



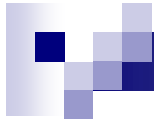
Vorlesung Softwaretechnik I (SS 2024)

3. Vorgehensmodelle

Prof. Dr. Jens Grabowski

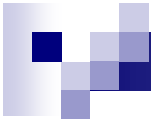
Tel. 39 172022

grabowski@informatik.uni-goettingen.de



Inhalt

- **Worum es in diesem Kapitel geht**
- Build-and-Fix-Cycle
- Lineare Vorgehensmodelle – Wasserfallmodell
- Nichtlineare Vorgehensmodelle
 - Prototyping
 - Evolutionäre Softwareentwicklung
 - Iterative Softwareentwicklung
 - Inkrementelle Softwareentwicklung
 - Treppenmodell
 - Zusammenfassung
- Phasenmodell
- Lernziele



Software-Life-Cycle 1(2)

- Wiederkehrende Schritte bei der Software-Entwicklung:

Analyse

Spezifikation der Anforderungen

Grobentwurf, Spezifikation der Module

Feinentwurf

Codierung und Modultest

Integration, Test, Abnahme

Betrieb, Wartung

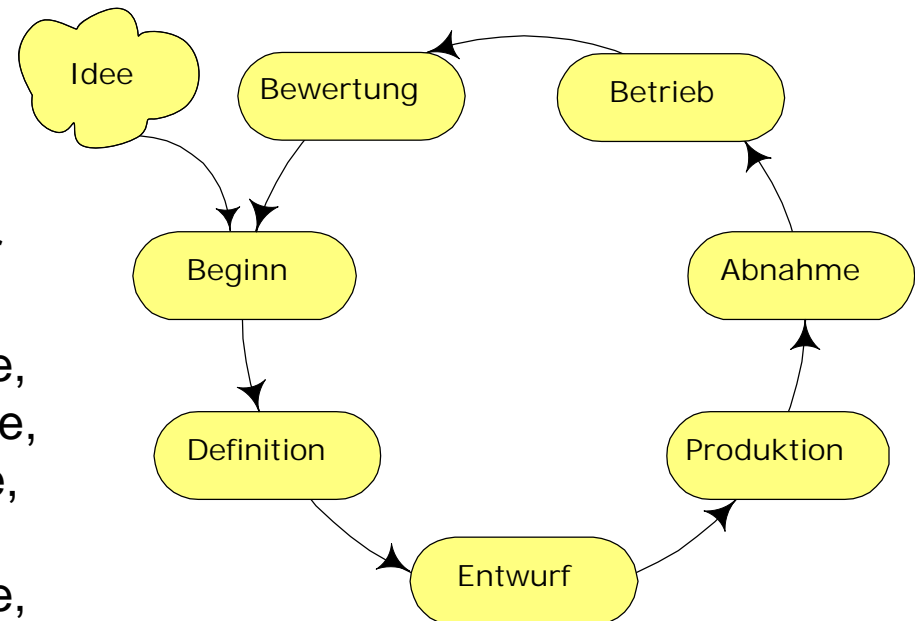
© Ludewig, Lichter, 2006

Software-Life-Cycle 2(2)

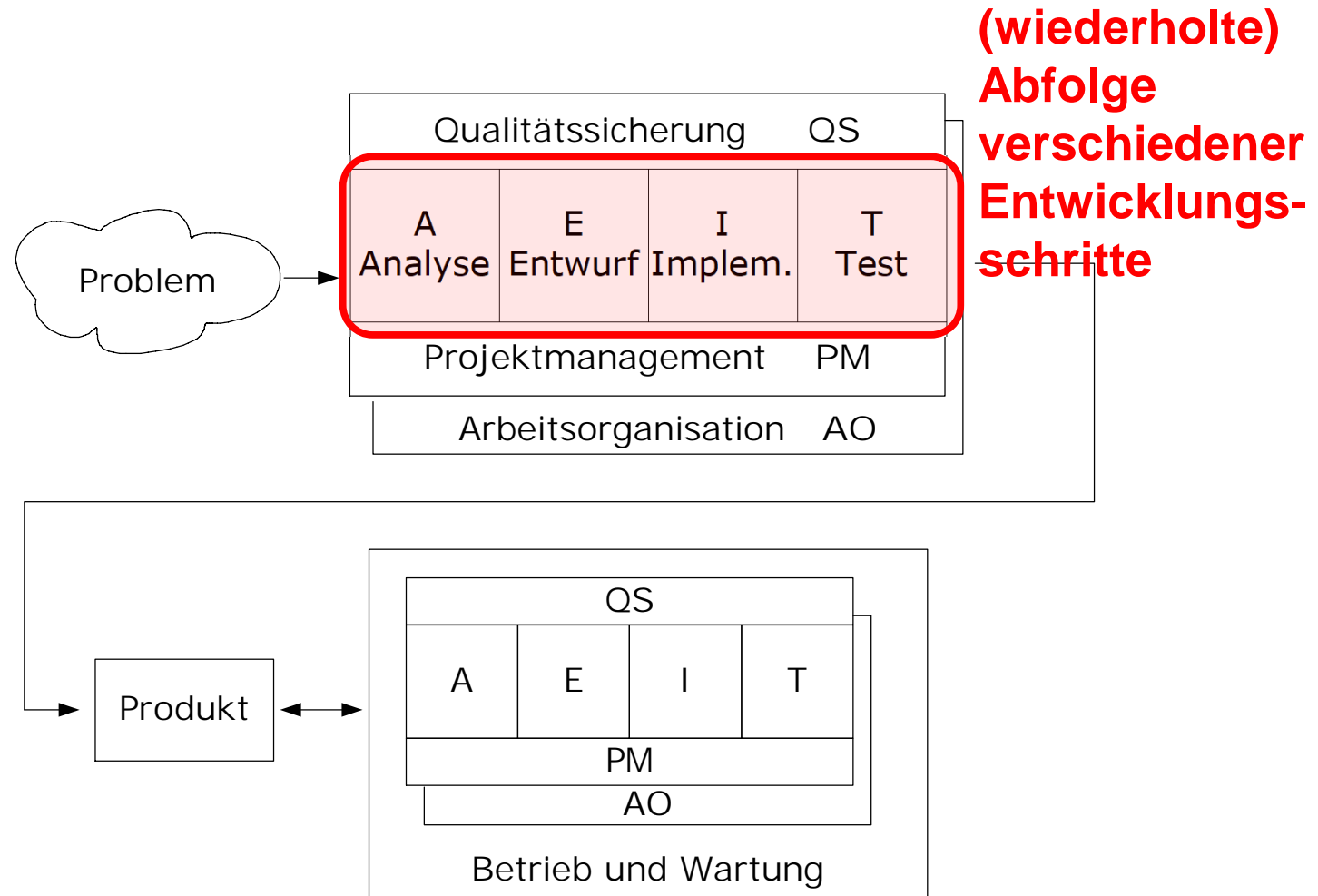
- Def. IEEE Glossar

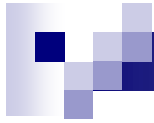
(IEEE Std. 610.12, 1990):

- The period of time that begins when a software product is conceived and ends when the software is no longer available for use. The software life cycle typically includes a concept phase, requirements phase, design phase, implementation phase, test phase, installation and checkout phase, operation and maintenance phase, and, sometimes, retirement phase.
- Note: These phases may overlap or be performed iteratively.



Worum geht es in diesem Kapitel?





Vorgehensmodelle

- **Strategien** für die Durchführung eines Projektes, z.B.
 - Build-and-Fix (Code-and Fix)
 - Software-Life-Cycle
 - Wasserfallmodell
 - Prototyping
 - Nichtlineare Vorgehensmodelle
 - Rapid Prototyping
 - Evolutionäre Software-Entwicklung
 - Iterative Software-Entwicklung
 - Inkrementelle Software-Entwicklung
 - Treppenmodell
 - Phasenmodell



Inhalt

- Worum es in diesem Kapitel geht
- **Build-and-Fix-Cycle**
- Lineare Vorgehensmodelle – Wasserfallmodell
- Nichtlineare Vorgehensmodelle
 - Prototyping
 - Evolutionäre Softwareentwicklung
 - Iterative Softwareentwicklung
 - Inkrementelle Softwareentwicklung
 - Treppenmodell
 - Zusammenfassung
- Phasenmodell
- Lernziele



Das einfachste Vorgehensmodell – Build-and-Fix-Cycle

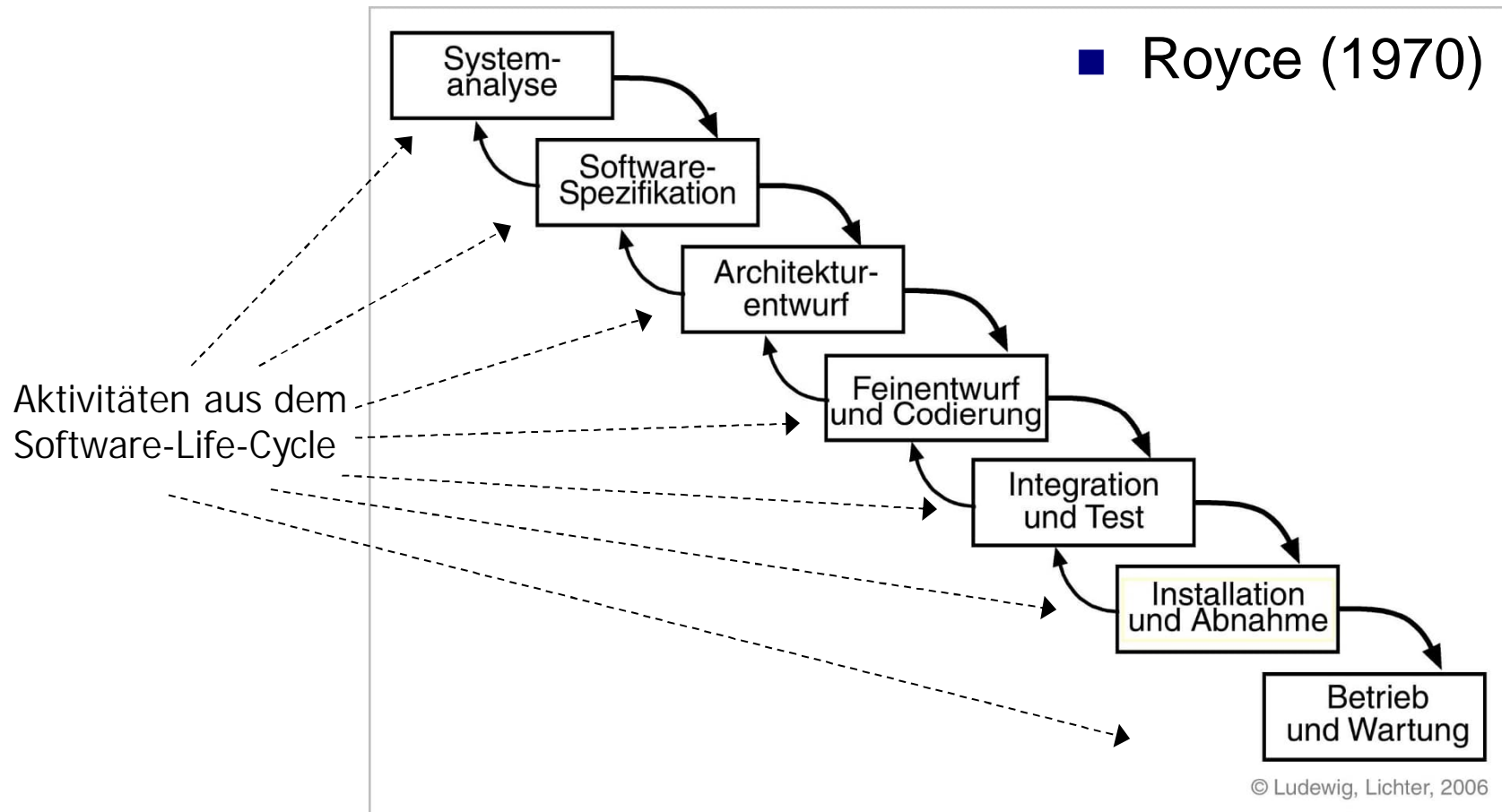
- Typische Vorgehensweise für kleine 1-Personen-Projekte und Übungsaufgaben im (Bachelor-)Studium:
 1. Code schreiben und übersetzen
 2. Code testen (debuggen)
 3. Code „verbessern“
(Fehlerbeseitigung, Erweiterung, Effizienzsteigerung, ...)
 4. GOTO 2
- Probleme mit dieser Vorgehensweise:
 - ☐ Wartbarkeit und Zuverlässigkeit nehmen kontinuierlich ab.
 - ☐ Wenn der Programmierer kündigt ist (oft) alles vorbei.
 - ☐ Wenn Entwickler und Anwender nicht identisch sind, gibt es oft Meinungsverschiedenheiten über erwarteten/realisierten Funktionsumfang.



Inhalt

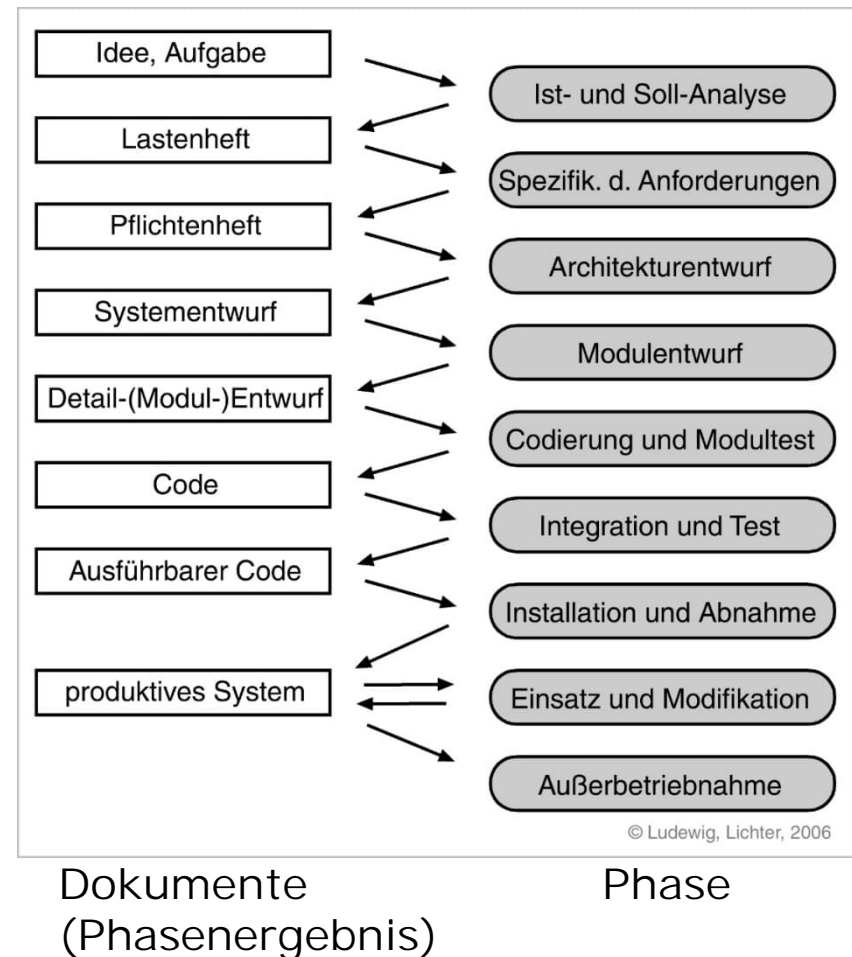
- Worum es in diesem Kapitel geht
- Build-and-Fix-Cycle
- **Lineare Vorgehensmodelle – Wasserfallmodell**
- Nichtlineare Vorgehensmodelle
 - Prototyping
 - Evolutionäre Softwareentwicklung
 - Iterative Softwareentwicklung
 - Inkrementelle Softwareentwicklung
 - Treppenmodell
 - Zusammenfassung
- Phasenmodell
- Lernziele

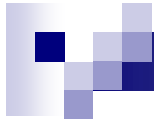
Lineare Vorgehensmodelle – Wasserfallmodell 1(2)



Lineare Vorgehensmodelle – Wasserfallmodell 2(2)

- Strenges Wasserfall- oder Einbahnstraßenmodell
 - Jeder Aktivität im aktivitätsorientierten Wasserfallmodell ist eine spezielle Phase zugeordnet.
- Kritik:
 - Einbahnstraßenmodell raubt die notwendige Flexibilität bei der Entwicklung, z.B. für das frühe Erkennen von kritischen Aspekten oder die explorative Entwicklung von Systemen ohne vollständige Spezifikation.





Inhalt

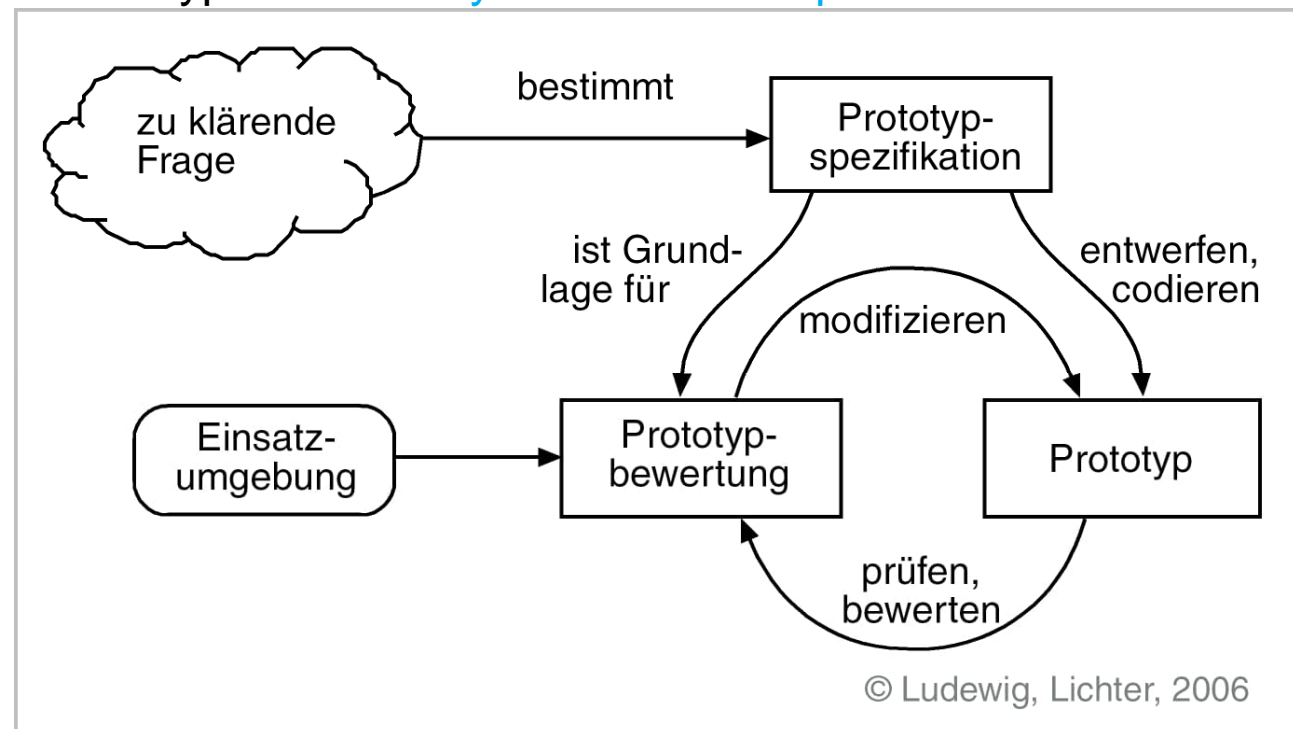
- Worum es in diesem Kapitel geht
- Build-and-Fix-Cycle
- Lineare Vorgehensmodelle – Wasserfallmodell
- **Nichtlineare Vorgehensmodelle**
 - Prototyping
 - Evolutionäre Softwareentwicklung
 - Iterative Softwareentwicklung
 - Inkrementelle Softwareentwicklung
 - Treppenmodell
 - Zusammenfassung
- Phasenmodell
- Lernziele

Nichtlineare Vorgehensmodelle – Prototyping 1(4)

■ (Rapid-) Prototyping:

- Erstellung von „Software-Attrappen“ um Anforderungen zu klären.
- Erstellung von Prototypen zur Analyse kritischer Aspekte.

■ Prototyp-Entwicklung





Nichtlineare Vorgehensmodelle – Prototyping 2(4)

■ Charakteristiken von Prototypen

- Ein Prototyp ist **lauffähig**.
 - Reine Bildschirmmaske ist noch kein Prototyp
- Ein Prototyp realisiert **ausgewählte Aspekte des Zielsystems**.
 - Die ausgewählten Aspekte müssen vor der Realisierung des Prototyps festgelegt werden. Mögliche Aspekte sind: Anforderungen an die Bedienoberfläche, Zusammenarbeit mit anderen Komponenten.
- Ein Prototyp wird von Anwendern **geprüft und detailliert bewertet**.
 - Prototyp wird solange modifiziert, bis die Wünsche des Anwenders im Wesentlichen erfüllt sind.



Nichtlineare Vorgehensmodelle – Prototyping 3(4)

- Spezielle Arten von Prototypen

- Demonstrationsprototyp

- zeigt die prinzipiellen Einsatzmöglichkeiten oder die mögliche Handhabung des zukünftigen Systems.
 - ist ein „Wegwerfprodukt“ (Keine Qualitätsanforderungen).
 - Wird häufig für Start- und Akquisitionsphase eines Projekts benötigt.

- Funktionale Prototyp

- modelliert Ausschnitte der Bedienoberfläche oder der Funktionalität.
 - unterstützt die Anforderungsanalyse.

- Labormuster

- modelliert technischen Aspekt des Zielsystems. Der zu klärende Aspekt kann sich auf Funktionalität oder Architektur des Zielsystems beziehen.

- Pilotsystem

- ist Prototyp, dessen Funktionalität und Qualität (mindestens) für einen vorübergehenden echten Einsatz ausreicht.



Nichtlineare Vorgehensmodelle – Prototyping 4(4)

- Achtung:
 - ☐ Prototyp ersetzt keine Spezifikation, kann aber Teil der Spezifikation werden.
 - ☐ Prototyp sollte auch nicht beliebig schlecht sein.
- Prototypen sollten nicht in das Produkt eingebaut werden. In der Praxis wird diese Regel häufig missachtet.



Inhalt

- Worum es in diesem Kapitel geht
- Build-and-Fix-Cycle
- Lineare Vorgehensmodelle – Wasserfallmodell
- Nichtlineare Vorgehensmodelle
 - Prototyping
 - **Evolutionäre Softwareentwicklung**
 - Iterative Softwareentwicklung
 - Inkrementelle Softwareentwicklung
 - Treppenmodell
 - Zusammenfassung
- Phasenmodell
- Lernziele



Nichtlineare Vorgehensmodelle – Evolutionäre Softwareentwicklung

- Evolutionäre Software-Entwicklung:
 - Vorgehensweise, die eine Evolution der Software unter dem Einfluss ihrer praktischen Erprobung einschließt. Neue und veränderte Anforderungen werden dadurch berücksichtigt, dass die Software in sequentiellen Evolutionsstufen entwickelt wird. [Ludewig, Lichter 2006]
- Evolutionäre Entwicklung ist in der OO-Entwicklung sehr populär und zudem ein Kernpunkt aller agilen Prozesse. In agilen Prozessen wird die Tendenz einer strukturellen Korrosion auf Grund häufiger Änderungen durch Refactoring bekämpft.



Inhalt

- Worum es in diesem Kapitel geht
- Build-and-Fix-Cycle
- Lineare Vorgehensmodelle – Wasserfallmodell
- Nichtlineare Vorgehensmodelle
 - Prototyping
 - Evolutionäre Softwareentwicklung
 - **Iterative Softwareentwicklung**
 - Inkrementelle Softwareentwicklung
 - Treppenmodell
 - Zusammenfassung
- Phasenmodell
- Lernziele



Nichtlineare Vorgehensmodelle – Iterative Softwareentwicklung 1(2)

- Iterative Software-Entwicklung

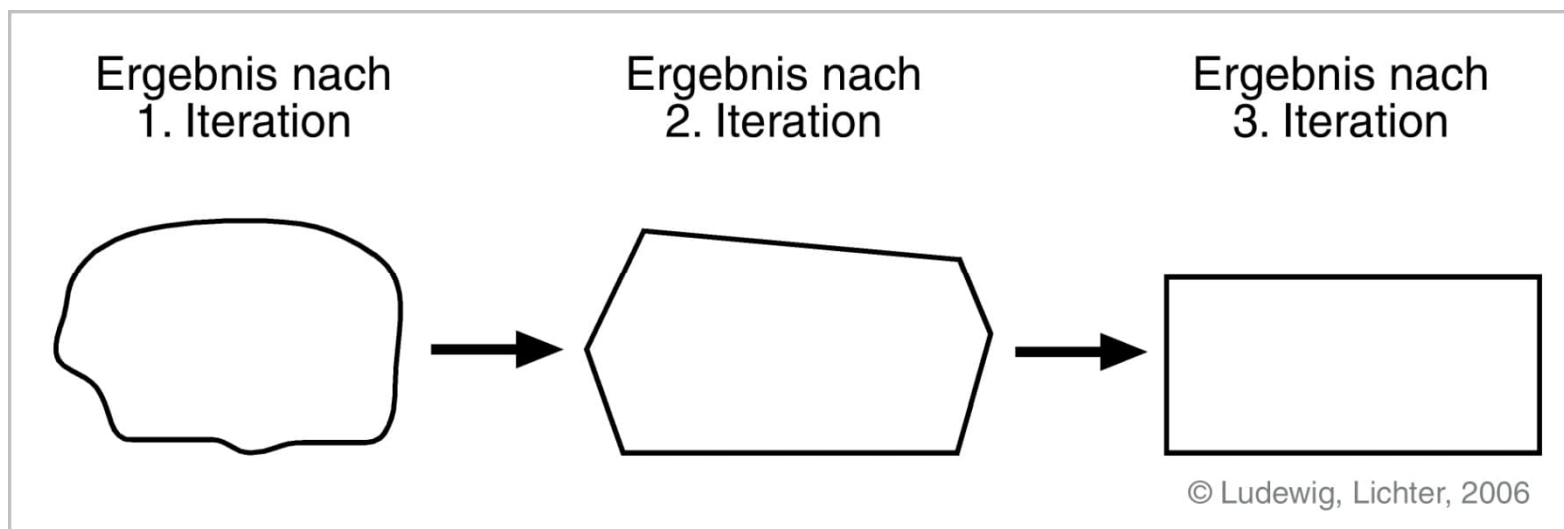
- ☐ Software wird in mehreren geplanten und kontrolliert durchgeführten Iterationsschritten entwickelt. Ziel dabei ist, dass in jedem Iterationsschritt – beginnend bei der zweiten Iteration – das vorhandene System auf der Basis der im Einsatz erkannten Mängel korrigiert und verbessert wird. Bei jedem Iterationsschritt werden die charakteristischen Tätigkeiten Analysieren, Entwerfen, Codieren und Testen durchgeführt. [Ludewig, Lichter 2006]

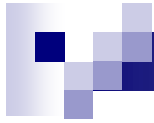
- Vorgehen basiert auf der Beobachtung

- ☐ Auch mit einer aufwendigen Analyse ist es bei komplexen Systemen schwierig ein den tatsächlichen Anforderungen genügendes System zu entwickeln.
- ☐ Existenz und Einsatz eines Systems verändern die Anforderungen an ein System.

Nichtlineare Vorgehensmodelle – Iterative Softwareentwicklung 2(2)

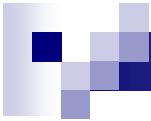
- Cockburn (1993):
 - We get things wrong before we get them right.
 - We make things badly before we make them well.
- Annäherung durch iterative Entwicklung:





Inhalt

- Worum es in diesem Kapitel geht
- Build-and-Fix-Cycle
- Lineare Vorgehensmodelle – Wasserfallmodell
- Nichtlineare Vorgehensmodelle
 - Prototyping
 - Evolutionäre Softwareentwicklung
 - Iterative Softwareentwicklung
 - **Inkrementelle Softwareentwicklung**
 - Treppenmodell
 - Zusammenfassung
- Phasenmodell
- Lernziele

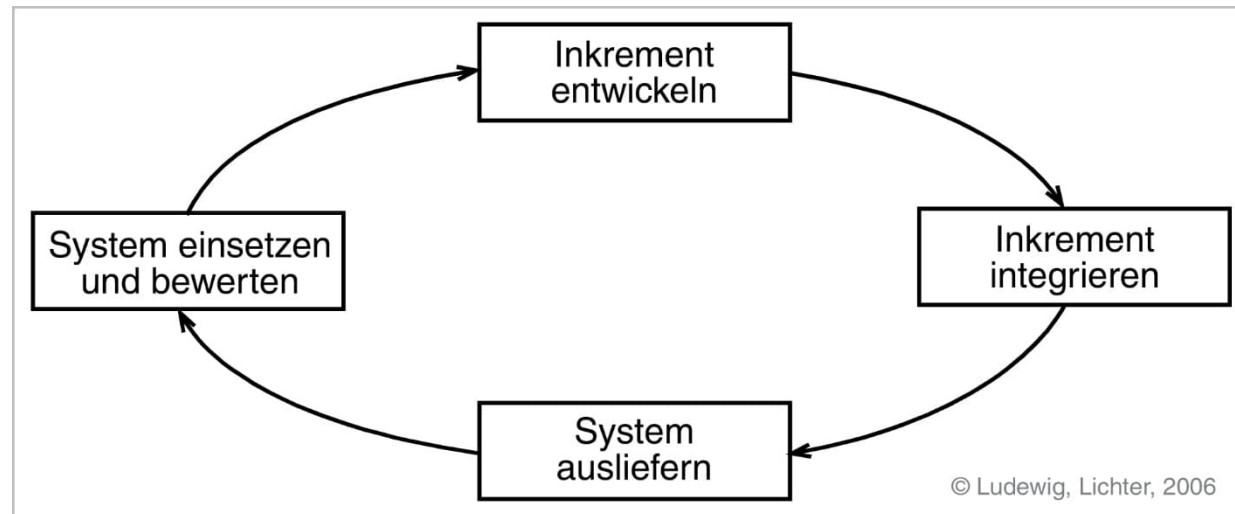


Nichtlineare Vorgehensmodelle – Inkrementelle Softwareentwicklung 1(2)

- Inkrementelle Software-Entwicklung:
 - Das zu entwickelnde System bleibt in seinem Gesamtumfang offen; es wird in Ausbaustufen realisiert. Die erste Stufe ist das Kernsystem. Jede Ausbaustufe erweitert das vorhandene System und wird in einem eigenen Projekt erstellt. Mit der Bereitstellung einer Erweiterung ist in aller Regel auch (wie bei der iterativen Entwicklung) eine Verbesserung der alten Komponenten verbunden. [Ludewig, Lichter 2006]
- Unterschied zur iterativen Software-Entwicklung:
 - Iterative Entwicklung:
 - System wird von Iteration zu Iteration besser!
 - Inkrementelle Entwicklung:
 - Funktionalität des Systems wird von Inkrement zu Inkrement erweitert.
- **Achtung:** Inkrementelle Entwicklung erfordert **Kernsystem!**

Nichtlineare Vorgehensmodelle – Inkrementelle Softwareentwicklung 2(2)

■ Vorgehen:



■ Vorteile inkrementeller Entwicklung:

- Kernsystem wird früh eingesetzt, wodurch technische und fachliche Schwachstellen frühzeitig erkannt und behoben werden können.
- Entwicklungszeiten der einzelnen Inkremente sind in Relation zur Gesamtentwicklungszeit kurz. Fehler in den Inkrementen können schneller kostengünstig korrigiert werden.



Inhalt

- Worum es in diesem Kapitel geht
- Build-and-Fix-Cycle
- Lineare Vorgehensmodelle – Wasserfallmodell
- Nichtlineare Vorgehensmodelle
 - Prototyping
 - Evolutionäre Softwareentwicklung
 - Iterative Softwareentwicklung
 - Inkrementelle Softwareentwicklung
 - **Treppenmodell**
 - Zusammenfassung
- Phasenmodell
- Lernziele



Nichtlineare Vorgehensmodelle – Treppenmodell 1(3)

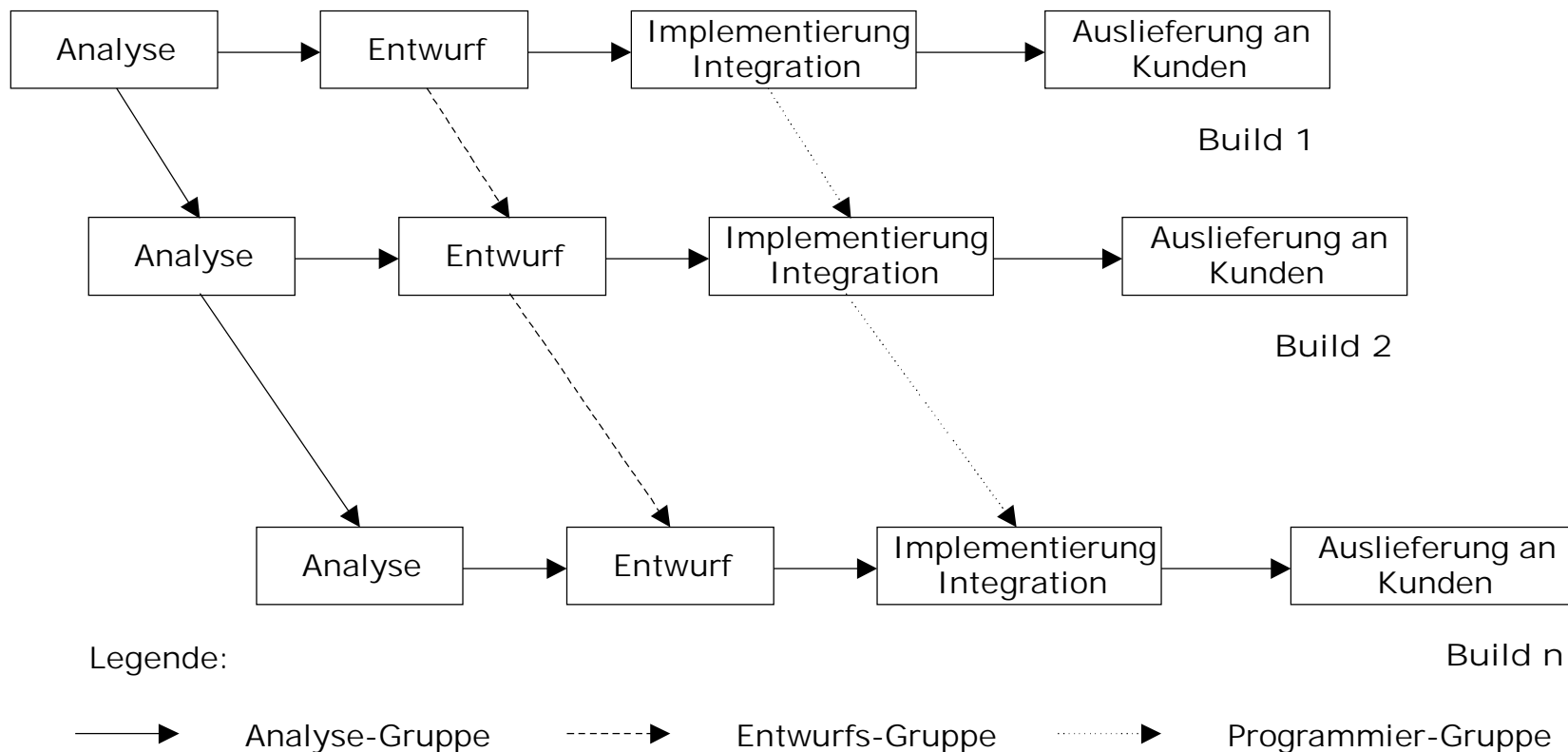
- Treppenmodell

- Das zu entwickelnde System wird in definierten Ausbaustufen realisiert und ausgeliefert. Die erste Stufe ist das Kernsystem. Jede weitere Ausbaustufe erweitert das vorhandene System um Leistungen und Merkmale, die überwiegend bereits zu Beginn des Gesamtprojekts geplant worden sind.
[Ludewig, Lichter 2006]

- Einsatzgebiet:

- Immer dann, wenn die Einführung auf dem Markt oder die Installation beim Kunden drängt.

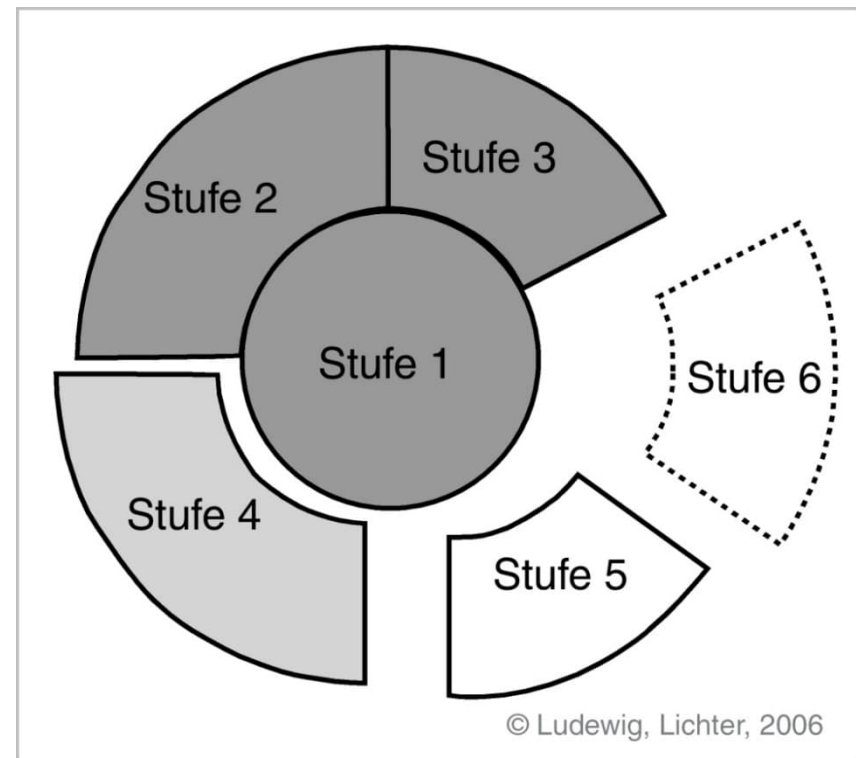
Nichtlineare Vorgehensmodelle – Treppenmodell 2(3)



Nichtlineare Vorgehensmodelle – Treppenmodell 3(3)

- Momentaufnahme einer Entwicklung nach dem Treppenmodell:

- ☐ Fertig: 1, 2, 3
- ☐ Fast fertig: 4
- ☐ Angefangen: 5
- ☐ In Planung: 6

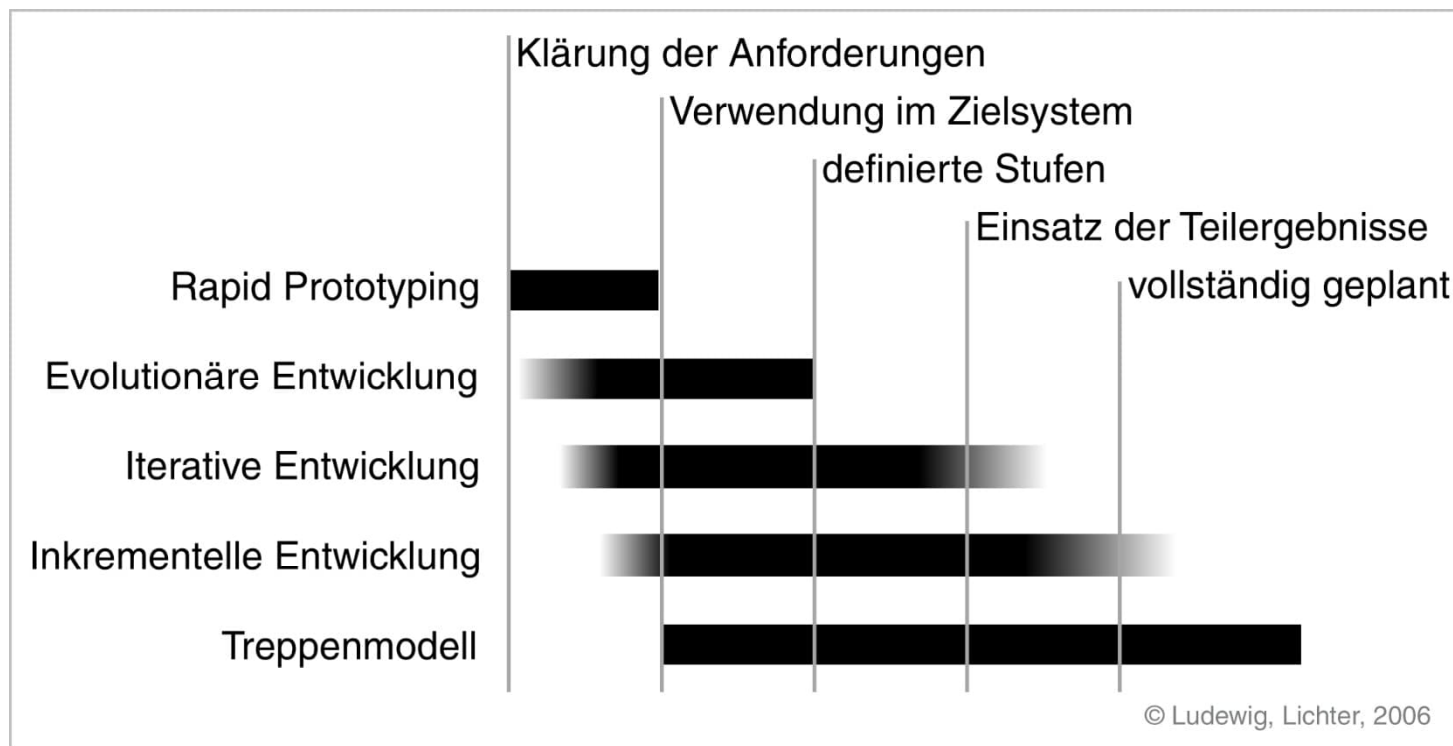




Inhalt

- Worum es in diesem Kapitel geht
- Build-and-Fix-Cycle
- Lineare Vorgehensmodelle – Wasserfallmodell
- Nichtlineare Vorgehensmodelle
 - Prototyping
 - Evolutionäre Softwareentwicklung
 - Iterative Softwareentwicklung
 - Inkrementelle Softwareentwicklung
 - Treppenmodell
 - **Zusammenfassung**
- Phasenmodell
- Lernziele

Nichtlineare Vorgehensmodelle – Zusammenfassung





Inhalt

- Worum es in diesem Kapitel geht
- Build-and-Fix-Cycle
- Lineare Vorgehensmodelle – Wasserfallmodell
- Nichtlineare Vorgehensmodelle
 - Prototyping
 - Evolutionäre Softwareentwicklung
 - Iterative Softwareentwicklung
 - Inkrementelle Softwareentwicklung
 - Treppenmodell
 - Zusammenfassung
- **Phasenmodell**
- Lernziele



Phasenmodell

1(7)

- Phasenmodell beinhaltet die Gliederung des Software-Entwicklungsprozesses in Abschnitte (Phasen), an deren Grenzen eine wesentliche Änderung auftritt.
- **Phasen** sind also Abschnitte,
 - die Anfang und Ende haben,
 - nicht unterbrochen werden können und
 - nicht überlappen.
- Zwischen den Phasen liegen **Meilensteine**, d.h. markante Punkte/Ereignisse mit bekannten Positionen/Ergebnissen.



Phasenmodell

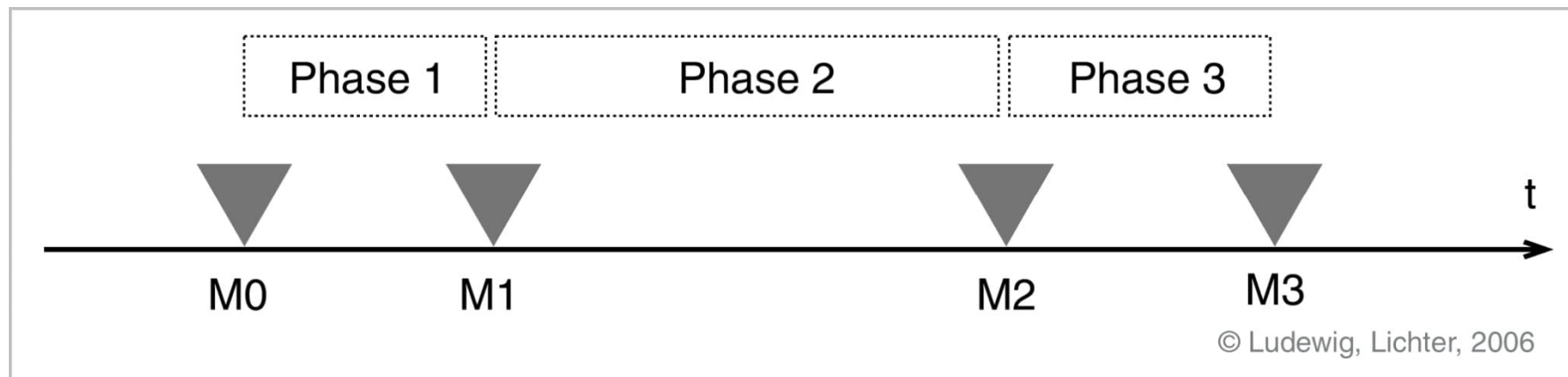
2(7)

- Def. Meilenstein [Wallin et al. 2002]:
 - A milestone is defined as a scheduled event that marks the completion of one or more important tasks, and it is used to measure achievements and development progress. At a milestone, a predefined set of deliverables should have reached a predefined state to enable a review.
- Für jeden Meilenstein wird definiert:
 - welche Ergebnisse (Dokumente) vorliegen müssen,
 - wer nach welchen Kriterien welche Prüfung vornimmt
 - wer schließlich entscheidet, ob der Meilenstein erreicht ist.

Phasenmodell

3(7)

- Charakterisierung Phasenmodell (Ludewig, Lichter 2006):
 - Die Software-Entwicklung wird vor Beginn in Phasen gegliedert, die streng sequentiell durchlaufen werden. Für jede Phase gibt es ein eigenes Budget; dieses Budget wird erst freigegeben, wenn der vorangehende Meilenstein erreicht ist. Dann kann die nächste Phase beginnen.





Phasenmodell

4(7)

- Abgrenzung zum Wasserfallmodell:
 - Im Gegensatz zum Wasserfallmodell enthält das Phasenmodell keine Zyklen:
 - Eine abgeschlossene Phase kann nicht wieder geöffnet werden, sie ist Vergangenheit. Das Erreichen eines Meilensteins dokumentiert einen Fortschritt im Projekt.
 - Beim Wasserfallmodell ist die Verantwortung bei Rückkehr zu früheren Tätigkeiten nicht geregelt, bzw. liegt in der Verantwortung des Entwicklers.



Phasenmodell

5(7)

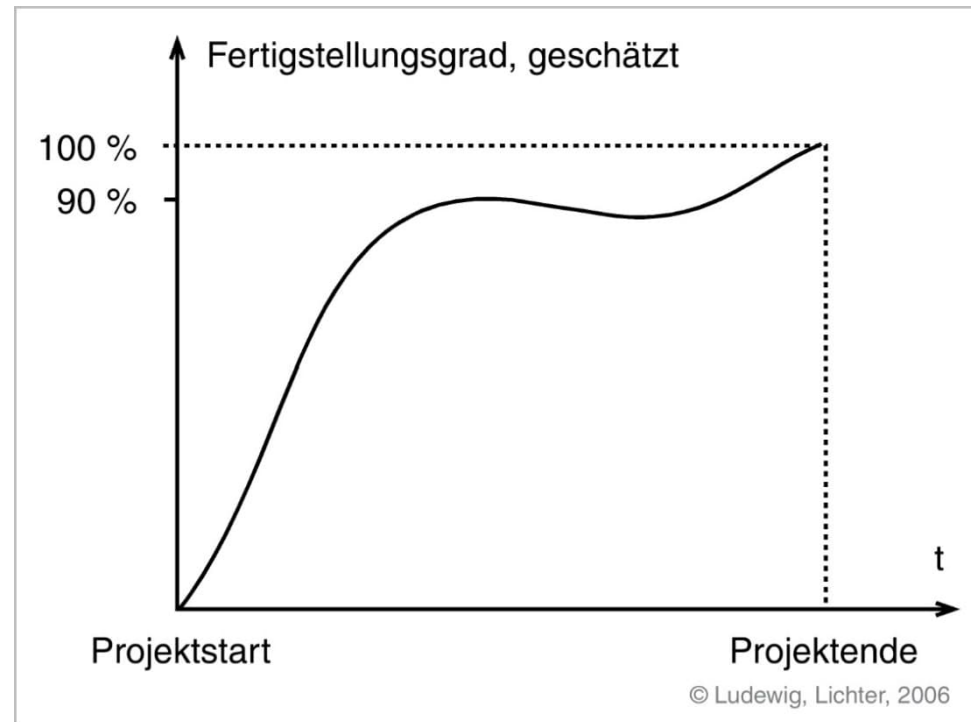
■ Vorteile Phasenmodell

- Projekte sind präzise geplant und organisiert.
 - Abweichungen von der Planung sind leicht und schnell durch das Nicht-Erreichen von Meilensteinen erkennbar.
- Dokumente, deren Erstellung geplant wurde, liegen nach Abschluss der zugehörigen Phase vor und werden geprüft.
 - Entsprechen sie nicht den Anforderungen, wurde der zugehörige Meilenstein nicht erreicht.
- Durchführung von Prüfungen aller Phasenergebnisse ist geplant und gewährleistet.
- Personalbedarf wird für jede Phase frühzeitig geklärt.
- Das 90%-fertig Syndrom wird ausgeschlossen.

Phasenmodell

6(7)

- 90%-fertig Syndrom
 - Entwickler neigen dazu den Entwicklungsstand als 90%-fertig zu charakterisieren.
 - „Die ersten 90% des Projektes brauchen 10% der Zeit. Die restlichen 10% benötigen die anderen 90%.“
- Phasenmodell beschränkt diese Neigung auf einzelne Phasen.





Phasenmodell

7(7)

- Nachteile Phasenmodell

- ☐ Erheblicher Aufwand für Organisation und Prüfungen.
- ☐ Phasenmodell sagt nicht all zuviel aus, sondern gibt nur einen zusätzlichen Rahmen.

- *Vorteile des Phasenmodells sind jedoch so gewichtig, das man eine Planung ohne Phasen und Meilensteine als sehr riskant einstufen muss.*



Inhalt

- Worum es in diesem Kapitel geht
- Build-and-Fix-Cycle
- Lineare Vorgehensmodelle – Wasserfallmodell
- Nichtlineare Vorgehensmodelle
 - Prototyping
 - Evolutionäre Softwareentwicklung
 - Iterative Softwareentwicklung
 - Inkrementelle Softwareentwicklung
 - Treppenmodell
 - Zusammenfassung
- Phasenmodell
- **Lernziele**



Lernziele

- Was sind Vorgehensmodelle?
- Welche Vorgehensmodelle gibt es?
- Was beschreiben die verschiedenen Vorgehensmodelle und wie unterscheiden sie sich?