Vorlesungsveranstaltung

– Softwareengineering –

Dipl.-Inform. Adriano Gesué

Gesue@t-online.de

Vorlesungsübersicht

- Inhalte
- Einführung
- Software und Softwareengineering
- Vorgehensmodelle
- Klassische Entwurfsmethoden
- Objektorientierung und UML
- Datenorganisation
- Praktische Programmierung

Was ist Software-Engineering ?

Anwendung von

- Prinzipien,
- Vorgehensmodellen,
- Methoden und Werkzeugen

während des gesamten Softwarelebenszyklus

Was ist Software-Engineering ?

bietet

- Management Theorie
- Organisation Methoden
- Werkzeuge Techniken

Was ist Software-Engineering ?

Ergebnis Softwareprojekte

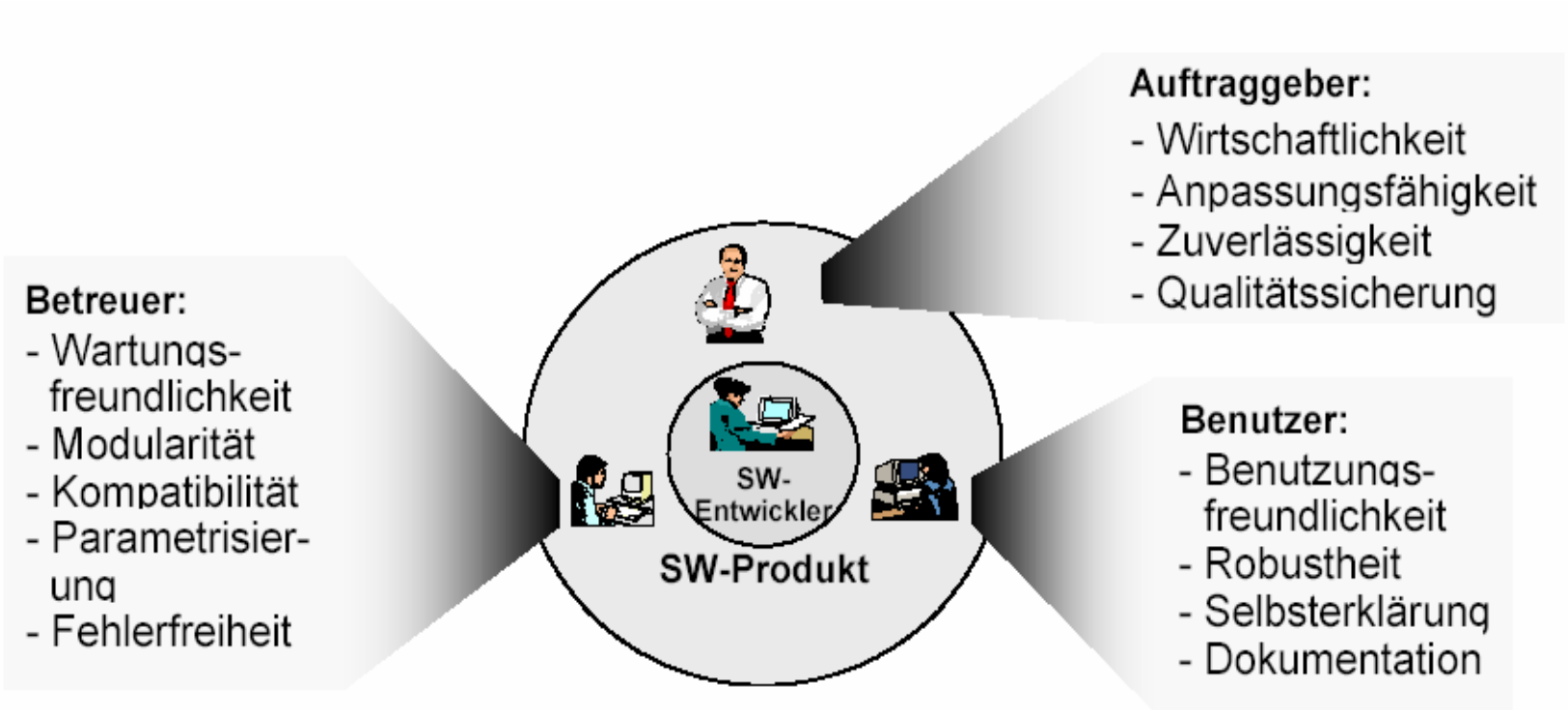
- Ziel: Einhalten von Qualitätsstandards
- Vorbild: klassische Ingenieurdisziplinen wie Maschinenbau

Problem:

Software ist unstetig:

- Fehlt an einer Brücke eine Schraube, so stürzt sie noch nicht ein.
- Stimmt in einem Programm 1 Bit nicht, kann das schon die Katastrophe bedeuten

Softwareanforderungen



Quellen: Balzert, Schwarz

Prinzipien des Software-Engineering

- Modularisierung
- Strukturierung
- Reduktion
- Kapselung

Einige bekannte Software-Katastrophen

Ein schlimmes Beispiel

Röntgengerät verstrahlt 6 Menschen -- 3 Tote (1987)
Therac-25: medizinisches Bestrahlungsgerät (Röntgen- und Elektronenstrahlen) Nachfolger von Therac-6 und Therac-20
Therac-6 und Therac-20 benutzten Hardware-Sicherungen gegen zu hohe Strahlungsdosis Therac-25 realisierte Sicherungen in Software

Fehlerbeschreibung

Wenn

der Operator die ursprünglichen Strahlungsart und Dosis eingab, der Operator dann aber innerhalb von 8 Sekunden die Strahlungsart oder Dosis änderte, wurde der Magnet nicht zurückgesetzt falsche Strahlungsdosis Fehlermeldungen traten sporadisch auf, waren aber unverständlich da die Herstellerfirma die Fehler zunächst nicht reproduzieren konnte, nahm sie sie lange nicht ernst. Außerdem: Die Vorgänger Therac-6 und Therac-20 waren bewährt

Ärgerlicher Beispiel

- **Intel Pentium: 1/824633702441.0 um 0.00005 zu groß (1994)**
- Intel benötigte 1 Jahr, um den Fehler durch Zufallstesten (= Testen mit zufälligen Werten) zu finden

Lustiger Beispiel

Wahlbeteiligung von 104% in Neu-Ulm (1994)

”Bei der Wahl des Oberbürgermeisters in Neu-Ulm 1994 wurde zunächst eine Wahlbeteiligung von 104% ermittelt. Später mußte man feststellen, daß sich in die Auswertungssoftware ein mysteriöser Faktor 2 eingeschlichen hatte.“

Software-Entwicklung als Problem

- Ohne Software geht heute nichts mehr
- Die Entwicklung von Software wird nur ungenügend beherrscht
- Software-Kosten domieren die Kosten von Informatiksystemen

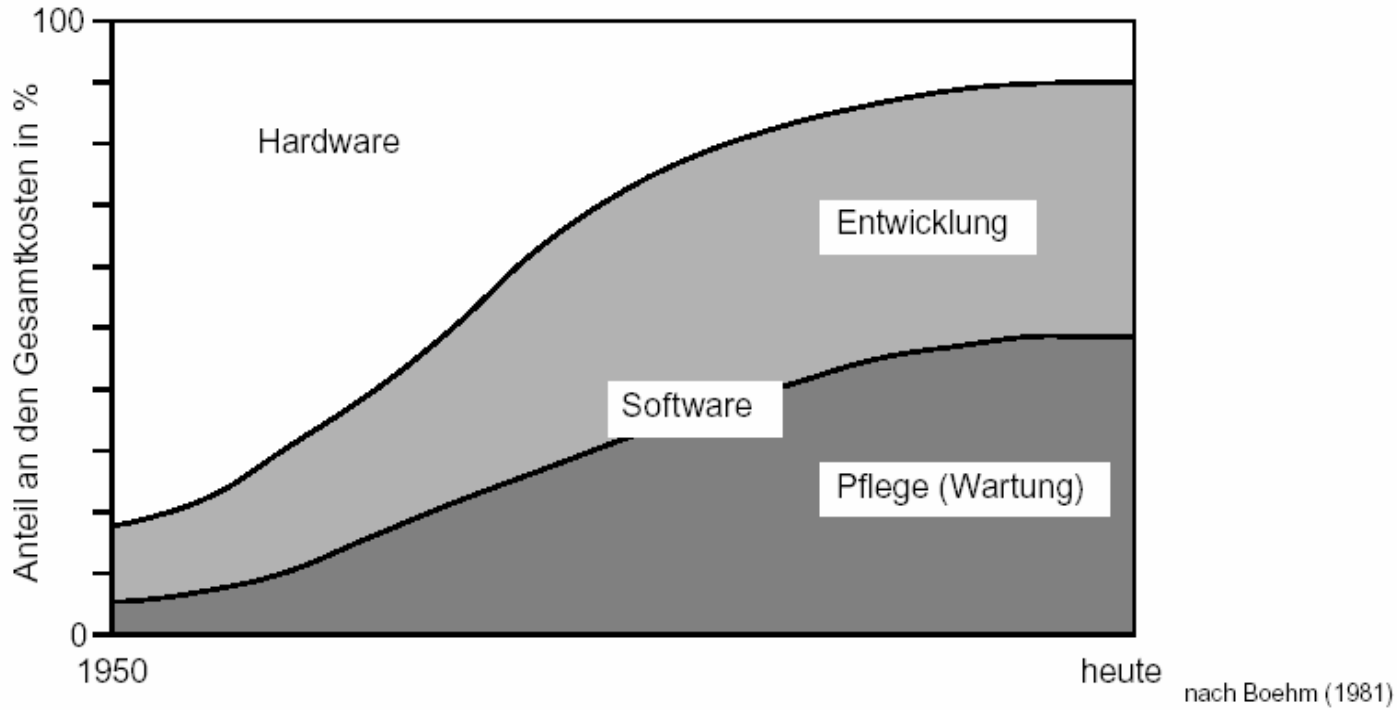
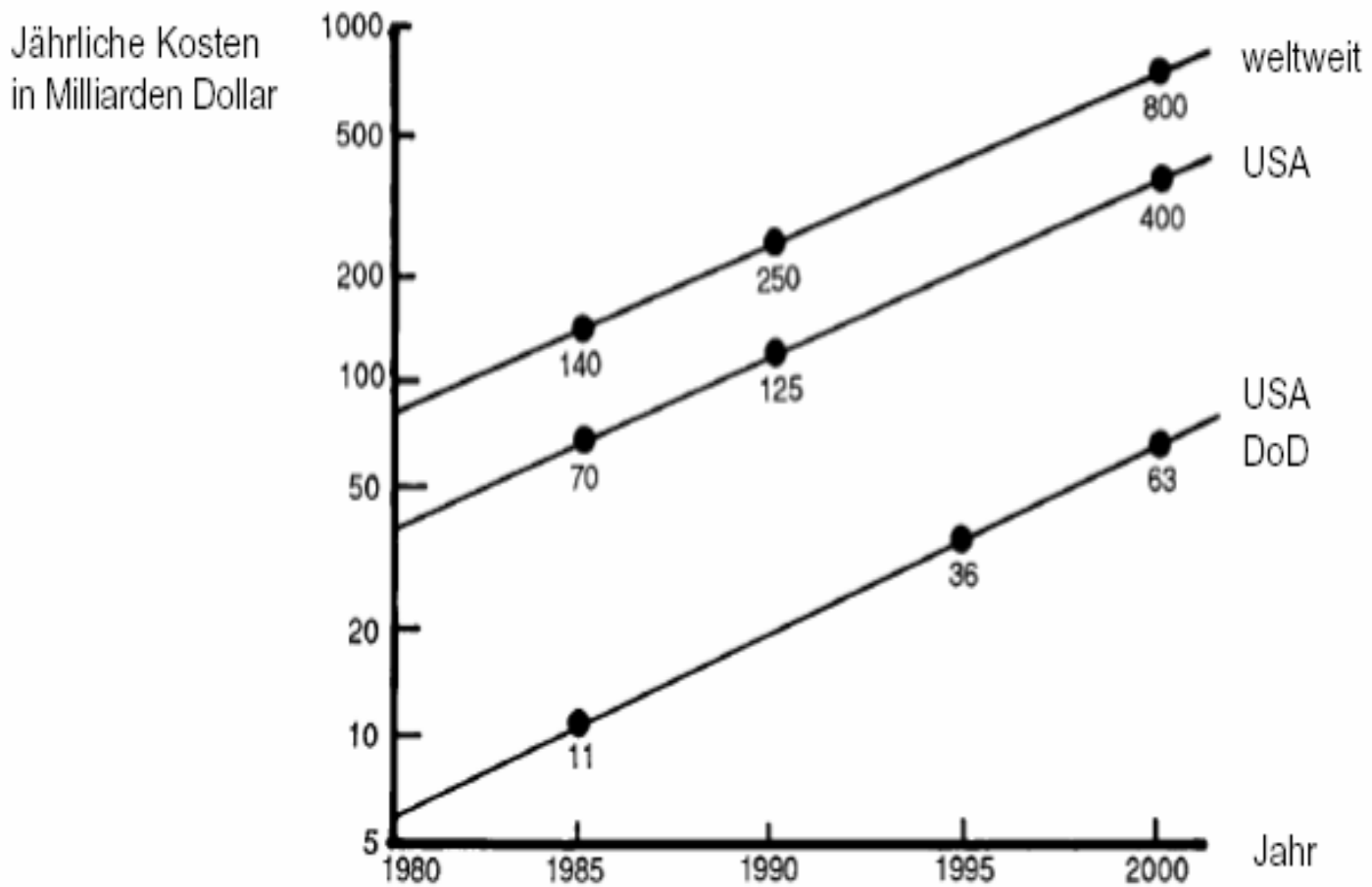


BILD 1.1. Entwicklung des Kostenverhältnisses von Software und Hardware

Weltweit werden horrende Summen für Software ausgegeben



nach Boehm (1987)

BILD 1.2. Tendenzen im Wachstum der Software-Kosten

Verteilung des Aufwands über die Lebensdauer

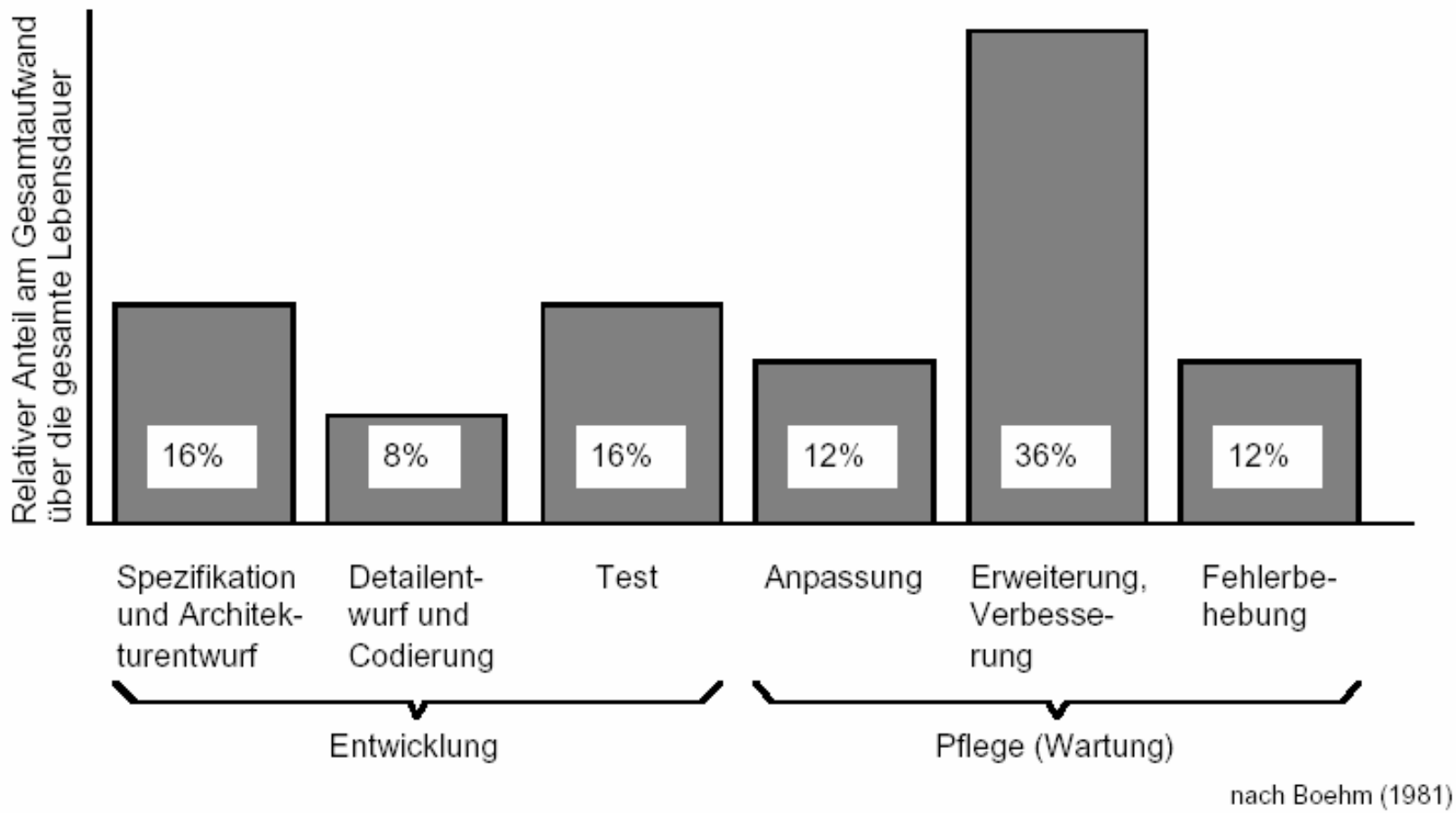


BILD 1.4. Verteilung des Aufwands über die Lebensdauer eines Software-Produkts

Wozu dient Software?

- **Problem lösen**

- **Automatisierung oder Unterstützung von**
  - **menschlicher Arbeit**
  - **technischen Abläufen**
- **steht in ständiger Wechselwirkung mit**
  - **Arbeitsprozessen**
  - **Produktionsprozessen**
  - **Menschen**

Software

DEFINITION

Software. Die Programme, Verfahren, zugehörige Dokumentation und Daten, die mit dem Betrieb eines Computersystems zu tun haben (IEEE 610.12). Diese Definition erlaubt uns, drei wichtige Feststellungen über Software zu machen, die wesentliche Hinweise darauf geben, warum die Entwicklung von Software schwierig ist.

FESTSTELLUNG

Software umfasst erheblich mehr als nur Programme.

Software-Entwicklung

Definition

Software-Entwicklung umfasst alle Tätigkeiten und Ressourcen, die zur Herstellung Software notwendig sind.

Umfasst Spezifikation der Anforderungen,
Konzept der Lösung, Entwurf und Programmierung
der Komponenten, Zusammensetzung der Komponenten
und ihre Einbindung in vorhandene Software,
Inbetriebnahme der Software sowie Überprüfung
des Entwickelten nach jedem Schritt.

Einige grundlegende Gesetzmäßigkeiten

Regel

Der Aufwand für die Erstellung von Software steigt mit wachsender Produktgröße überproportional an.
Nach Boehm (1981) wird der Zusammenhang beschrieben durch eine Gleichung der Art

$$A = bP^m$$

Dabei ist A der Aufwand, P die Größe der Software, b ein konstanter Faktor und m ein Exponent im Bereich zwischen 1,05 und 1,2. Die Regel ist eine der wenigen, welche empirisch erhärtet sind (vgl. Boehm (1981)).

Wachstum und Quantensprünge beim Kommunikationsbedarf

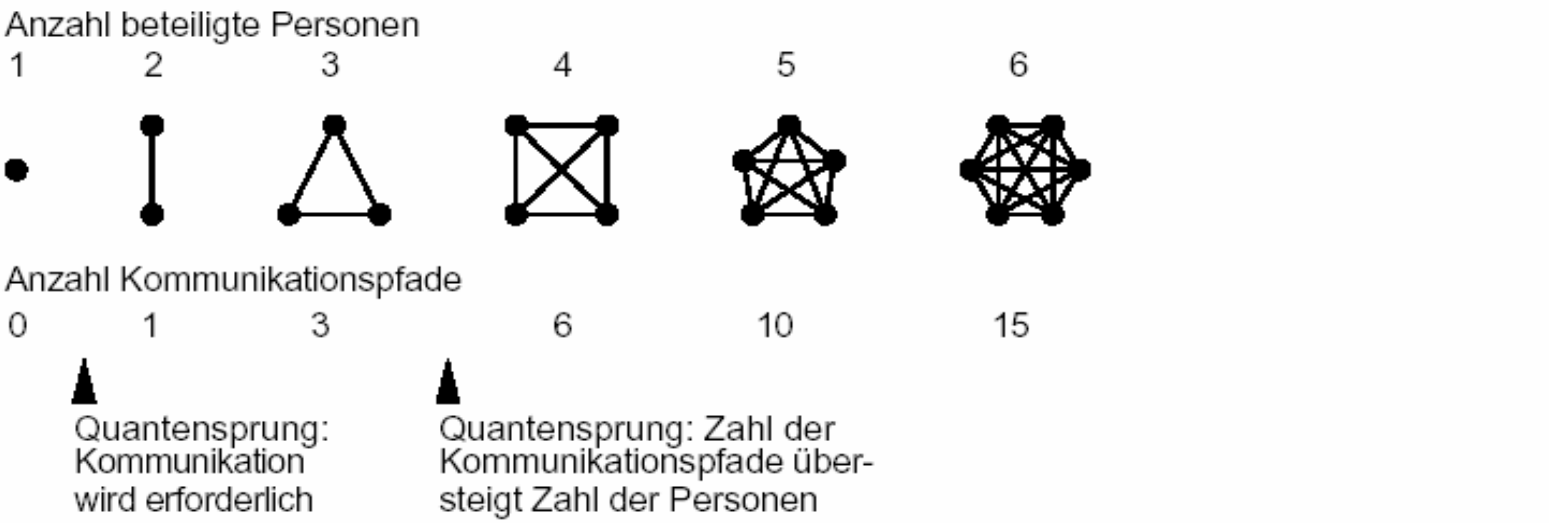


BILD 1.6. Wachstum und Quantensprünge beim Kommunikationsbedarf

Software Engineering

- Software Engineering ist das technische und planerische Vorgehen zur systematischen Herstellung und Pflege von Software, die zeitgerecht und unter Einhaltung der geschätzten Kosten entwickelt bzw. modifiziert wird.(Fairley 1985)
- Software Engineering. Die Anwendung eines systematischen, disziplinierten und quantifizierbaren Ansatzes auf die Entwicklung, den Betrieb und die Wartung von Software, das heißt, die Anwendung der Prinzipien des Ingenieurwesens auf Software. (IEEE 610. 12).

Ziele

Mit Software Engineering werden drei grundsätzliche Ziele verfolgt.

- **Die *Produktivität* bei der Herstellung von Software soll gesteigert werden.**
- **Die *Qualität* der erstellten Software soll verbessert werden.**
- **Die *Durchführbarkeit* von Software-Entwicklungsprojekten soll erleichtert werden.**

Mittel des Software-Engineerings und ihre Wirkung

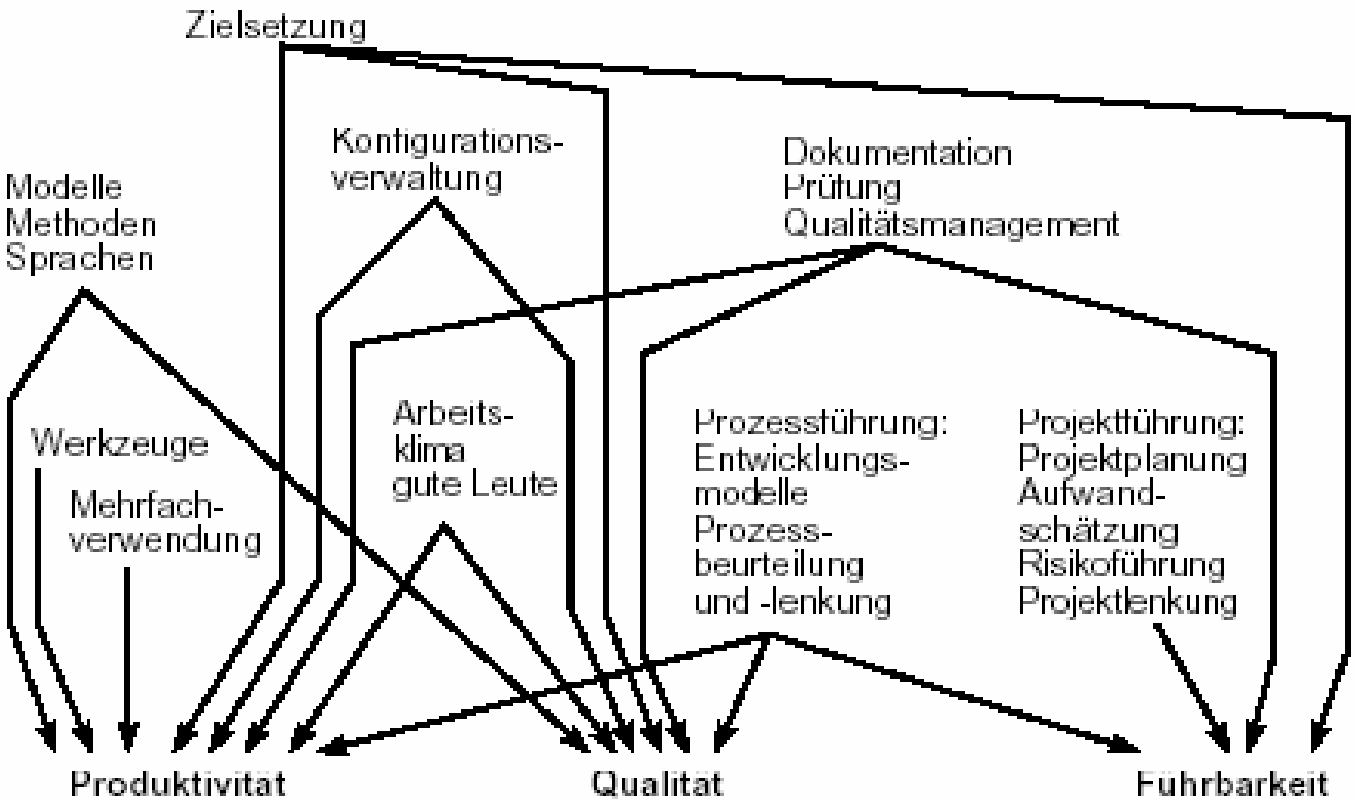


BILD 1.8. Mittel des Software Engineerings und ihre Wirkung

Der Software-Prozess

Software-Prozesse beschreiben den Ablauf der Entwicklung und der pflege von Software

DEFINITION

Prozess. *Eine Folge von Schritten, die zur Erreichung eines gegebenen Zwecks ausgeführt wird. (IEEE 610.12).*

Software-Projekte

DEFINITION

Projekt. *Eine zeitlich befristete Organisationsform zur Bearbeitung einer gegebenen Aufgabe. Dient zur Regelung der Verantwortlichkeiten und zur Zuteilung der für die Arbeit notwendigen Ressourcen**.***

Der Software-Prozess

Meilensteine

Meilensteine sind das wichtigste Mittel, um den Ablauf eines Prozesses zu strukturieren und den Fortschritt wirksam zu kontrollieren.

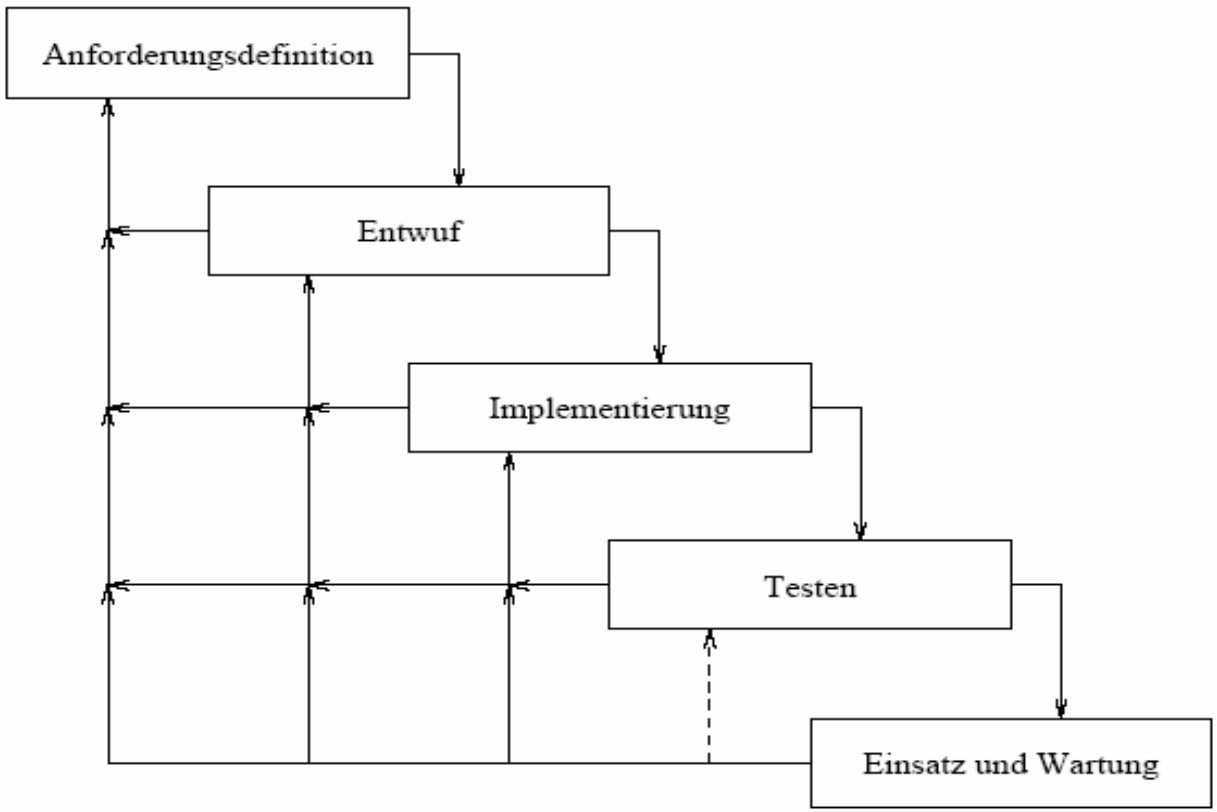
DEFINITION

Ein Meilenstein ist eine Stelle in einem Prozess, an dem ein geplantes Ergebnis vorliegt.

Der Software-Prozess

Ergebnis in einem Software-Prozess	Eignung für Meilenstein
Codierung zu 90% beendet	unbrauchbar, da nicht messbar, sondern nur schätzbar
Die Komponente xyz hat internen Abnahmetest bestanden	geeignet, da messbar (an Hand des Abnahmetest-Protokolls)
Die Anforderungsspezifikation liegt vor	geeignet, wenn Inhalt und Form des Dokuments durch ein Review überprüft werden

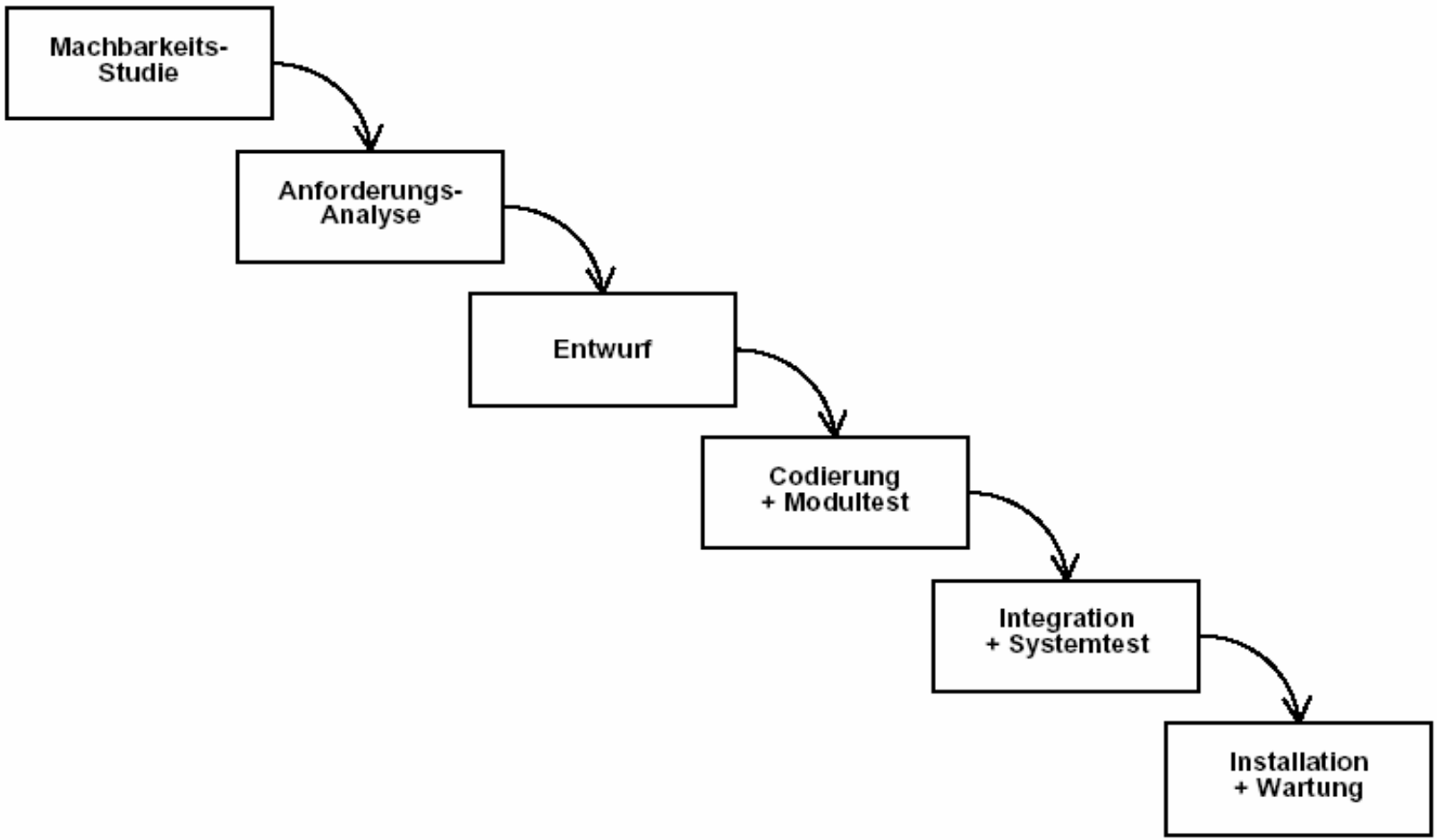
Der Software-Prozess



Prozeßmodelle

- Ein Prozeßmodell bietet eine Anleitung für den Entwickler, welche Aktivitaten als nächstes kommen.

Wasserfallmodells



Eigenschaften des Wasserfallmodells

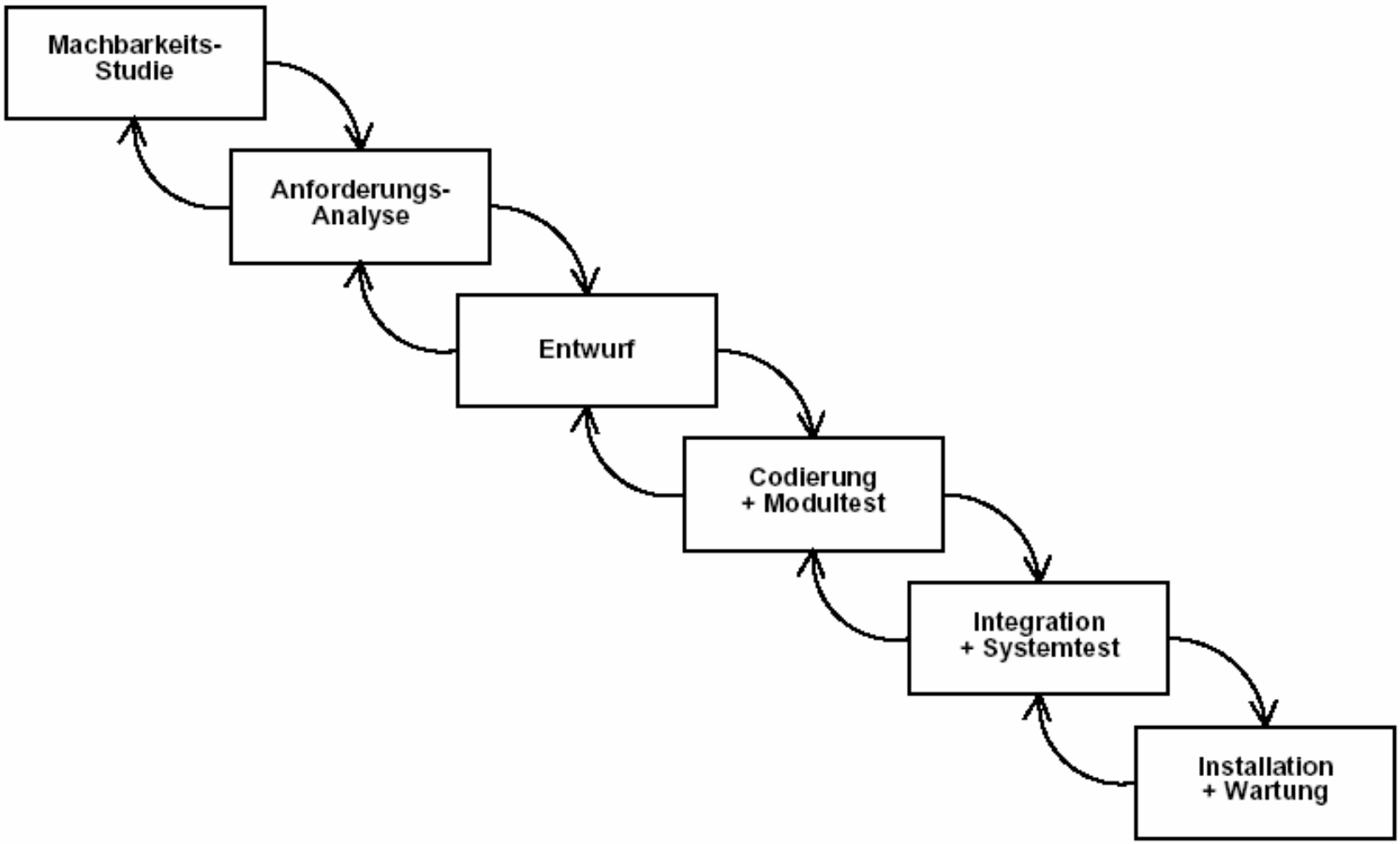
- Der Entwicklungsprozeß wird in *Phasen* zerlegt
- Eine Phase muß *abgeschlossen* sein, bevor die nächste beginnt
- Jede Phase produziert ein *Dokument* (auch ein Programm)

Die Phasen können in *Unterphasen* unterteilt werden

Wasserfallmodells mit Rückkopplung

- In der Praxis kann man nie eine strenge Trennung der Phasen erreichen; dies ist nur der Idealfall.
- Deshalb: Möglichkeit, in frühere Phasen zurückzukehren (etwa wenn Fehler oder Inkonsistenzen entdeckt wurden)

Wasserfallmodell mit Rückkopplung



Probleme mit dem Wasserfallmodell

- Am Projektanfang sind nur *ungenaue* Schätzungen der Kosten und Ressourcen möglich.
- Ein Pflichtenheft, egal wie sorgfältig erstellt, kann nie den Umgang mit dem fertigen System ersetzen. Dies ist für viele Kunden ein Problem.
- Es gibt Kunden, mit denen man keine präzisen Pflichtenhefte erstellen kann, *weil sie nicht wissen was sie wollen.*

Probleme mit dem Wasserfallmodell

- Oft werden die endgültigen Anforderungen erst nach einer gewissen *Experimentierphase* deutlich.
- Wahrnehmung des Wandels (von Anforderungen) wird überhaupt nicht unterstützt; stattdessen werden die Anforderungen frühzeitig eingefroren.
- Am Ende jeder Phase muß ein *Dokument* abgeliefert werden. Dies kann zu einer Papierflut und überbordender Bürokratie führen, ohne daß die Qualität profitiert.

Evolutionäres Modell

Ein Produkt wird als Folge von Approximationen realisiert. Jede Annährung entsteht durch Änderungen/Erweiterungen aus der vorangegangenen Version. Einige oder sogar alle Phasen des Lebenszyklus können evolutionär realisiert werden.

Prototyping

Software-Prototypen sind Approximationen an das endgültige System mit

- reduziertem Funktionsumfang
- reduzierter Benutzerschnittstelle
- reduzierter Leistung

Spezialfälle des Prototyping

Rapid Prototyping:

Verwendung von Generatoren, ausführbaren Spezifikationssprachen oder Skriptsprachen

- ➓ Vorteil: sehr schnelle Realisierung, frühzeitige Validierung der funktionalen Spezifikation
- ➓ Nachteil: u.U. ineffizient, schlechte Benutzerschnittstelle, bei Skriptsprachen: schlechte Systemarchitektur

Spezialfälle des Prototyping

Horizontaler Prototyp:

Nur eine Systemschicht, z.B. hohle

Benutzerschnittstelle ohne echte Funktionalität

- Vorteil: frühzeitiges Einbinden der zukünftigen Benutzer; Validierung, der Spezifikation

- Nachteil: nur für Demos benutzbar

Spezialfälle des Prototyping

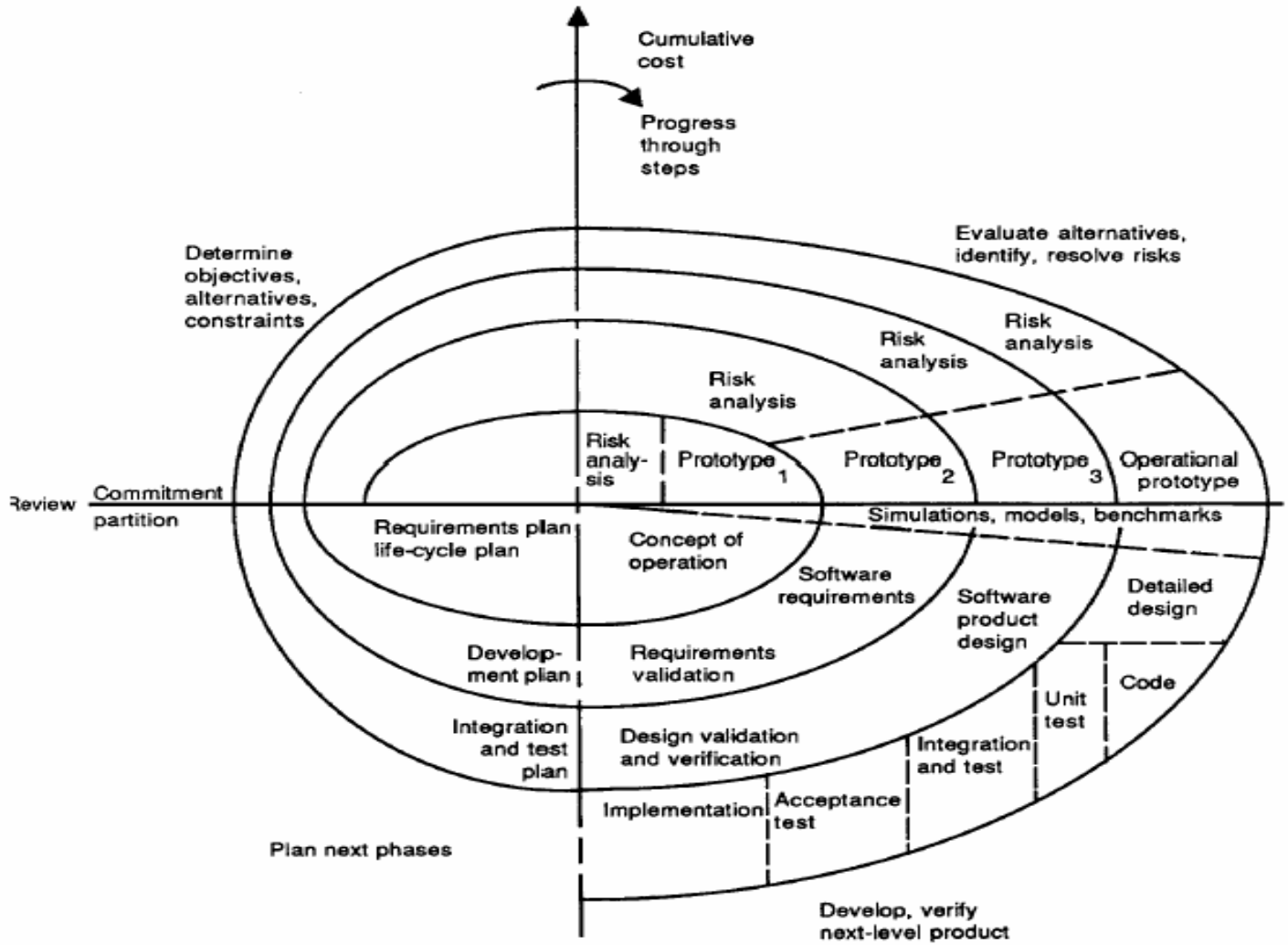
Vertikaler Prototyp:

Abgemagerter Funktionsumfang wird durch alle Ebenen implementiert

- Vorteil: frühzeitige Validierung technischer Systemeigenschaften; frühzeitige Auslieferung eines Kernsystems

- Nachteil: teuer

Spiralmodell



Spiralmodell

Das Spiralmodell ist ein *Metamodell*, das evolutionäre Aspekte und Risikobewertung umfasst. Jeder Umlauf der Spirale entspricht dem nächsten Prototyp;

Winkel ~ Zeit, Radius ~ Kosten

Spiralmodell

Jeder Umlauf zerfällt in 4 Hauptphasen (Quadranten):

1. Anforderungsdefinition
2. Bewertung von Alternativen und Risiken
3. Entwurf, Implementierung, Test
4. Auswertung und Planung des n¨achsten Umlaufs

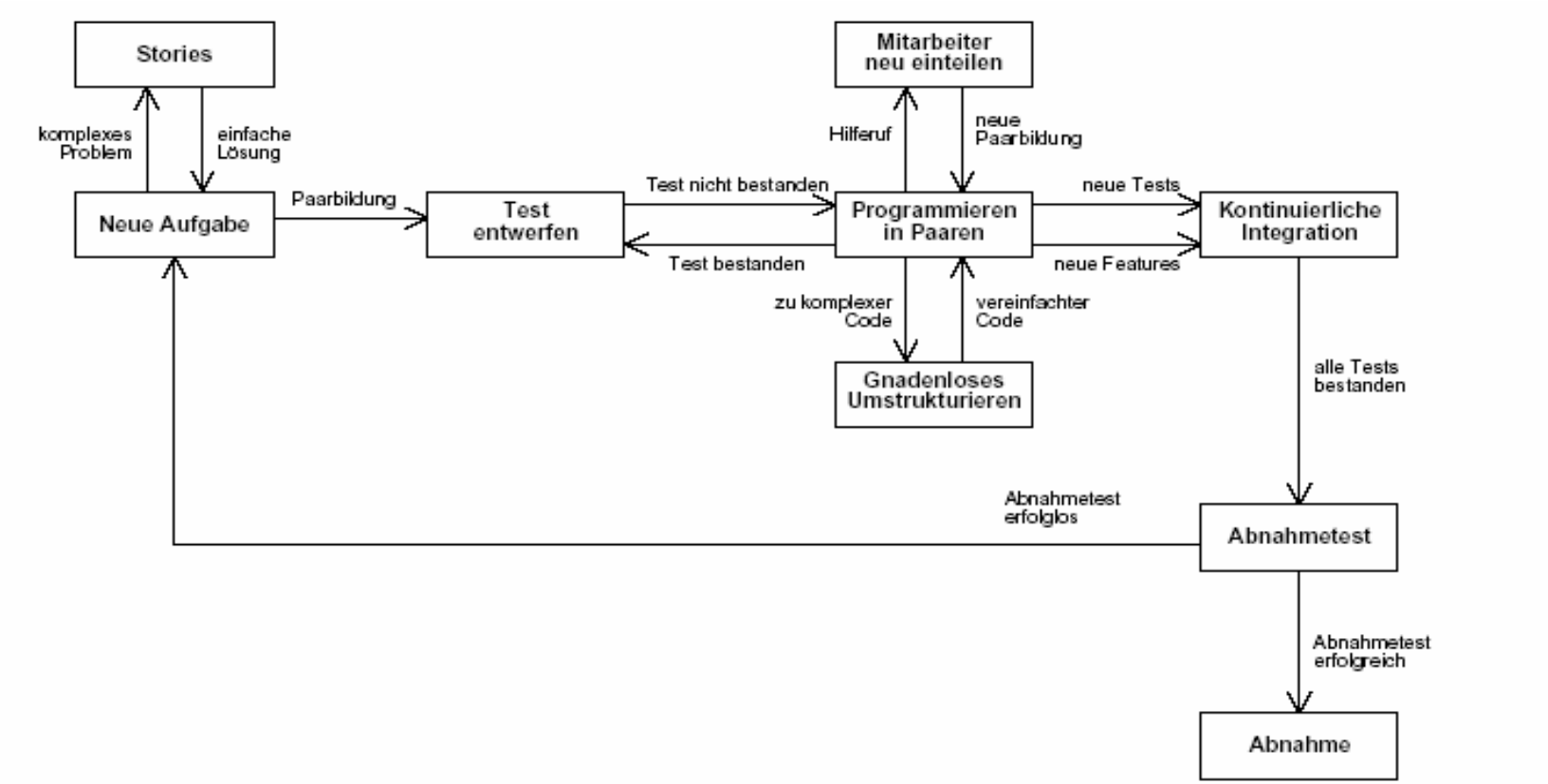
Extreme Programming

Extreme Programming (”XP“) ist ein Prozeß zum *schnellen Programmerstellen in unsicherem Umfeld.*

Antwort auf übermäßige *Prozeßmodellierung* der 90er.

- Arbeiten in kleinen Gruppen (max. 10 Personen)
- Kurzfristige Planung (bis zum n¨achsten Termin)
- Konzentration auf das, was gemacht werden muß
- Flexibilität ist Trumpf!

Das XP-Vorgehensmodell



Eigenschaften des XP-Vorgehensmodells

- *Stories* (Szenarien) treiben den Entwicklungsprozeß:
- Eine Story beschreibt einen konkreten Vorfall, der für den Betrieb des Produktes notwendig ist.
- Zu jeder Story wird ein Testfall entworfen (der die Funktionalität der Story abdeckt)
- Die Entwicklung ist beendet, wenn alle Testfälle erfüllt sind

Eigenschaften des XP-Vorgehensmodells

- *Pair Programming* (Programmieren in Paaren) sorgt für ständiges Gegenlesen durch den Partner
- Teams werden stets neu zusammengestellt
- Auch der Kunde kann so beteiligt sein
- *Ständiges automatisches Testen* sorgt dafür, daß Funktionalität erhalten bleibt
- Für jede neue Funktionalität gibt es einen Testfall
- Testfälle werden *vor* dem Programm geschrieben
- Jeder kann jederzeit die Software *umstrukturieren*
- Software muß entsprechend wandlungsfähig sein
- Code-Qualität muß auf hohem Niveau bleiben

Einsatzbedingungen XP

Extreme Programming ist *geeignet*. . .

- wenn die Anforderungen *vage* sind
- wenn die Anforderungen *sich schnell ¨andern*
- wenn das Projekt klein bis mittelgroß ist (10–12 Programmierer)

Einsatzbedingungen XP

Extreme Programming ist *ungeeignet*. . .

- wenn es auf *beweisbare* Programmeigenschaften ankommt
- wenn späte Änderungen zu teuer werden
- wenn häufiges Testen zu teuer ist
- wenn das Team zu groß oder nicht an einem Ort ist

Vor- und Nachteile XP

Positiv

- Testfälle vor dem Codieren schreiben
- Programmieren in Paaren

Umstritten

Kein Entwurf

Keine externe Dokumentation

Nachteile XP

Negativ

- Erschwerte Wiederverwendung
- Nur für kleine, hochqualifizierte Teams geeignet
- Erst wenig Erfahrung vorhanden

Lastenheft

- grobe Produktbeschreibung

1. Zielbestimmung
2. Anwendungsbereiche und Zielgruppen
3. Produktfunktionen
   Hauptfunktionen aus Auftraggebersicht
   (keine Details!)
4. (wichtigste) zu speichernde Daten
5. Leistungsanforderungen
   bzgl. Laufzeit, Platzbedarf, Genauigkeit, ...
6. Qualitätsanforderungen
   bzgl. Zuverlässigkeit, Benutzbarkeit, ...
7. Ergänzungen (Glossar, ...)