta-Modell: Iterationen können beliebigen Modellen folgen
- + bei unübersichtlichen Ausgangslagen wird die Entwicklung in einzelne Schritte zerlegt, die jeweils unter den gegebenen Bedingungen das optimale Teilziel verfolgen
 - schwierige Planung (was jedoch dem Problem inhärent ist)
 - setzt große Flexibilität in der Organisation und beim Kunden voraus

Rational Unified Process (RUP) nach Gornik (2001)

