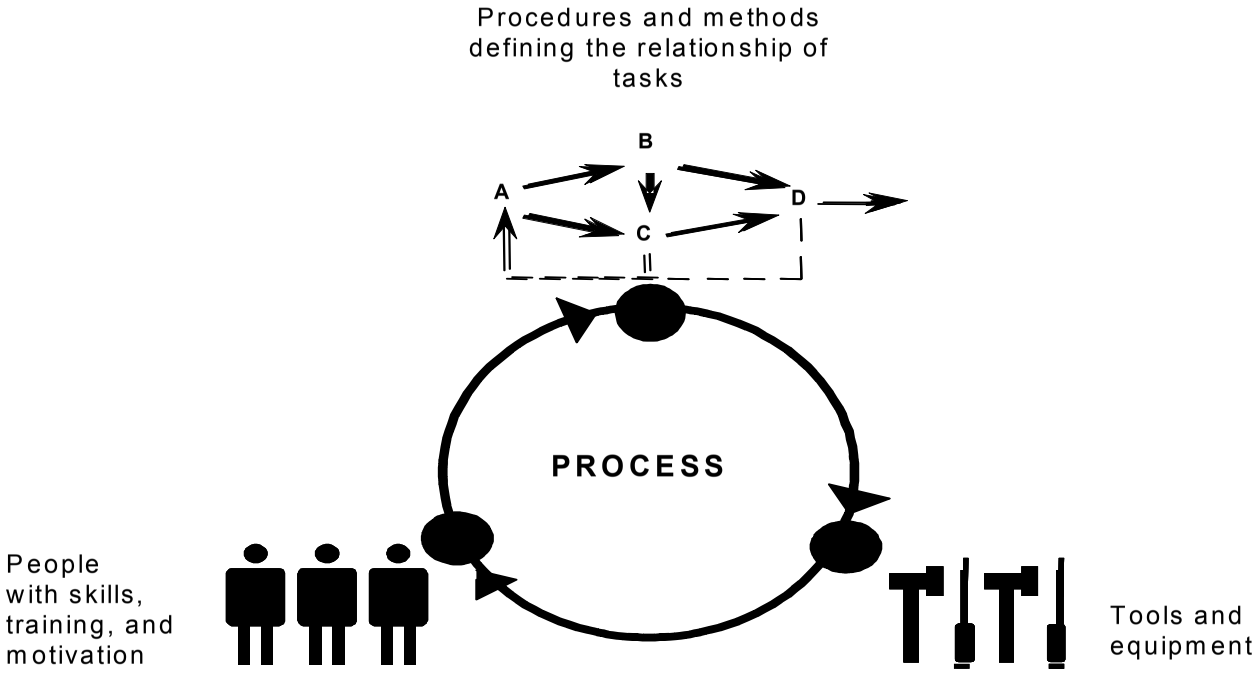
Grundlagen Software Engineering

Prozesse

Organisation: Prozessmodelle
Inhalt

- Das Wasserfall-Modell
- Das V-Modell
- Das Prototypen-Modell
- Das evolutionäre/inkrementelle Modell
- Das nebenläufige Modell
- Das Spiralmodell

Kritische Faktoren in der Softwareentwicklung



- Organisationen konzentrieren sich primär auf
  - Personen
  - Prozeduren und Methoden
  - Werkzeuge
- Es ist aber der Prozess der alles zusammenhält

Organisation
Prozessmodelle

- Ein Prozessmodell legt fest
  - Reihenfolge des Arbeitsablaufs
    - Entwicklungsstufen
    - Phasenkonzepte
  - Jeweils durchzuführende Aktivitäten
  - Definition der Teilprodukte einschließlich Layout und Inhalt
  - Fertigstellungskriterien
  - Notwendige Mitarbeiterqualifikationen
  - Verantwortlichkeiten und Kompetenzen
  - Anzuwendende Standards, Richtlinien, Methoden und Werkzeuge

Organisation
Prozessmodelle

- Modell der Software-Technik: code & fix
  1. Schreibe ein Programm
  2. Finde und behebe die Fehler in dem Programm
  - Nachteile
    - Nach Behebung von Fehlern wurde das Programm so umstrukturiert, dass weitere Fehlerbehebungen immer teurer wurden
      - Dies führt zu der Erkenntnis, dass eine Entwurfsphase vor der Programmierung benötigt wird
    - Selbst gut entworfene Software wurde vom Endbenutzer oft nicht akzeptiert
      - Dies führte zu der Erkenntnis, dass eine Definitionsphase vor dem Entwurf benötigt wird
    - Fehler waren schwierig zu finden, da Tests schlecht vorbereitet und Änderungen unzureichend durchgeführt wurden
      - Dies führte zu einer separaten Testphase

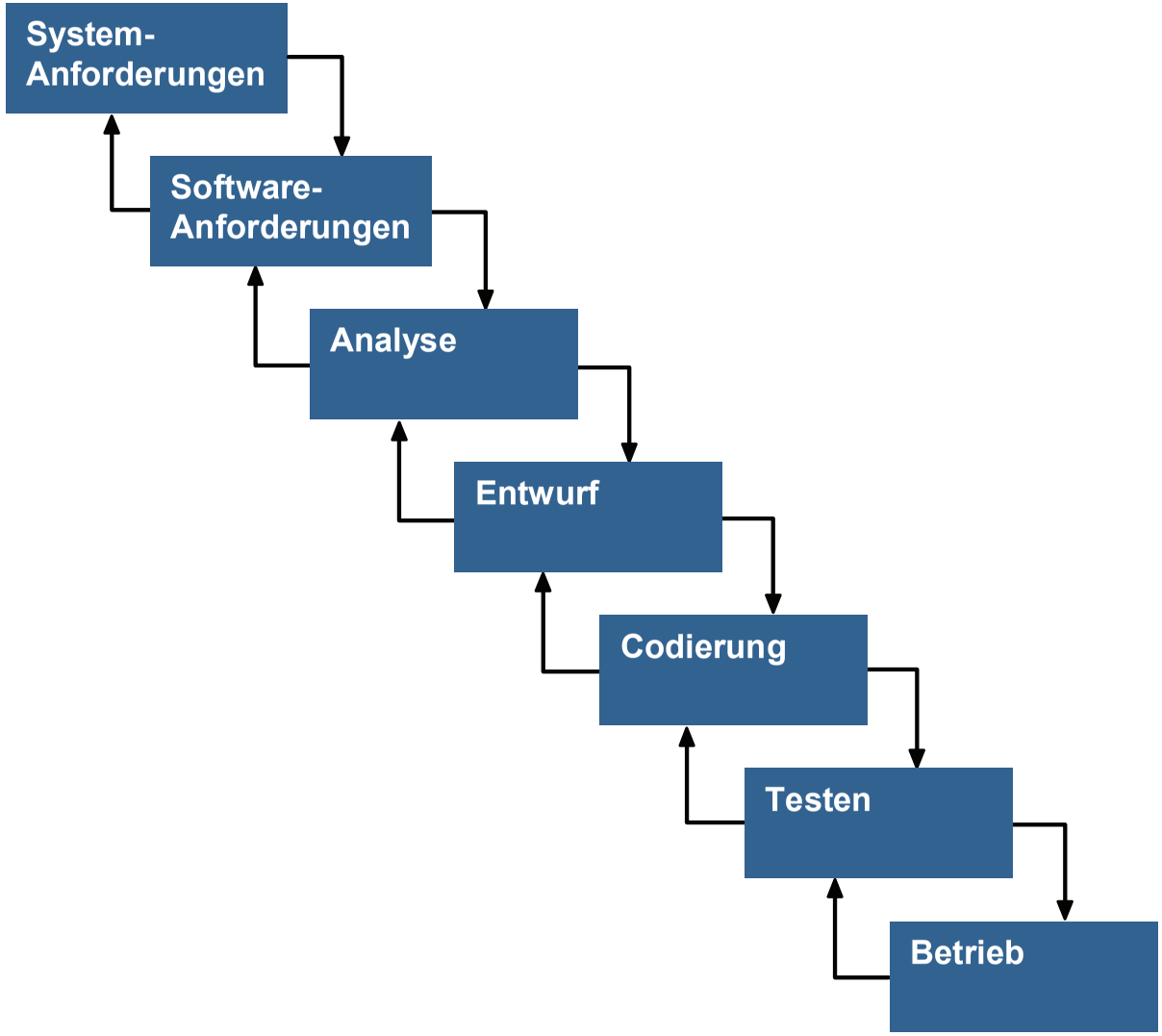
Organisation
Prozessmodelle

- Neue Modelle
  - Das Wasserfall-Modell
  - Das V-Modell
  - Das Prototypen-Modell
  - Das evolutionäre/inkrementelle Modell
  - Das objektorientierte Modell
  - Das nebenläufige Modell
  - Das Spiralmodell

Das Wasserfall-Modell

- Weiterentwicklung des »*stagewise models*«
- Software wird in sukzessiven Stufen entwickelt.
- Rückkopplungsschleifen zwischen den Stufen
  - Rückkopplungen werden auf angrenzende Stufen begrenzt
- Name Wasserfall-Modell
  - Ergebnisse einer Phase fallen wie bei einem Wasserfall in die nächste Phase

Das Wasserfall-Modell



Das Wasserfall-Modell
Charakteristika

- Jede Aktivität ist in der richtigen Reihenfolge und in der vollen Breite vollständig durchzuführen
- Am Ende jeder Aktivität steht ein fertiggestelltes Dokument
  - Dokumenten-getriebenes Modell
- Der Entwicklungsablauf ist sequentiell
  - Jede Aktivität muss beendet sein, bevor die nächste anfängt.
- Orientierung am top-down-Vorgehen
- Einfach, verständlich und benötigt nur wenig Managementaufwand
- Benutzerbeteiligung ist nur in der Definitionsphase vorgesehen

Das Wasserfall-Modell

- Beispiel: »Seminarorganisation«
  1. Unter Einbeziehung des Auftraggebers/Benutzers wird eine Produktdefinition für alle Anforderungen des Auftraggebers ermittelt
     - Nach Abnahme des Produktmodells durch den Auftraggeber ist die Definitionsphase abgeschlossen
  2. In der Entwurfsphase wird ausgehend vom Produktmodell eine Produktarchitektur für das gesamte Produkt entwickelt
     - Sind Anforderungen des Produktmodells nicht realisierbar, wird ein Änderungsdokument erstellt
     - Anschließend beginnt wieder die Entwurfsphase
     - Als Ergebnis der Entwurfsphase wird die Produktarchitektur an die Implementierungsphase übergeben
  3. Die Produktarchitektur wird implementiert
     - Es entsteht das fertige Produkt

Das Wasserfall-Modell

- Nachteile
  - Es ist nicht immer sinnvoll
    - alle Entwicklungsschritte in der vollen Breite und vollständig durchzuführen
    - alle Entwicklungsschritte sequentiell durchzuführen
  - Gefahr, dass die Dokumentation wichtiger wird als das eigentliche System
  - Risikofaktoren werden u. U. zu wenig berücksichtigt, da der einmal festgelegte Entwicklungsablauf durchgeführt wird

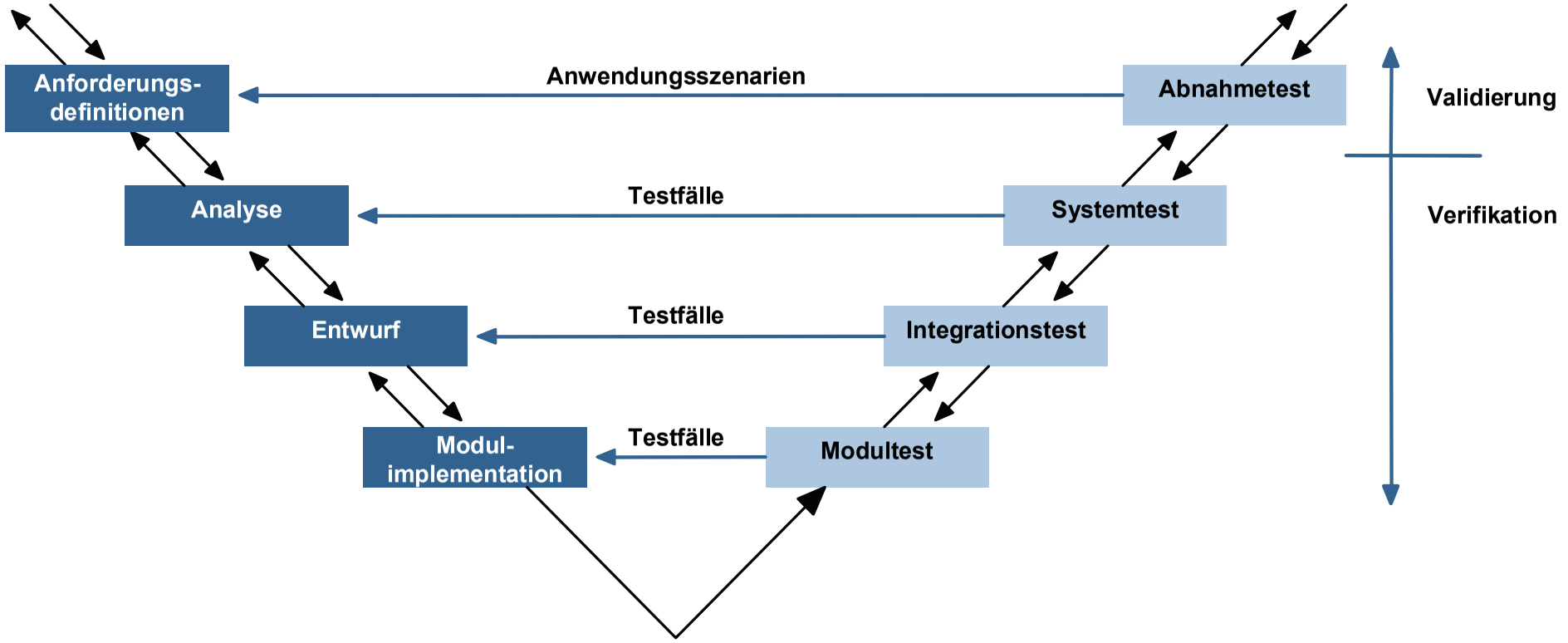
Das V-Modell

- Erweiterung des Wasserfall-Modells
- Integriert die Qualitätssicherung in das Wasserfall-Modell
- Verifikation und Validation der Teilprodukte sind Bestandteile des V-Modells
  - Verifikation
    - Überprüfung der Übereinstimmung zwischen einem Software-Produkt und seiner Spezifikation
    - »Wird ein korrektes Produkt entwickelt?«
  - Validation
    - Eignung bzw. der Wert eines Produktes bezogen auf seinen Einsatzzweck
    - »Wird das richtige Produkt entwickelt?«

Vorsicht: Der Begriff "Verifikation" wird auch mit einer anderen Bedeutung verwendet

Das V-Modell

- Prozessmodell: V-Darstellung



Das V-Modell
Weiterentwicklung

- Verbindlich für Bundeswehr und Behörden
- Software wird immer als Bestandteil eines informationstechnischen Systems (IT-System) angesehen
- Sehr umfangreiches Modell, das für eine konkrete Entwicklung angepaßt werden muss (Tailoring)
- Extrem viele Schritte in mehreren Subsystemen
- Extrem viele Rollen (Manager, Entwickler, ...)

Das V-Modell
Bewertung

- Integrierte, detaillierte Darstellung von
  - Systemerstellung
  - Qualitätssicherung
  - Konfigurationsmanagement und
  - Projektmanagement
- Generisches Vorgehensmodell mit definierten Möglichkeiten zum Maßschneidern
- Ermöglicht eine standardisierte Abwicklung von Systemerstellungs-Projekten
- Gut geeignet für große Projekte, insbesondere für eingebettete Systeme

Das V-Modell
Bewertung

- Konzepte, die für große eingebettete Systeme sinnvoll sind, werden unkritisch auf andere Anwendungstypen übertragen
- Für kleine und mittlere Softwareentwicklungen führt das V-Modell zu einer Software-Bürokratie
- Rechnerunterstützung erforderlich
- Die 25 definierten Rollen im V-Modell sind für den Durchschnitt der Softwareentwicklungen unrealistisch
- V-Modell ist nicht methodenneutral

Das Prototypen-Modell
Probleme traditioneller Prozessmodelle

- Auftraggeber ist oft nicht in der Lage, die Anforderungen an ein neues System explizit und/oder vollständig zu formulieren
  - Traditionelle Prozessmodelle verlangen jedoch zu Beginn der Entwicklung eine vollständige Spezifizierung der Anforderungen
- Während der Entwicklung ist oft eine wechselseitige Koordination zwischen Entwicklern und Anwendern erforderlich
  - Traditionelle Prozessmodelle beenden diese Kooperation, wenn die Anforderungen fertiggestellt sind
- Software-Entwicklungsabteilungen ziehen sich nach der Definitions-Phase vom Auftraggeber zurück
  - Präsentation des Ergebnisses erst nach der Fertigstellung
  - Diese Organisationsstruktur wird durch traditionelle Prozessmodelle unterstützt

Das Prototypen-Modell
Probleme traditioneller Prozessmodelle

- Manchmal gibt es unterschiedliche Lösungsmöglichkeiten
  - Diese müssen experimentell erprobt und mit dem Auftraggeber diskutiert werden
- Die Realisierbarkeit lässt sich manchmal theoretisch nicht garantieren
  - Beispiel: Echtzeitanforderungen
  - Diese speziellen Anforderungen müssen vor Abschluss der Definitionsphase realisiert werden
- In der Akquisitionsphase muss der Auftraggeber von der prinzipiellen Durchführbarkeit einer Idee oder der Handhabung überzeugt werden

$\Rightarrow$ Diese Probleme können durch das Prototypen-Modell (teilweise) gelöst werden

Das Prototypen-Modell
Software-Prototyp vs. Prototyp

- Unterschiede
  - Ein Software-Prototyp ist **nicht** das erste Muster einer großen Serie von Produkten.
    - Beispiel: Massenproduktion in der Autoindustrie
  - Ein Software-Prototyp zeigt ausgewählte Eigenschaften des Zielprodukte im praktischen Einsatz.
    - Er ist nicht nur eine Simulation des Zielproduktes.
    - Beispiele: Windkanal- oder Architekturmodell
- Gemeinsamkeiten
  - Anforderungen oder Entwicklungsprobleme klären
  - Diskussionsbasis
  - Entscheidungshilfe
  - Verwendung für experimentelle Zwecke
  - Sammeln von praktischen Erfahrungen

Das Prototypen-Modell

- Unterstützt systematisch die frühzeitige Erstellung ablauffähiger Modelle (Prototypen) des zukünftigen Produkts, um die Umsetzung von Anforderungen und Entwürfen in Software zu demonstrieren und mit ihnen zu experimentieren
- Vorgehensweise
  - *prototyping*
- 4 Arten von Prototypen
  - Demonstrationsprototyp
  - Prototyp im eigentlichen Sinne
  - Labormuster
  - Pilotsystem

Das Prototypen-Modell
Demonstrationsprototyp

- Dient zur Auftragsakquisition
- Soll dem potentiellen Auftraggeber einen ersten Eindruck vermitteln, wie ein Produkt für das vorgesehene Anwendungsgebiet im Prinzip aussehen kann.
- In der Regel werden solche Prototypen schnell aufgebaut
  - *rapid prototyping*
- Sie werden nach der Erfüllung ihrer Aufgaben »weggeworfen«

Das Prototypen-Modell
Prototyp im eigentlichen Sinne

- Wird parallel zur Modellierung des Anwendungsbereichs erstellt
- Soll Aspekte der Benutzungsschnittstelle oder Teile der Funktionalität veranschaulichen
- Trägt dazu bei, den Anwendungsbereich zu analysieren
- Provisorisches, ablauffähiges Software-System

Das Prototypen-Modell
Labormuster

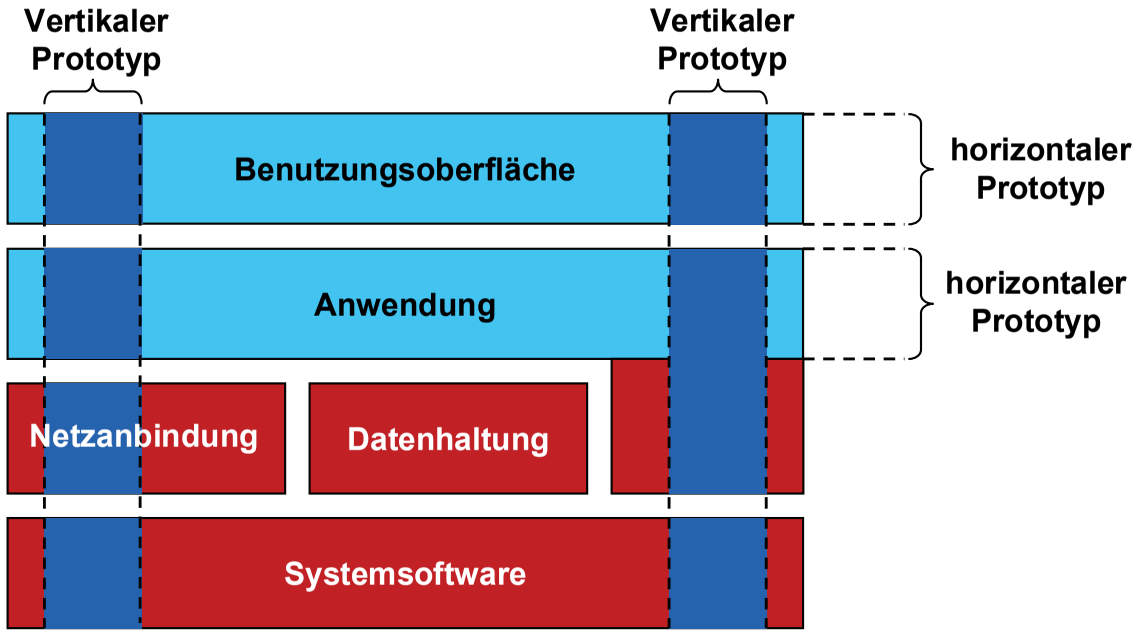
- Soll konstruktionsbezogene Fragen und Alternativen beantworten
- Demonstriert die technische Umsetzbarkeit des Produktmodells
- Nicht für Endbenutzer bestimmt
- Sollte technisch mit dem späteren Produkt vergleichbar sein

Das Prototypen-Modell
Pilotsystem

- Ist Kern eines Produkts
- Unterscheidung zwischen dem Prototyp und dem Produkt verschwindet
- Pilotsystem ist für die Benutzung in der Einsatzumgebung entworfen und nicht nur unter Laborbedingungen

Das Prototypen-Modell

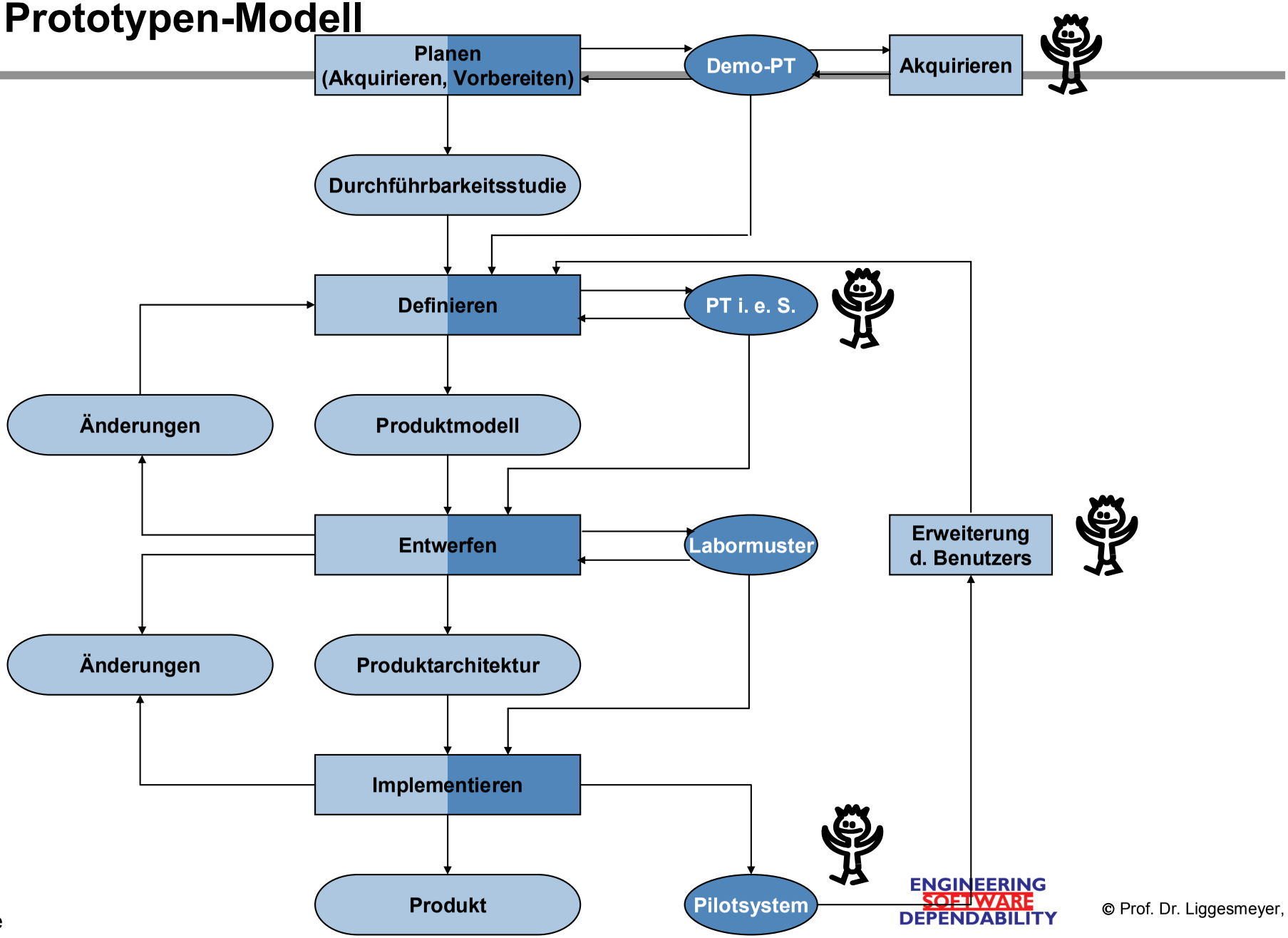
- Horizontaler Prototyp
  - Realisiert nur spezifische Ebenen des Systems
  - Die betreffende Ebene wird möglichst vollständig realisiert.
- Vertikaler Prototyp
  - Implementiert ausgewählte Teile des Zielsystems vollständig durch alle Ebenen hindurch
  - Dort geeignet, wo die Funktionalitäts- und Implementierungsoptionen noch offen sind.



Das Prototypen-Modell
Prototyp vs. fertiges Software-System

- Prototypen dienen nur zur Klärung von Problemen
- Ein Prototyp ist Teil der Produktdefinition
- Prototypen werden inkrementell weiterentwickelt, um ein marktfähiges Produkt zu erhalten

Das Prototypen-Modell



ENGINEERING SOFTWARE DEPENDABILITY

Das Prototypen-Modell
Bewertung

- \+ Reduzierung des Entwicklungsrisikos durch frühzeitigen Einsatz von Prototypen
- \+ Prototypen können sinnvoll in andere Prozessmodelle integriert werden
- \+ Prototypen können heute durch geeignete Werkzeuge schnell erstellt werden
- \+ *Prototyping* verbessert die Planung von Software-Entwicklungen
- \+ Labormuster fördern die Kreativität für Lösungsalternativen
- \+ Starke Rückkopplung mit dem Endbenutzer und dem Auftraggeber
- – Höherer Entwicklungsaufwand, da Prototypen zusätzlich erstellt werden
- – Gefahr, dass ein »Wegwerf«-Prototyp Teil des Endprodukts wird
- – Verträge für die Software-Erstellung berücksichtigen noch nicht das Prototypen-Modell
- – Prototypen werden oft als Ersatz für die fehlende Dokumentation angesehen
- – Beschränkungen und Grenzen von Prototypen sind oft unbekannt

Das Prototypen-Modell
Voraussetzungen

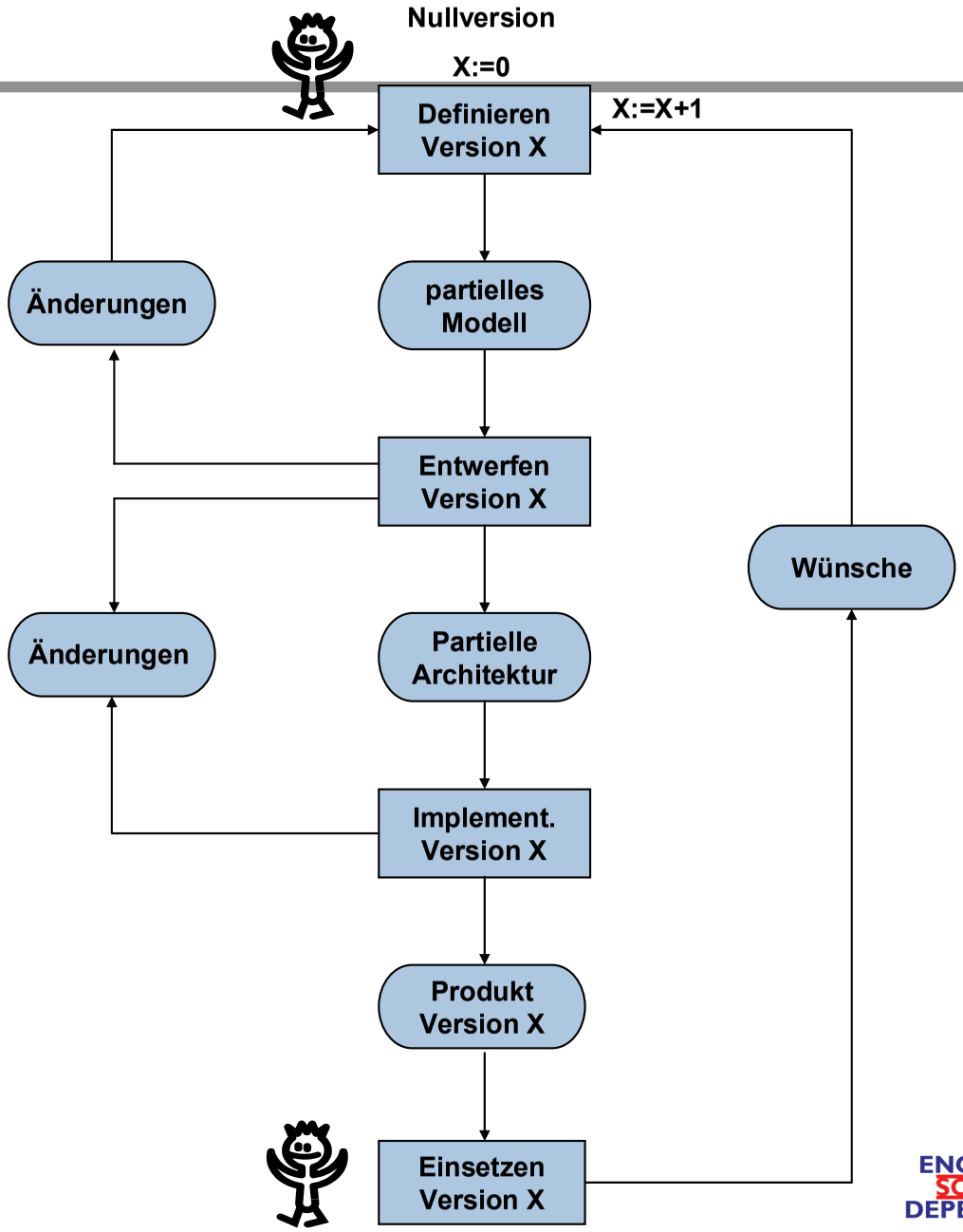
- Ausreichendes Wissen über das Anwendungsgebiet muss vorhanden sein
- Nur auf der Basis schriftlicher Dokumente kann kein Prototyp erstellt werden ⇒ Die Entwickler müssen Zugang zu den Benutzern haben
- Die Endbenutzer müssen am *Prototypingprozess* beteiligt werden
- Die Benutzerbeteiligung ersetzt nicht die kreativen Ideen der Entwickler
- Alle beteiligten Personengruppen müssen in direktem Kontakt stehen.
- Prototypen müssen dokumentiert werden
- Die Vorgehensweise hängt von der untersuchten Fragestellung ab
- Geeignete Werkzeuge müssen verfügbar sein

⇒ Motto: »Redo until Right«

Das evolutionäre/inkrementelle Modell

- Ausgangspunkt
  - Kern- oder Mussanforderungen des Auftraggebers
  - Nur dieser Produktkern wird entworfen und implementiert.
  - Das Kernsystem wird an den Auftraggeber ausgeliefert.
  - Der Auftraggeber sammelt Erfahrungen.
    - Daraus ermittelt er seine Produktanforderungen für eine erweiterte Version.
- Charakteristika
  - Das Software-Produkt wird allmählich und stufenweise entwickelt
  - Pflegeaktivitäten werden ebenfalls als Erstellung einer neuen Version betrachtet.
  - Gut geeignet, wenn der Auftraggeber seine Anforderungen noch nicht vollständig überblickt
    - »I can't tell you what I want, but I'll know it when I see it«
  - Die Entwicklung ist code-getrieben
    - Konzentration auf jeweils lauffähige Teilprodukte.

Das evolutionäre/inkrementelle Modell



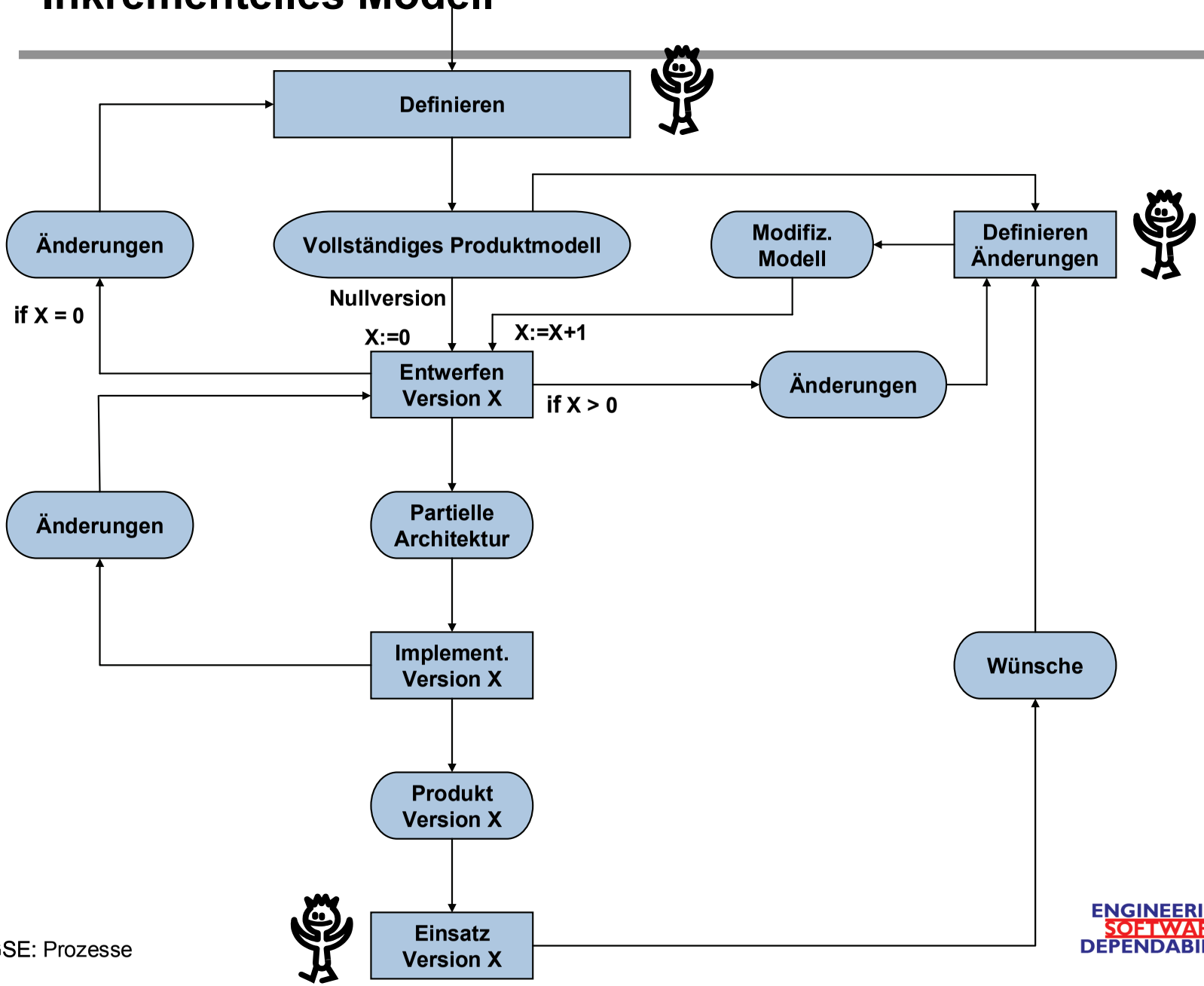
Das evolutionäre/inkrementelle Modell
Bewertung evolutionäres Modell

+ Der Auftraggeber erhält in kürzeren Zeitabständen einsatzfähige Produkte.
+ Kombination mit dem Prototypen-Modell möglich
+ Erfahrungen aus dem Produkteinsatz können in die nächste Version eingebracht werden.
+ Ein Produkt wird in einer Anzahl kleiner Arbeitsschritte überschaubarer Größe erstellt.
– Gefahr, dass in nachfolgenden Versionen die komplette Systemarchitektur überarbeitet werden muss
– Gefahr, dass die Nullversion nicht flexibel genug ist, um sich an ungeplante Evolutionspfade anzupassen.

Das evolutionäre/inkrementelle Modell
Inkrementelles Modell

- Vermeidet die Nachteile des evolutionären Modells
- Anforderungen an das zu entwickelnde Produkt werden möglichst vollständig erfasst und modelliert.
- Nur ein Teil der Anforderungen wird entworfen und implementiert.
- Anschließend wird die nächste Ausbaustufe realisiert.

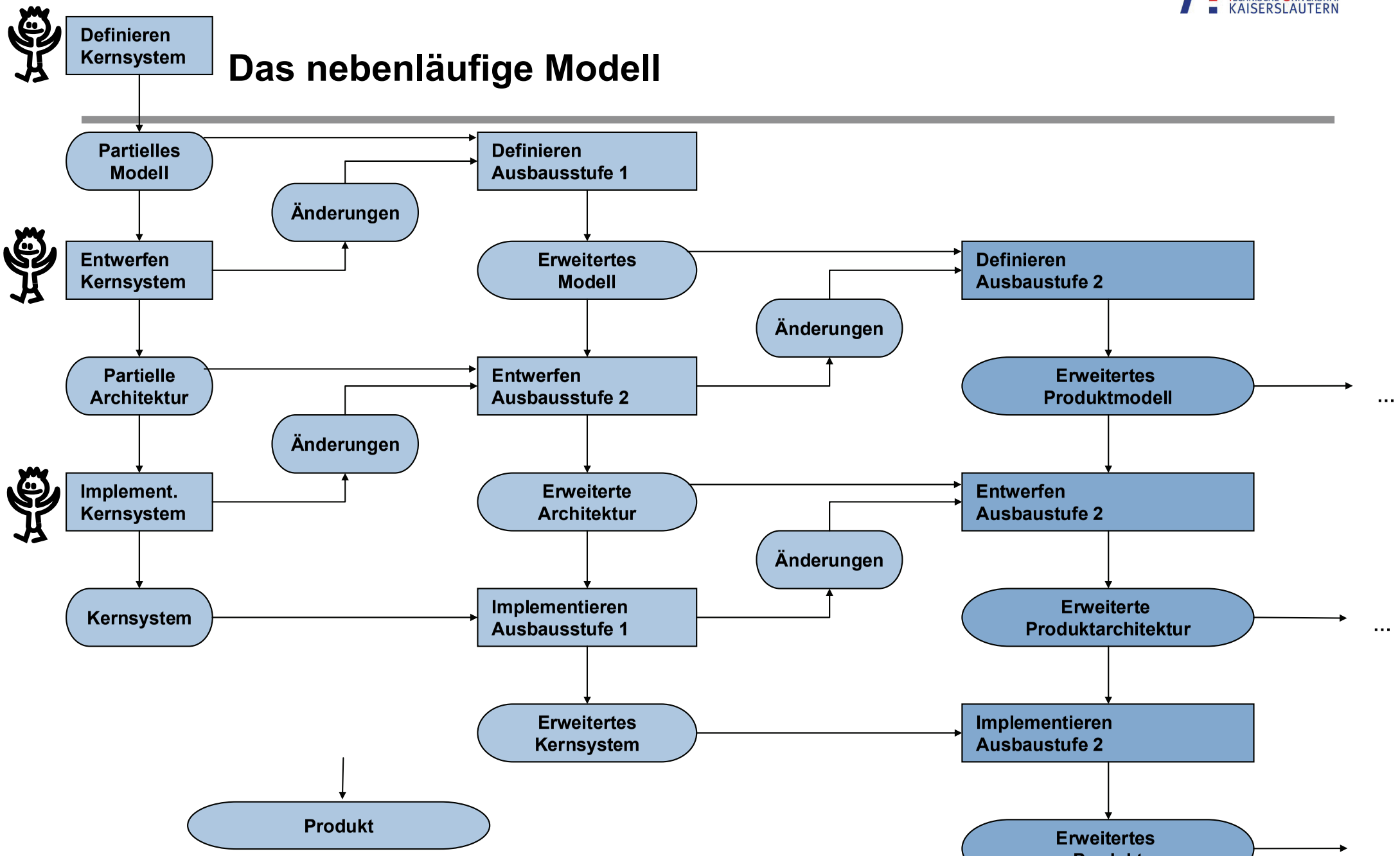
Das evolutionäre/inkrementelle Modell
Inkrementelles Modell



Das nebenläufige Modell

- Stammt aus der Fertigungsindustrie
- Alle Entwicklungsabteilungen einschließlich
  - Fertigung
  - Marketing
  - Vertrieb in einem Team vereint
- So viel wie möglich soll parallel ablaufen.
- Ziel: termingerechte Fertigstellung (time-to-market)

Das nebenläufige Modell



Das nebenläufige Modell
Charakteristika

- Es wird versucht einzelne Aktivitäten zu parallelisieren.
  - Durch organisatorische und technische Maßnahmen
- Förderung des auf Problemlösung gerichteten Zusammenarbeitens der betreffenden Personengruppen
- Erfahrungen der betroffenen Personengruppen werden frühzeitig zusammengebracht.
- Zeitverzögerungen werden reduziert durch
  - teilweise Parallelisierung vorwiegend sequentiell organisierter Arbeiten
  - Minimierung des Ausprobierens *(trial and error)*
    - Begrenzung des Improvisierens
  - Reduktion der Wartezeiten
    - Zwischen arbeitsorganisatorisch verbundenen Aktivitäten
- Ziel: vollständiges Produkt ausliefern

Das nebenläufige Modell
Bewertung

- \+ Frühes Erkennen und Eliminieren von Problemen durch Beteiligung aller betroffenen Personengruppen
- \+ Optimale Zeitausnutzung
- – Fraglich, ob das Ziel »right the first time« in der Software-Technik zu erreichen ist
- – Risiko, dass die grundlegenden und kritischen Entscheidungen zu spät getroffen werden und dadurch Iterationen nötig werden
- – Hoher Planungs- und Personalaufwand, um Fehler zu vermeiden bzw. Probleme frühzeitig zu antizipieren

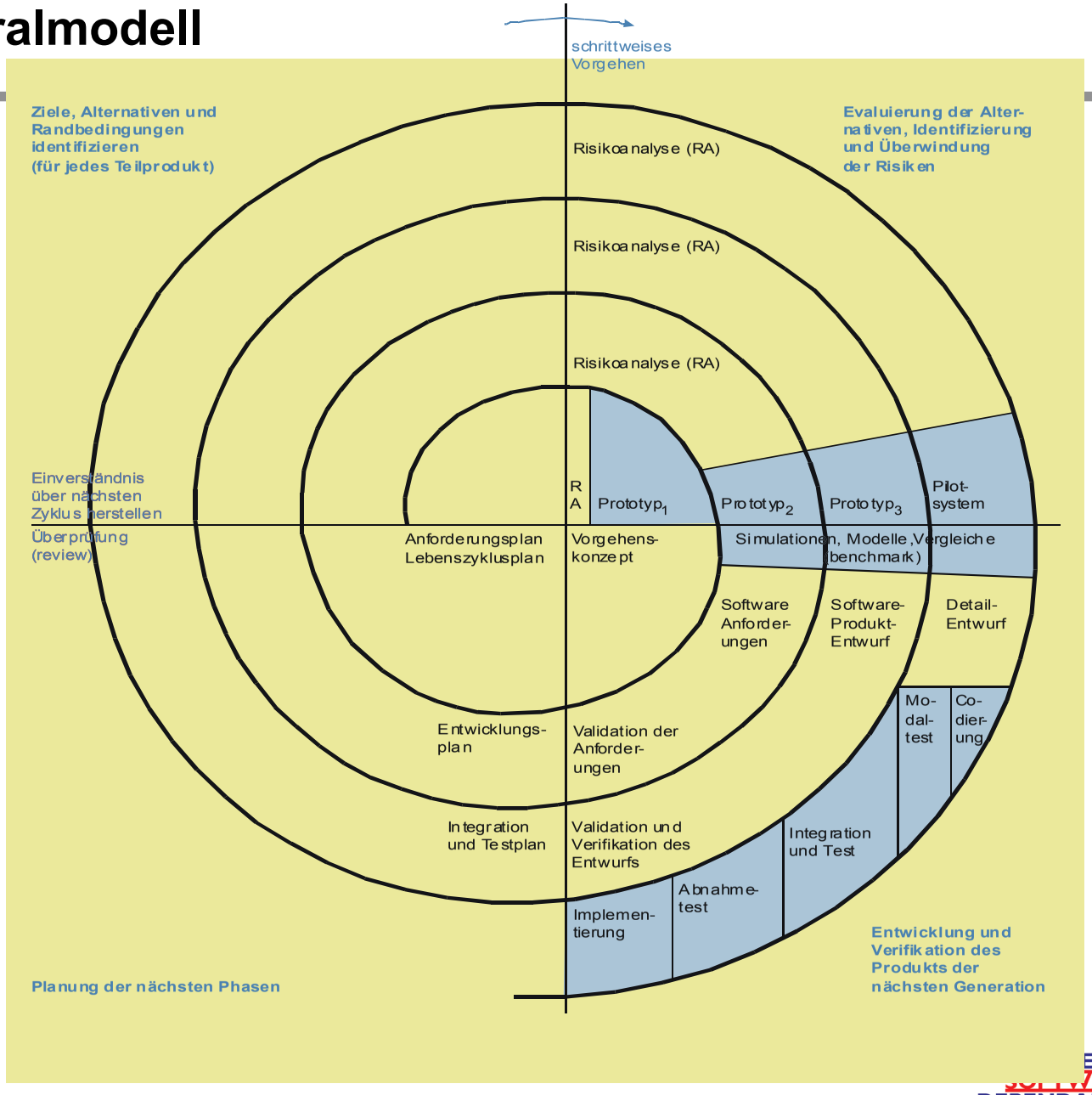
Das Spiralmodell

- Metamodell
  - Für jedes Teilprodukt und für jede Verfeinerungsebene vier zyklische Schritte
- Schritt 1
  - Identifikation der Ziele des zu erstellenden Teilprodukts
    - Leistung, Funktionalität usw.
  - Alternative Möglichkeiten, um das Teilprodukt zu realisieren
    - Entwurf A, Entwurf B, Wiederverwendung, Kauf usw.
  - Randbedingungen, die bei den verschiedenen Alternativen zu beachten sind
    - Kosten, Zeit, Schnittstellen usw.
- Schritt 2
  - Evaluierung der Alternativen
    - Berücksichtigung der Ziele und Randbedingungen
  - Zeigt die Evaluierung Risiken, dann eine kosteneffektive Strategie entwickeln um die Risiken zu überwinden

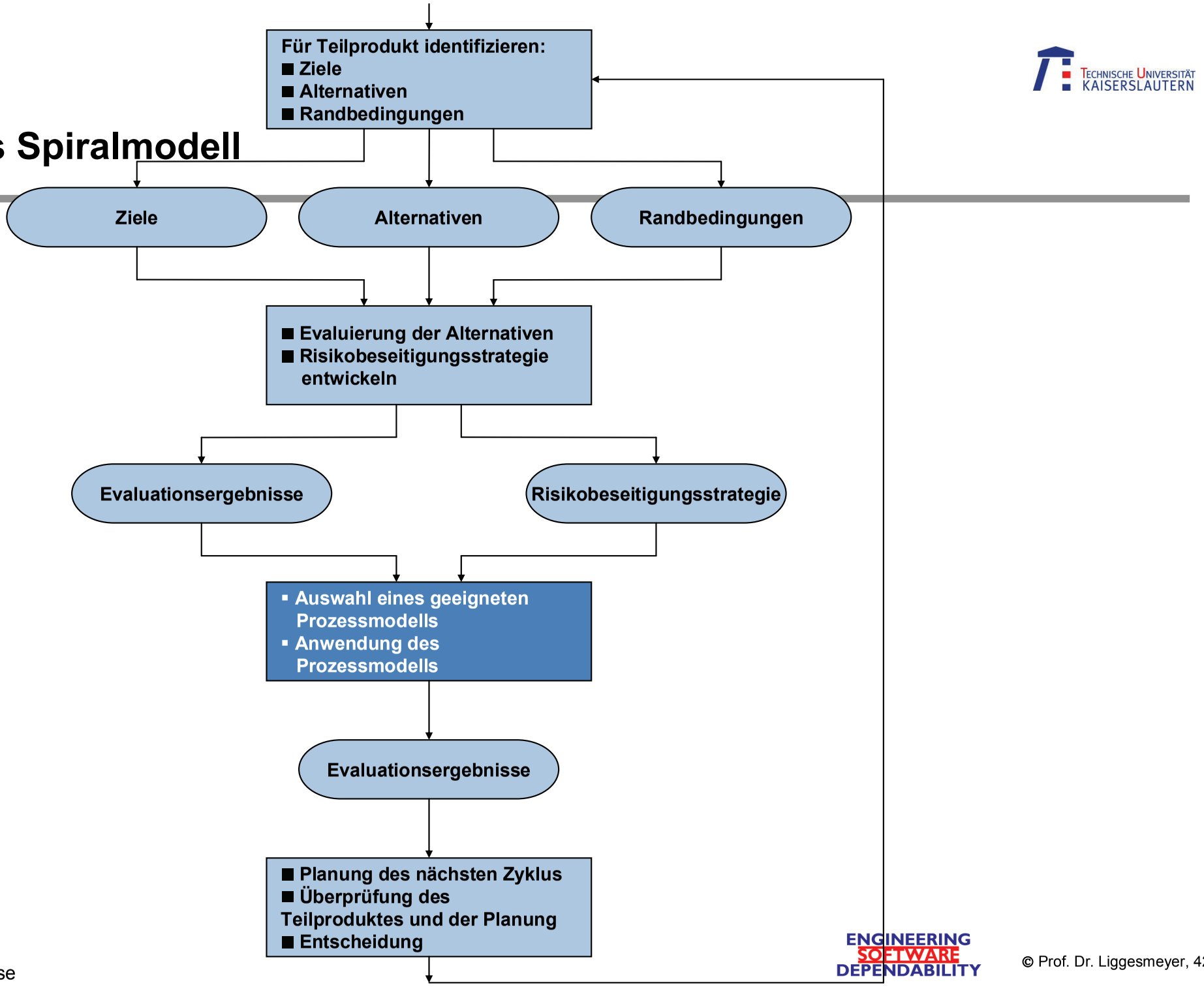
Das Spiralmodell

- Schritt 3
  - In Abhängigkeit der verbleibenden Risiken
    - Festlegung des Prozessmodells für diesen Schritt
  - Es kann eine Kombination verschiedener Modelle vorgenommen werden, wenn dadurch das Risiko minimiert wird.
- Schritt 4
  - Planung des nächsten Zyklus einschließlich der benötigten Ressourcen
    - Dies beinhaltet auch eine mögliche Aufteilung eines Produktes in Komponenten.
    - Diese werden dann unabhängig weiterentwickelt.
  - Überprüfung *(review)* der Schritte 1 bis 3
    - Einschließlich der Planung für den nächsten Zyklus
      - Durch die betroffenen Personengruppen oder Organisationen
  - Einverständnis *(commitment)* über den nächsten Zyklus herstellen.

Das Spiralmodell



Das Spiralmodell



Das Spiralmodell

- Charakteristika
  - Risikogetriebenes Modell
  - Jede Spirale stellt einen iterativen Zyklus durch dieselben Schritte dar
  - Die Ziele für jeden Zyklus werden aus den Ergebnissen des letzten Zyklus abgeleitet
  - Separate Spiralzyklen können für verschiedene Software-Komponenten durchgeführt werden
  - Keine Trennung in Entwicklung und Wartung
- Ziel
  - Beginne im Kleinen
  - Halte die Spirale so eng wie möglich
  - Erreiche so die Entwicklungsziele mit minimalen Kosten
- Bei der Zielbestimmung werden auch Qualitätsziele aufgeführt
- Für jede Aktivität und jeden Ressourcenverbrauch wird gefragt
  - »Wie viel ist genug?«
  - Dadurch wird ein »Overengineering« vermieden

Das Spiralmodell
Bewertung

- \+ In periodischen Intervallen Überprüfung des Prozessablaufs in Abhängigkeit von den Risiken
- \+ Prozessmodell wird nicht für die gesamte Entwicklung festgelegt
- \+ Integration anderer Prozessmodelle als Spezialfälle
- \+ Fehler und ungeeignete Alternativen werden frühzeitig eliminiert
- \+ Flexibles Modell
- \+ Entwicklung kann wesentlich leichter umdirigiert werden, wenn die Erkenntnisse dies erfordern
- \+ Unterstützt die Wiederverwendung von Software durch die Betrachtung von Alternativen
- – Hoher Managementaufwand
- – Für kleine und mittlere Projekte weniger gut geeignet
- – Wissen über das Identifizieren und Managen von Risiken noch nicht weit genug verbreitet

Überblick über die Prozessmodelle

Prozessmodell	Primäres Ziel	Antreibendes Moment	Benutzer-beteiligung	Charakteristika
Wasserfall	minimales Management	Dokumente	gering	sequentiell, volle Breite
V-Modell	maximale Qualität	Dokumente	gering	sequentiell, volle Breite, V&V
Prototypen	Risiko-minimierung	Code	hoch	nur Teilsysteme (horizontal oder vertikal)
Evolutionär	minimale Zeit	Code	mittel	sofort: nur Kernsystem
Inkrementell	minimale Zeit & Risiko	Code	mittel	volle Definition, dann zunächst nur Kernsystem
Nebenläufig	minimale Zeit	Zeit	hoch	volle Breite, nebenläufig
Spiralmodell	Risiko-minimierung	Risiko	mittel	Entscheidung pro Zyklus