SOFTWARE ENGINEERING

Elite Graduate Program

Projektmanagement: Prozessmodelle

Martin Wirsing

Institut für Informatik
Ludwig-Maximilians-Universität München

WS 2006/07

Ludwig Maximilians Universität München

Ziele

- **Wichtige Prozessparadigmen und Vorgehensmodelle wiederholen und in Zusammenhang mit Projekten stellen:**
  - Sequentiell („Wasserfall“),
  - iterativ („spiralförmig“),
  - leichtgewichtig („agil“).

Tätigkeiten bei der Software-Entwicklung

- Anforderungsanalyse & Fachliche Konzeption
- Technische Konzeption (Design, Entwurf)
- Realisierung
- Integration & Test

- Und deren Einordnung in **Vorgehensmodelle**

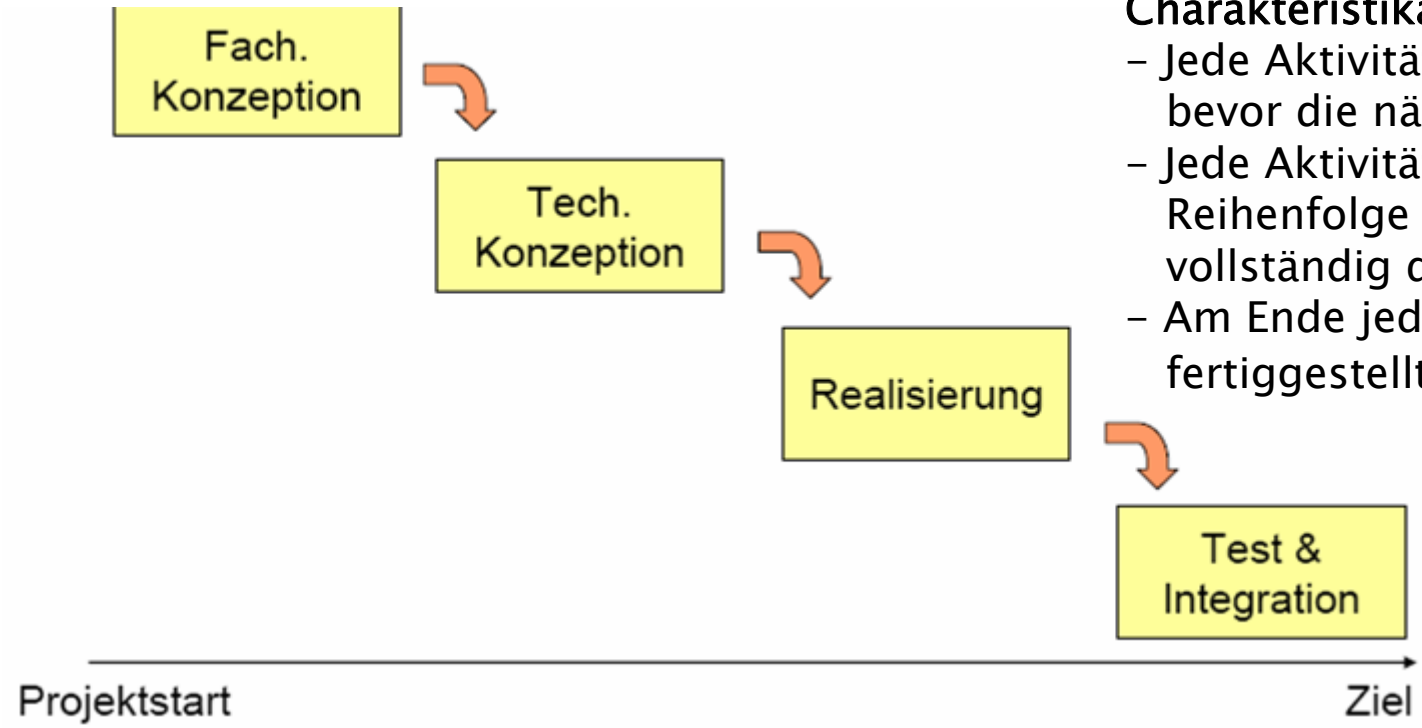
Das sequentielle Paradigma

- **Das sequentielle Paradigma bezeichnet**
  ein sequentielles Vorgehen mit
  klar definierten Phasen und Ergebnissen

- **Bekannteste Vorgehensmodelle:**
  - **Wasserfall-Modell**
  - **V-Modell**

Wasserfallmodell (schematisch)

- Eingeführt von Walker Royce 1970, Weiterentwicklung der „stufenweisen Entwicklung“ („stagewise development“) von Benington 1956.
- Ansatz:
  - Top-down Stufenmodell mit eingeschränkter Rückkopplung
  - Phasensynchronisation durch (Zwischen-) Produkte
  - Geringer Managementaufwand



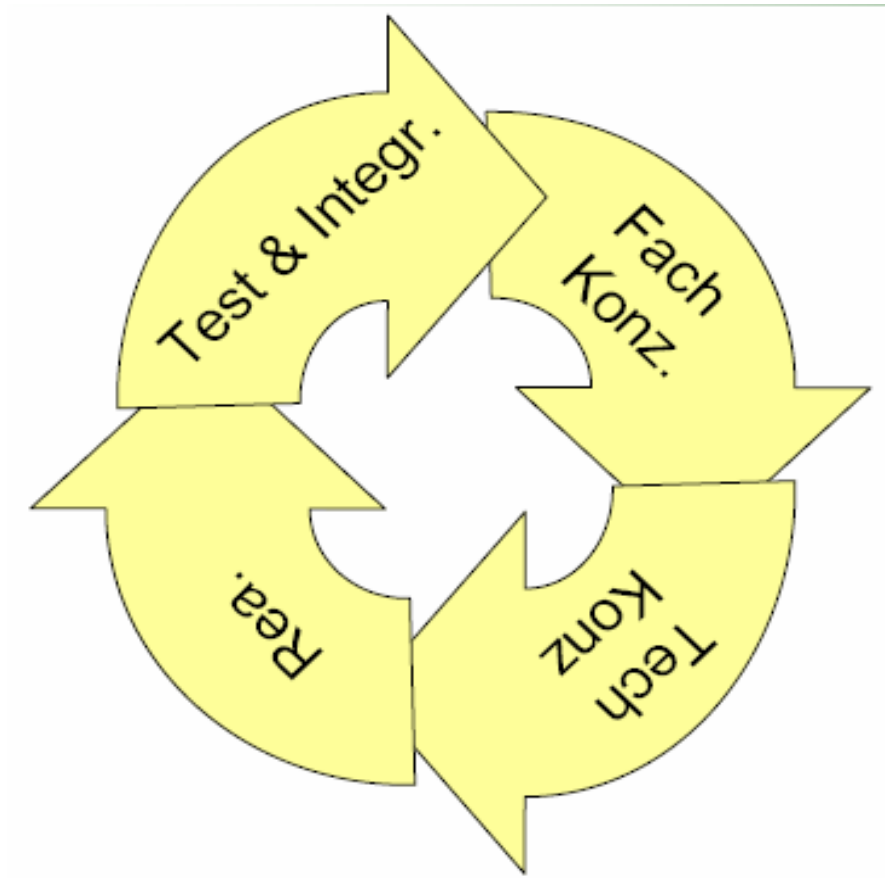
Charakteristika:

- Jede Aktivität muss beendet sein, bevor die nächste beginnt.
- Jede Aktivität ist in der richtigen Reihenfolge und in voller Breite vollständig durchzuführen.
- Am Ende jeder Aktivität steht ein fertiggestelltes Dokument.

Das sequentielle Paradigma: Vor- und Nachteile

- **Gut anwendbar bei klarer, relativ fixer Funktionalität:**
  - System- und Basissoftware wie OS, DB, Web-Server;
  - Branchen-Software wie SAP R/3.
- **Notwendig bei hohen Qualitäts- und Zuverlässigkeitsanforderungen:**
  - eingebettete Systeme (Automotive, Maschinenbau, Medizinische Geräte, Anlagensteuerung, Aerospace, Defense, ...);
  - Telekommunikations-Systeme;
  - rechtliche Anforderungen wie Revisionssicherheit oder Nachvollziehbarkeit.
- **Vorteile**
  - einfach durchzuführen
  - schränkt Freiheitsgrade stark ein, daher auch für sehr große Projekte anwendbar
  - sehr effizient bei bekannten und konstanten Anforderungen
  - ist gut zu vermessen (notwendig z.B. für Prozessverbesserung)
- **Nachteile**
  - Risiken gesammelt am Schluss („Big Bang“)
  - starr während des Ablaufs

Das iterative Paradigma



Iterativ heißt, dass der Entwicklungsprozess mehrfach iteriert wird: statt den „Wasserfall“ einmal zu durchlaufen, werden kleine Wasserfälle hintereinander gesetzt.

Das iterative Paradigma

- **Das iterative Paradigma ist eine Weiterentwicklung des sequentiellen Paradigmas aus der Erkenntnis, dass Software länger lebt als erwartet und auch vom Funktionsumfang her gepflegt werden muss.**

- **Das iterative Paradigma unterstützt inkrementelle Entwicklung, d.h. dass das zu erstellende System nicht in einem Rutsch freigegeben wird, sondern in mehreren Stufen.**

- **Bekannteste Modelle:**
  - **Spiral-Modell (ein Meta-Vorgehensmodell)**
  - **Unified Process (RUP)**

- **Das agile Paradigma ist eine Form des iterativen Paradigmas in Verbindung mit frühem Prototyping, bei der nicht alle Tätigkeiten iteriert werden.**

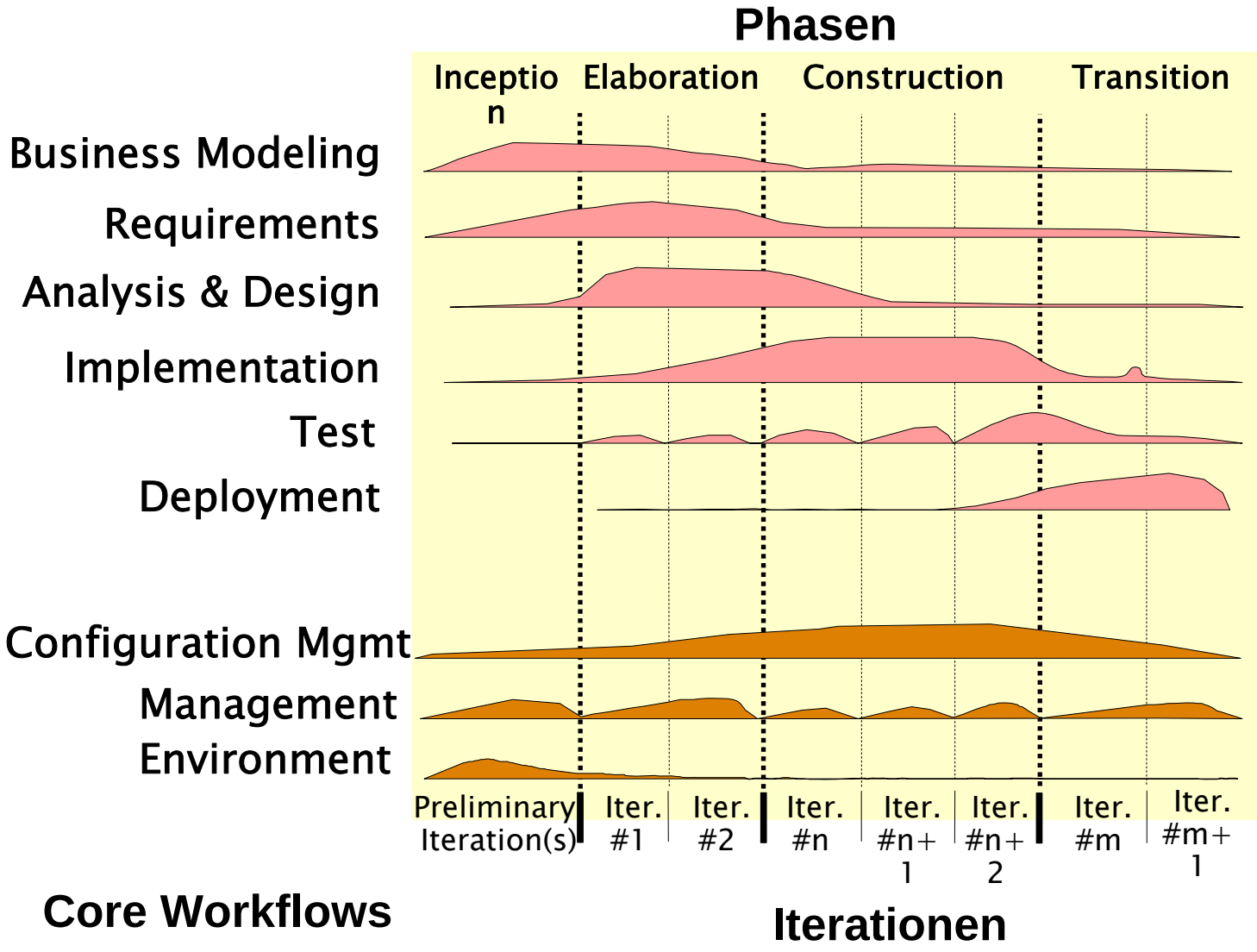
Rational Unified Process (RUP)

RUP ist eine Weiterentwick-lung von Jacobsons Objectory

RUP hat UML als Sprache voreingestellt

RUP wird von Rational/IBM bezeichnet als

- inkrementell & iterativ,
- architektur-zentriert, und
- anwendungsfall-getrieben.



Das iterative Paradigma: Vor- und Nachteile

- **Vorteile**
  - Risiken können früher erkannt werden
  - volatile Anforderungen können besser berücksichtigt werden
  - inkrementelle Auslieferung wird erleichtert

- **Nachteile**
  - Mehrarbeit
  - komplexeres Projektmanagement
  - schwerer messbar

Das adaptive Paradigma

- **Unter einem adaptiven SW-Prozess versteht man eine Weiterentwicklung des iterativen Paradigmas, bei der**
  - die Planung der Iterationen dynamisch erfolgt und
  - von Anfang an Prototypen erstellt werden

- **Charakteristikum:**
  - **kontinuierliche Anpassung an Änderungen**

- **Prinzipien:**
  - **Individuum und Interaktion (geht) vor Prozess und Werkzeug**
  - **ausführbare SW vor vollständiger Dokumentation**
  - **Zusammenarbeit mit Kunden vor Vertragsverhandlung**
  - **Berücksichtigung von Änderungen vor Beharren auf Plan**

- **Bekannte Vorgehensweisen:**
  - **XP**
  - **SCRUM, ...**

XP (Extreme Programming)

- Merkmale
  - **XP arbeitet mit kleinen Releases, unterteilt in Iterationen und Arbeitspakete.**
  - **Anforderungsanalyse**
    - **Aufgaben und Anforderungen in Form von Stories**
    - **Beschränkung auf wenige Anforderungen pro Iteration**
  - **Pair Programming**
    - **Zwei Personen vor einem Rechner, einer programmiert, der andere ist Sparringspartner**
  - **Enthält Elemente des Prototyping, allerdings ohne Wegwerfen**
  - **Auftraggeber wird mit besser eingebunden, kann konkrete Ergebnisse sehen**

- Prozess:
  - **Test-first: Zuerst Tests (Spezifikation) schreiben, dann programmieren.**
  - **Kontinuierliches Refactoring**

Das adaptive Paradigma: Vor- und Nachteile

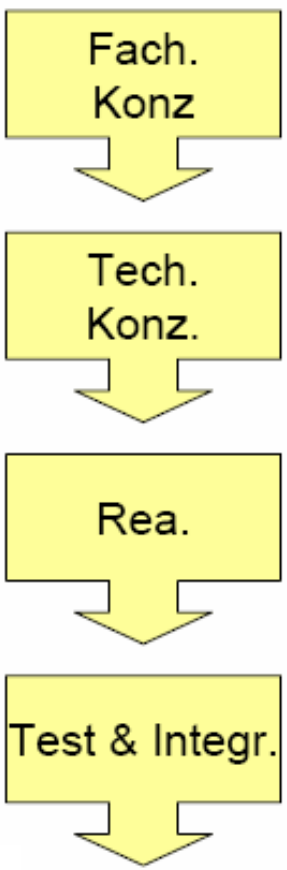
Vorteile

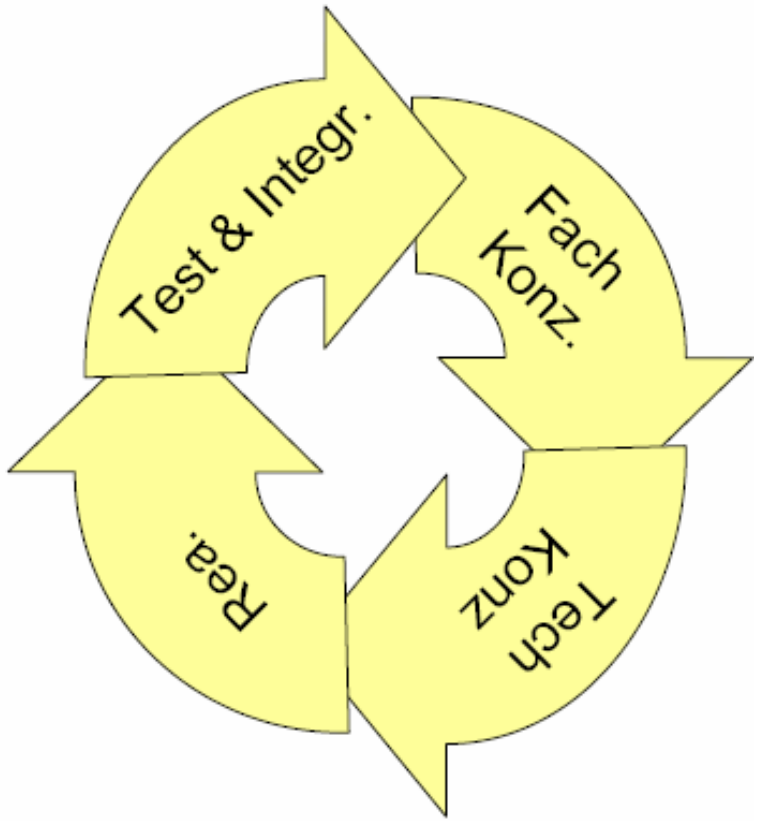
- **Gut einsetzbar bei unklaren Zielen und sich ändernden Anforderungen/Umgebung**
- **Verspricht besseres Kosten/Nutzen-Verhältnis**
- **Vermutlich durchschnittliche Code-Qualität besser**

Nachteile

- **Ergebnis ist nicht vorhersagbar**
  - Qualitätseigenschaften können nicht garantiert werden
  - Oft nicht nachvollziehbar, wie eine Funktion zustande kommt
- **80-90% aller Software läuft auf eingebetteten Systemen.**
  - Das bezieht sich z.B. auf Maschinen und Anlagen jeder Art und Größe, Verkehrsmittel, Militärische Anwendungen, Medizinische Geräte.
  - Hier sind Fehler fatal - ein „agiler“ Prozess kommt nicht (oft) in Frage.
- **Refactoring geht nicht bei Nicht-oo-Sprachen.**
  - Aber ein Großteil (ca. 90%) der bestehenden Applikationen weltweit ist in Cobol, PL1, Assembler und C, und OO-Sprachen sind nicht immer optimal.

Wasserfallmodell versus Inkrementelle Software-Entwicklung





- **Was ist besser?**

Wasserfallmodell versus Inkrementelle Software-Entwicklung

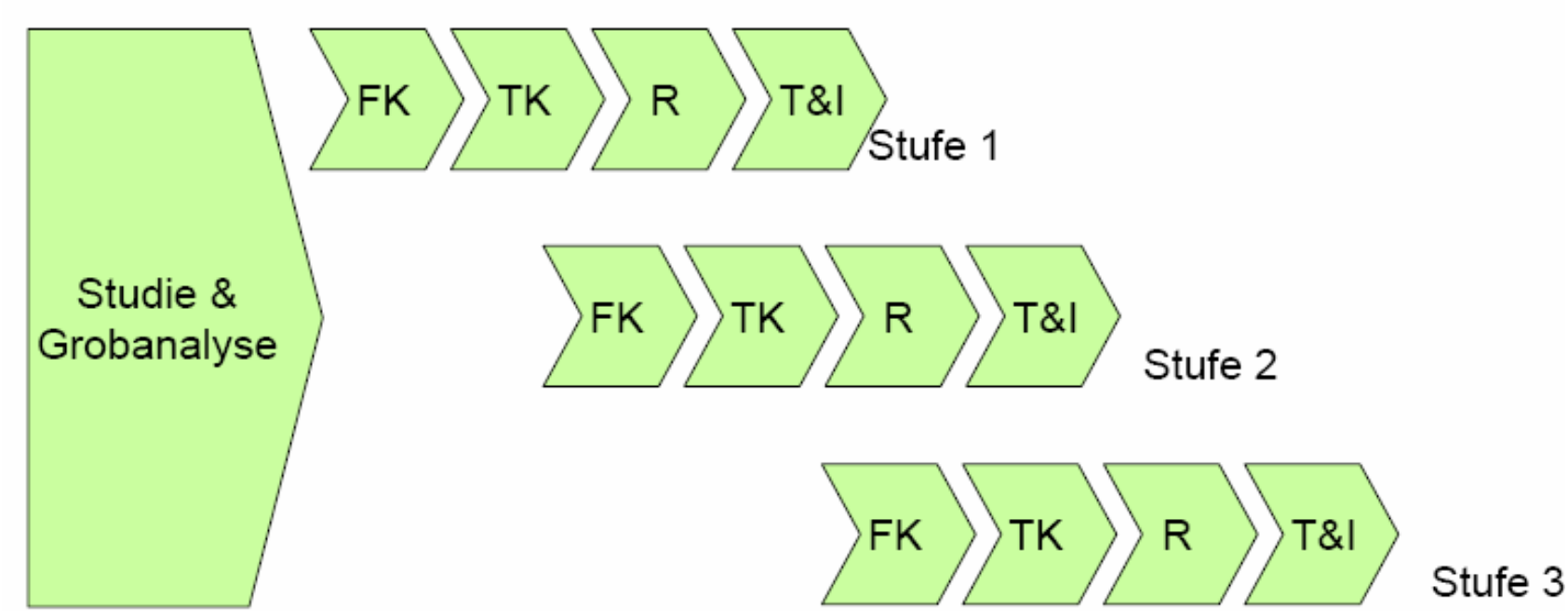
Wasserfall

- Hohe Sicherheit für Software-Anbieter
- Gesamtblick (aber: zu viele Details)
- Unüberschaubare Konzeptpapiere
- Geringere Flexibilität, aber Change-Request-Verfahren
- Nutzen erst bei Einführung
- „Deckel drauf bekommen“
- Einfache Struktur, Qualitätssicherung zwischen Phasen
- Entspricht Denkweise: Geld für definierte Leistung

Iteratives Vorgehen

- Früher Nutzen für Kunden
- Besseres, qualifizierteres Feedback
- Kosten für Übergangslösungen
- Schwieriger zu managen
- Geringeres Einführungsrisiko

Verbesserung: Gestuftes Vorgehen

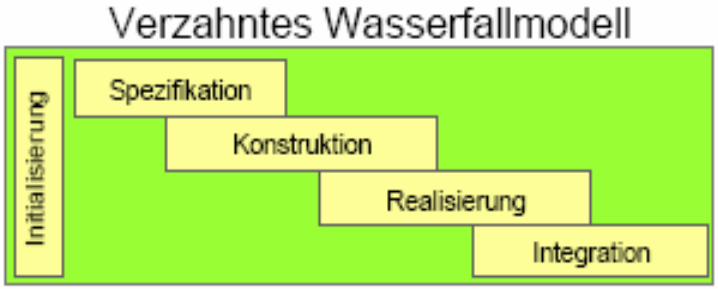


- **Wasserfall als Vorgehen stößt an Grenzen**
  - Große Projekte werden handhabbar
- **Verringern des Risikos**
  - Fachliches Risiko
  - Technisches Risiko
- **Die elegante Art, „nein“ zu sagen: Stufen bedeuten, dass Anforderungen nicht oder erst später erfüllt werden**
  - Widerstände müssen überwunden werden: Management-Aufgabe

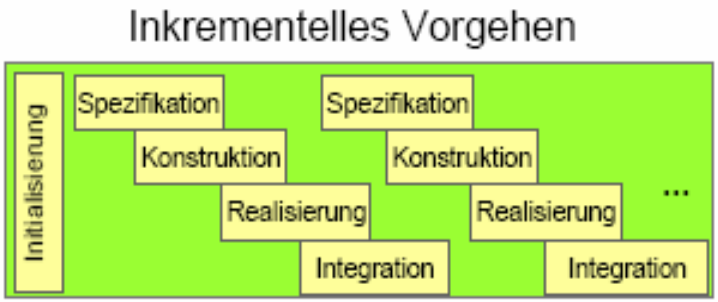
Gestuftes Vorgehen

- Bei Aufsetzen des Projekts
  - **Immer den Umfang des Projekts im Auge behalten**
  - **Bei größerem Umfang Stufung versuchen**
    - **Umfang niemals unterschätzen!**
- Nach Fachkonzeption oder Studie
  - **Zusätzlicher Schnittpunkt für Stufenbildung (fachlich)**
- Indizien für zu großen Projektumfang
  - **Umfang der Konzeptpapiere: größer als 200 Seiten?**
  - **Feedback der Reviewer/externen Beteiligten: wirken sie noch mit, lesen sie die Papiere?**

Beispiele für gestuftes Vorgehen im Projekt



Motivation / Einsatz	Besonderheiten
■ Kleines Projekt ■ Klarer, überschaubarer Funktionsumfang ■ Frühe Gesamtspezifikation erforderlich	■ Ist Spezialfall einer Stufe mit nur einem Inkrement ■ Funktionsumfang früh definiert und weitgehend fix



■ Schnelle Ergebnisse & schnelles Lernen – auch bei komplexer Funktionalität ■ Risiko reduzieren durch „Wichtigstes zuerst“	■ Frühes Feedback durch schnell lauffähiges Teilsystem ■ Schrittweises Verfeinern

■ Gesamtspezifikation zu Beginn der Stufe ■ Gesamtaufwand leichter planbar als bei inkrementellem Vorgehen	■ Mischform der beiden obigen Modelle ■ Schwerpunkt ab zweitem Inkrement auf Realisierung (weniger Konstruktion)

Zusammenfassung (1)

- **Das sequentielle Prozessparadigma (Wasserfall, klassisches V-Modell) ist**
  - einfach durchzuführen
  - auch für sehr große Projekte anwendbar
  - sehr effizient bei bekannten und konstanten Anforderungen

  ABER
  - Risiken gesammelt am Schluss („Big Bang“) und
  - starr während des Ablaufs
- **Das iterative Paradigma (RUP, Spiralmodell) erleichtert**
  - frühe Erkennung von Risiken
  - Berücksichtigung sich ändernder Anforderungen
  - inkrementelle Auslieferung

  ABER
  - es erfordert Mehrarbeit, komplexeres Projektmanagement und ist
  - schwerer messbar

Zusammenfassung (2)

- **Das agile Prozessparadigma (XP) ist**
  - Gut einsetzbar bei unklaren Zielen und sich ändernden Anforderungen/Umgebung
  - Verspricht besseres Kosten/Nutzen-Verhältnis
  - Vermutlich durchschnittliche Code-Qualität besser

  ABER
  - das Ergebnis ist nicht vorhersagbar
  - Qualitätseigenschaften können nicht garantiert werden
- **Ein gestuftes Vorgehen verbindet die Vorteile von sequentiellem und iterativem Vorgehen und vermeidet Nachteile wie „Big-Bang“ und hohem Mehraufwand.**