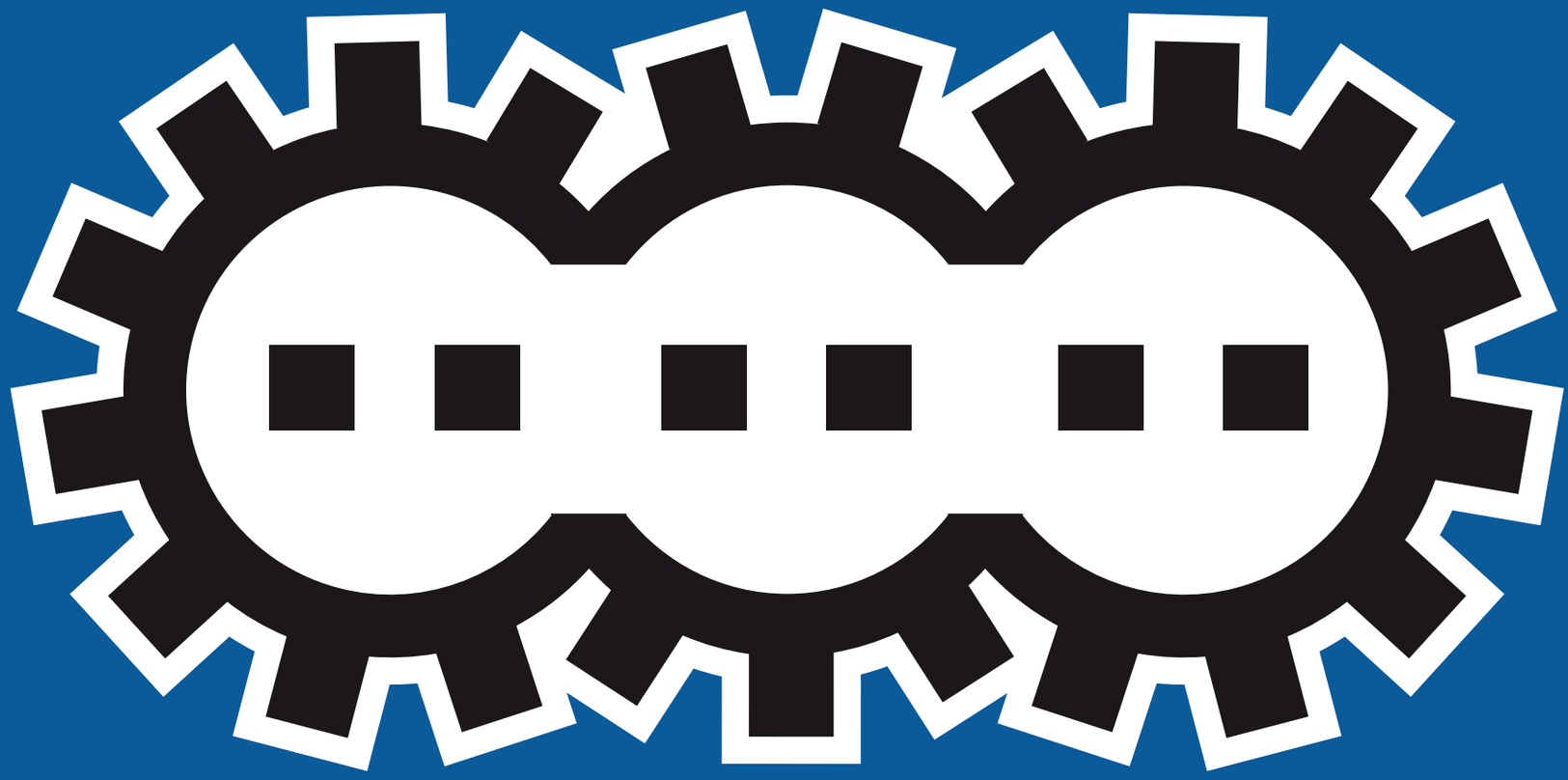
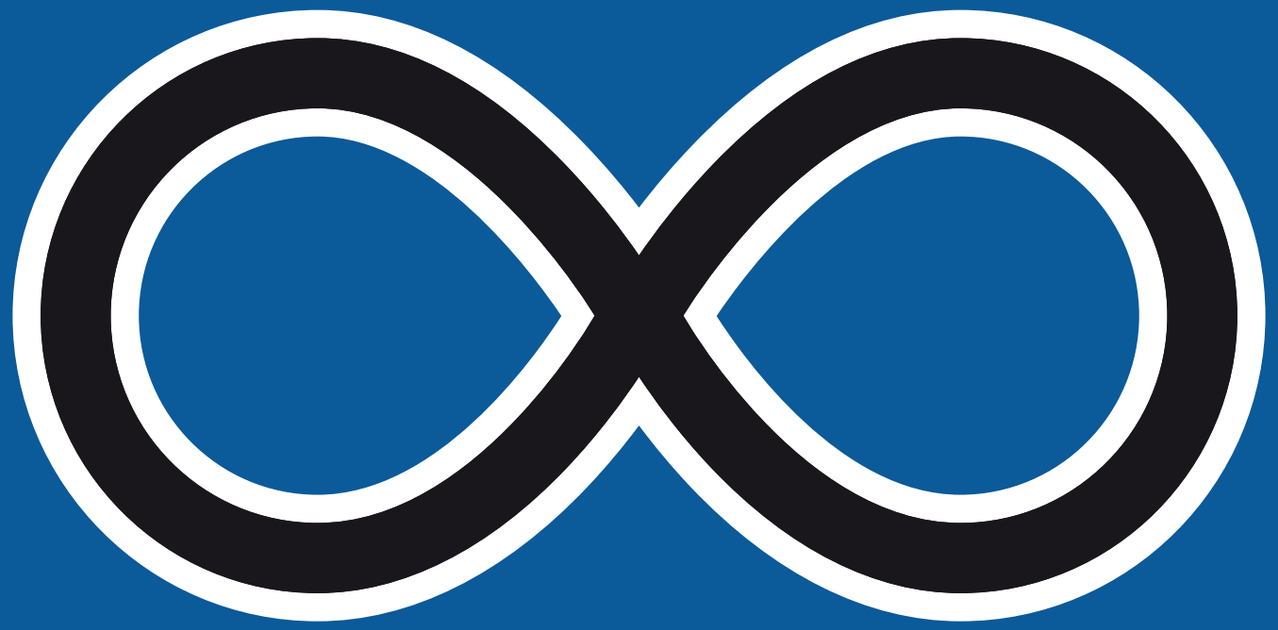


Software Engineering 2

01 - Entwicklungsprozesse

Prof. Dr. Stefan Betermieux | Fakultät Informatik | Hochschule Furtwangen



Wiederholung

Prof. Dr. Stefan Betermieux | Fakultät Informatik | Hochschule Furtwangen

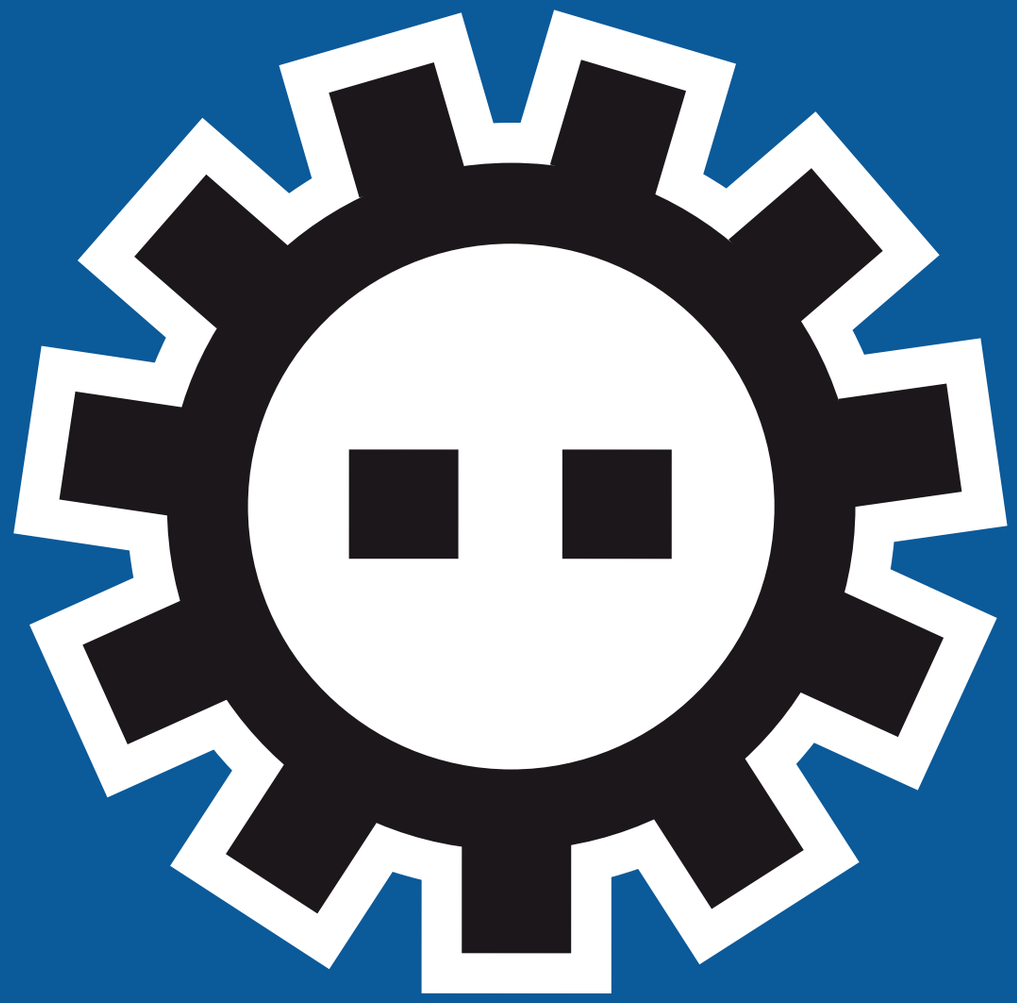
Definition (2)

- *Zielorientierte Bereitstellung* und *systematische Verwendung* von
  - Prinzipien,
  - Methoden,
    - » Konzepten,
    - » Notationen und
  - Werkzeugen
- für die arbeitsteilige, ingenieurmäßige Entwicklung und Anwendung von umfangreichen Software-Systemen.
- Zielorientiert bedeutet hierbei die Berücksichtigung z.B. von Kosten, Zeit, Qualität.

Quelle: H. Balzert 96: Lehrbuch der Software-Technik (Band 1)

Ingenieurmäßiges Vorgehen

- Ingenieurwesen ist auf erfahrungsgeleitetes Vorgehen angewiesen
  - »Normales Vorgehen«
- Das Vorgehen wird immer irgendwo von vorherigen Erfahrungen abweichen
  - »Radikales Vorgehen«
- Ingenieurmäßiges Vorgehen besteht darin, solche Abweichungen zu vermeiden/gering zu halten
  - Radikales Vorgehen nur, wo unbedingt nötig!
- Ansätze für normales Vorgehen:
  - Was sind die typischen Probleme in der SW-Entwicklung?
  - Was sind die dazu passenden Lösungsansätze des SE?

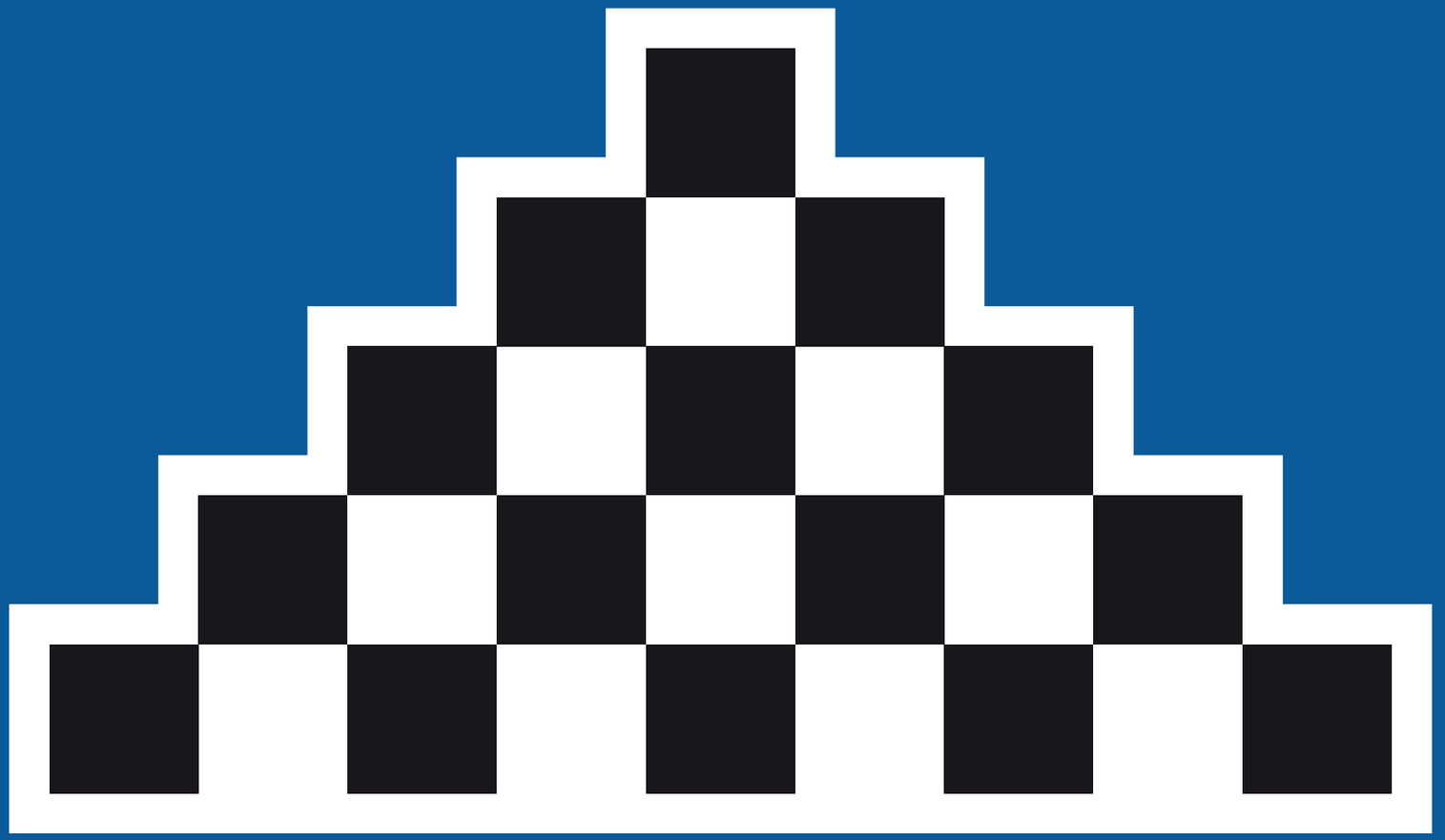


Motivation

Prof. Dr. Stefan Betermieux | Fakultät Informatik | Hochschule Furtwangen

Übersicht

- Code & Fix
- Wasserfallmodell
- Grundlegende Aktivitäten
- Inkrementelle Entwicklung
- Rational Unified Process
- Agiles Manifest
- Agile Softwareentwicklung
- Extreme Programming
- Scrum



Grundlagen

Prof. Dr. Stefan Betermieux | Fakultät Informatik | Hochschule Furtwangen

Code & Fix

- Vorgehensweise (nicht nur) in den Anfangstagen der Software-Entwicklung
- Grundprinzip (Try-and-Error):
  - schreibe ein Programm
  - finde und behebe die Fehler im Programm
- Probleme:
  - keinerlei systematische Vorgehensweise oder Fortschrittskontrolle
  - keine Definitionsphase vor der Implementierung (→ geringe Benutzerakzeptanz)
  - permanente Umstrukturierungen bei Fehlerbehebungen
  - fehlende Softwarearchitektur

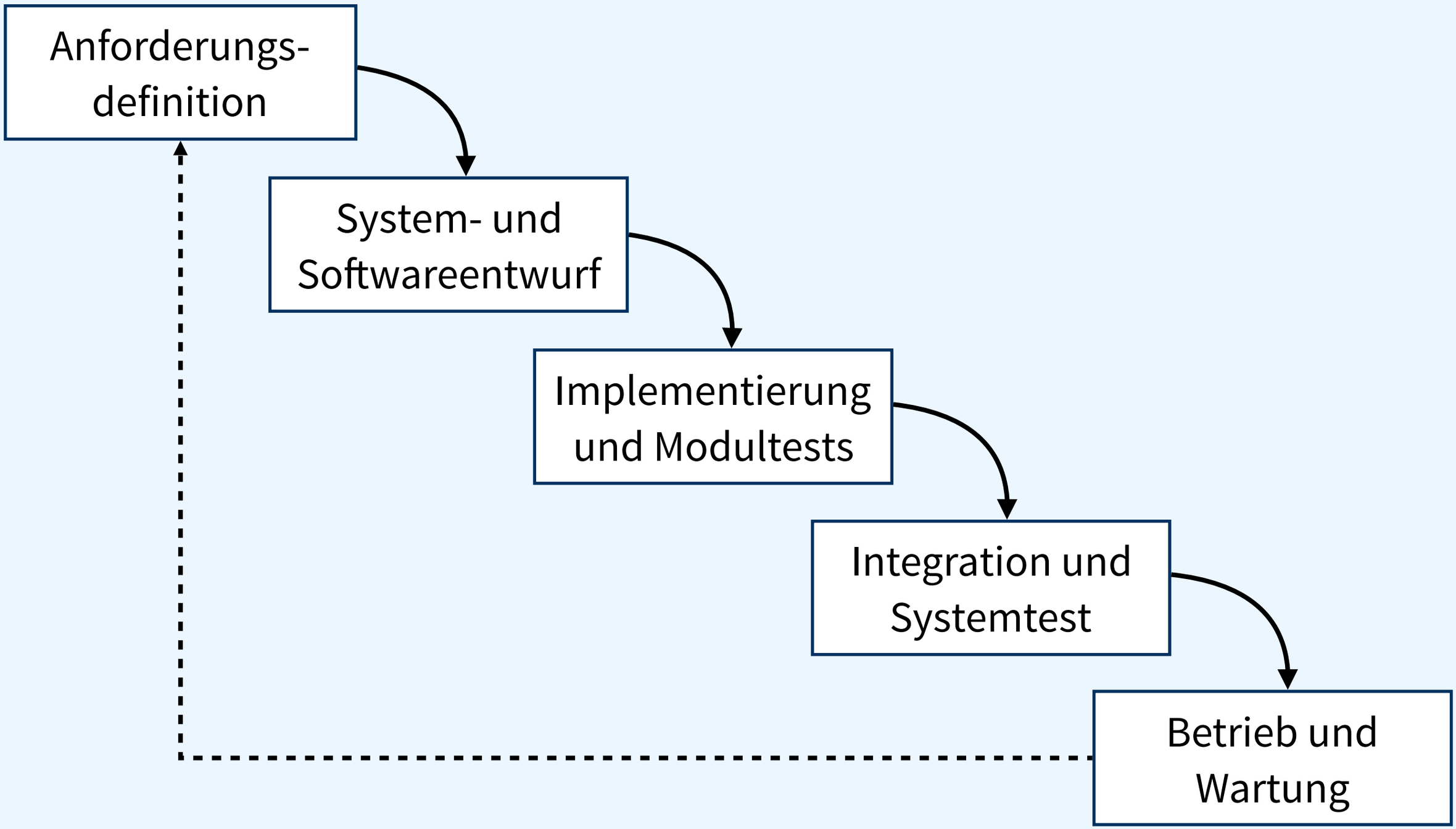
➡ Kein Vorgehen für ernsthafte Projekte.

Prozessdefinition

- Ein unstrukturiertes Vorgehen in der Softwareentwicklung ist mit (zu) großen Risiken verbunden
- Die Softwareentwicklung benötigt einen organisatorischen Rahmen – festgelegt durch ein Prozessmodell (auch Vorgehensmodell genannt)
- Prozessmodell zum strukturierten Vorgehen:
  - legt die Reihenfolge des Arbeitsablaufs fest
  - beschreibt die durchzuführenden Aktivitäten und zu erstellenden Produkte
  - definiert Vor- und Nachbedingungen für Aktivitäten
  - unabhängig von der Art der zu entwickelnden Software

Wasserfallmodell

Überblick



Charakterisierung

Das Wasserfallmodell reduziert die Komplexität eines Entwicklungsvorhabens durch dessen Zerlegung in Phasen

- Prinzipielles Vorgehen:
  - jede Phase in der gesamten Breite durchführen
  - sequentieller Durchlauf
- Problematisch:
  - typischerweise gibt es Rückkopplungen von einer Phase zu einer vorherigen
  - die in der jeweiligen Phase erzeugten Dokumente müssen dann eventuell geändert werden
  - spätere Änderungen von Anforderungen oder Fehlerbehebungen führen zu nicht geplanten Rücksprüngen in frühere Phasen

Grundlegende Aktivitäten

Prozessunabhängige Aktivitäten im Software-Engineering:

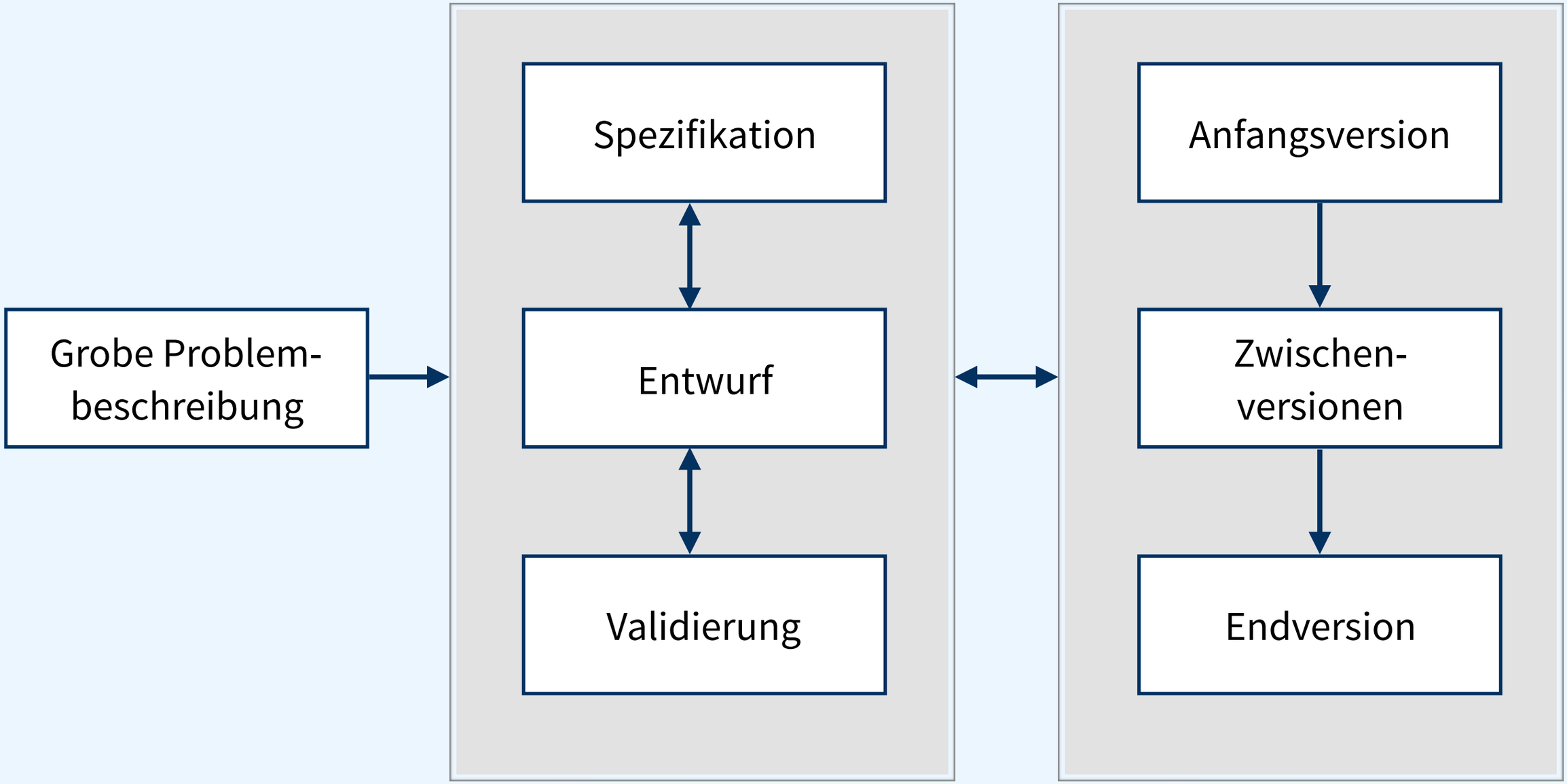
- Softwarespezifikation
  - die Funktionen der Software und die Beschränkungen ihrer Benutzung müssen definiert werden
- Softwareentwurf und -implementierung
  - die Software, die diese Spezifikation erfüllt, muss erstellt werden
- Softwarevalidierung
  - die Software muss validiert werden um sicherzustellen, dass sie die Kundenwünsche (tatsächlich) erfüllt
- Softwareevolution
  - die Software muss sich weiterentwickeln, um mit den sich ändernden Bedürfnissen des Kunden Schritt zu halten

Inkrementelle Entwicklung

Inkrementelle Entwicklung

- Verknüpft die Aktivitäten der
  - Spezifikation,
  - Entwurf und
  - Validierung.
- Diese Aktivitäten werden nicht als separate Abläufe betrachtet
  - das System wird als Folge von Versionen (→ Inkremente) entwickelt, wobei jede Version neue Funktionalität zu der vorherigen hinzufügt
- Entwicklung einer Anfangsimplementierung:
  - frühzeitige Rückmeldung von Benutzern wird eingefordert
  - frühzeitige Bewertung der Implementierung ist möglich

Ablauf



Vorteile

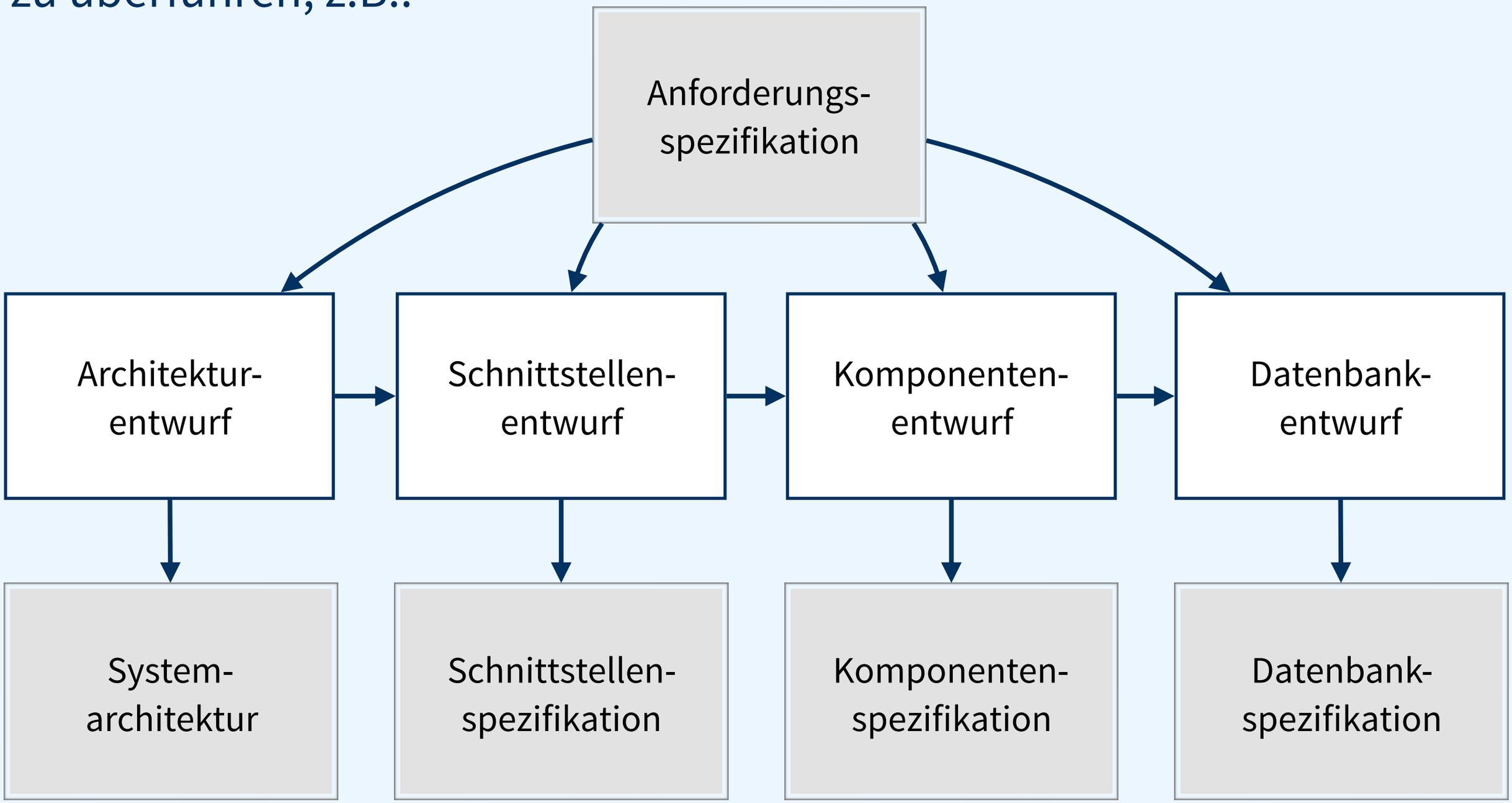
- Die Kosten für die Anpassung an sich ändernde Kundenanforderungen werden reduziert.
- Es ist einfacher, Rückmeldungen der Kunden zu bereits fertiggestellten Teilen der Entwicklungsarbeit zu bekommen. Den Fortschritt anhand von Entwurfsdokumenten zu beurteilen, fällt Kunden meist schwer.
- Eine schnelle Auslieferung und Installation von verwendungsfähiger Software an den Kunden ist auch dann möglich, wenn noch nicht die gesamte Funktionalität enthalten ist.
- Achtung!
  - (große) Systeme benötigen eine stabile Architektur
  - die Architektur muss im voraus geplant statt inkrementell entwickelt werden

Softwarespezifikation

- Prozess zum Verstehen und Definieren der vom System verlangten Funktionen sowie der Beschränkungen, denen die Entwicklung und der Betrieb unterliegen
- Besonders kritische Phase, da Fehler unweigerlich zu späteren Problemen beim Entwurf und der Implementierung des Systems führen
- Die Anforderungsanalyse hilft, ein vereinbartes Anforderungsdokument zu erstellen, das ein System spezifiziert, welches die Anforderungen der Projektbeteiligten erfüllt
- Zwei Detailstufen:
  - die Endnutzer und Kunden brauchen eine grobe Aufstellung der Anforderungen
  - die Entwickler benötigen eine detaillierte Systemspezifikation

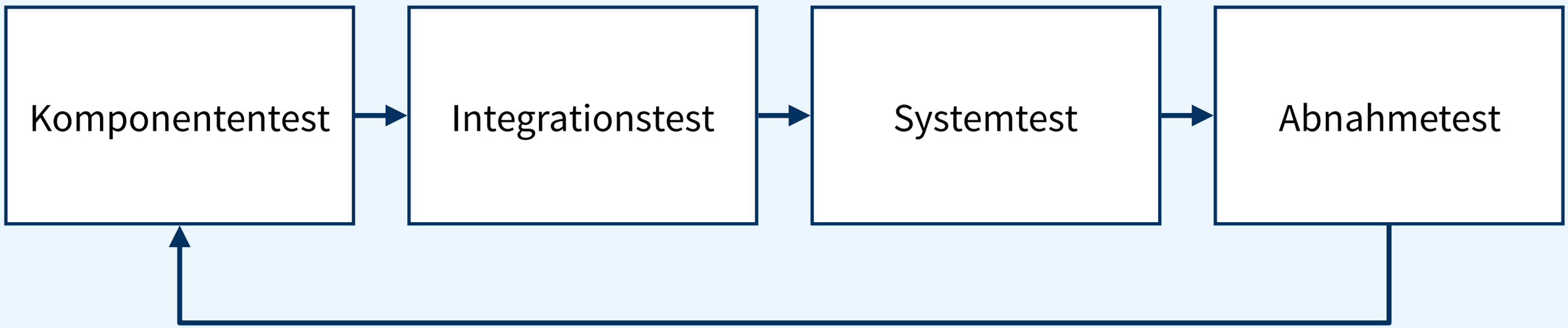
Softwareentwurf

Prozess, um die Anforderungsspezifikation in ein ausführbares System zu überführen, z.B.:

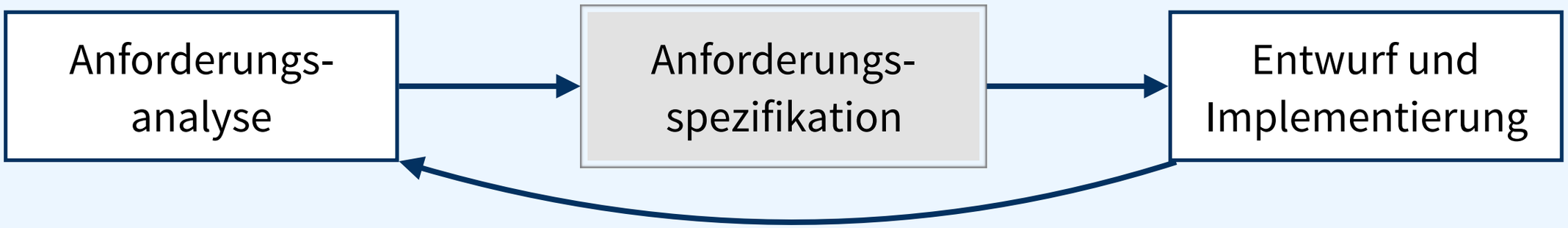


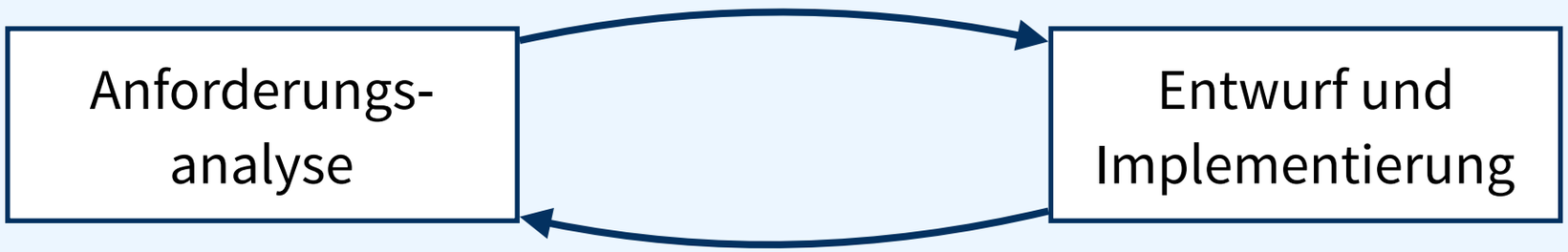
Softwarevalidierung

- Soll zeigen, dass ein System sowohl den Erwartungen des Kunden als auch seiner Spezifikation entspricht
- Iterativer Testprozess: Fehler in Komponenten können zum Beispiel während des Systemtests entdeckt werden



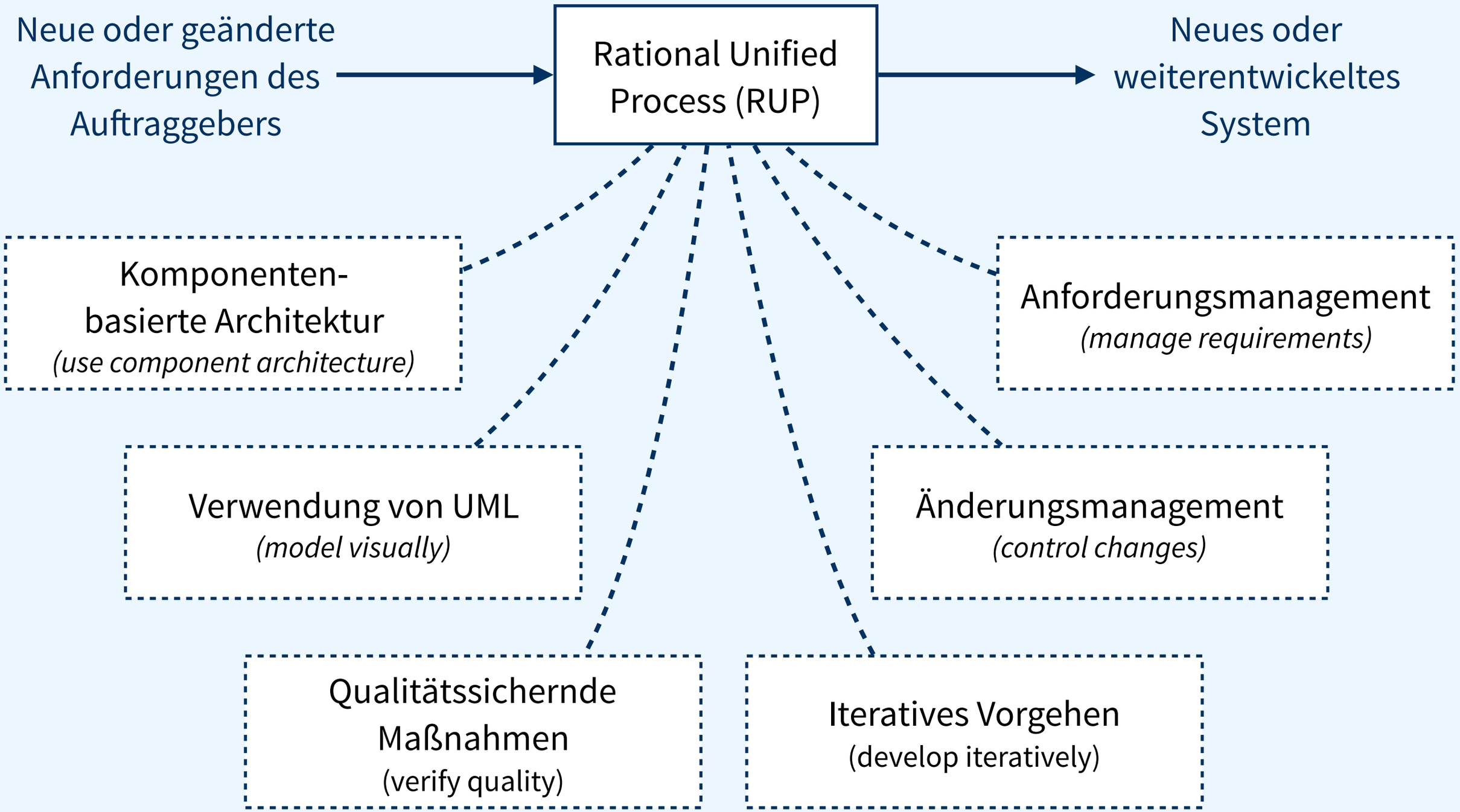
Planbasierte vs. agile Entwicklung





Rational Unified Process (RUP)

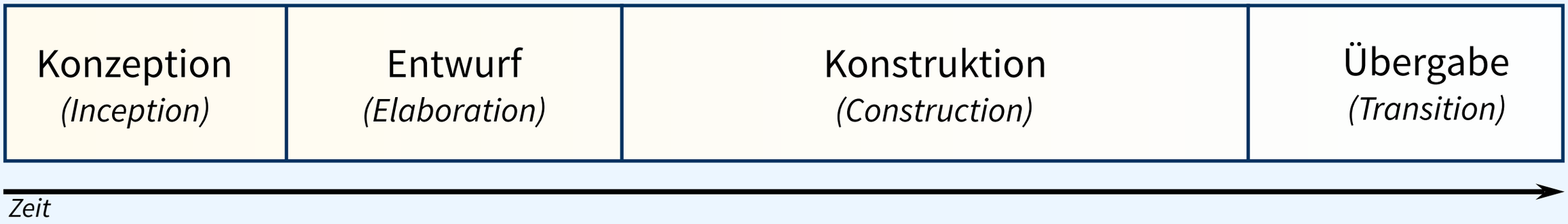
RUP: Best Practices in der Softwareentwicklung



Rational Unified Process (RUP)

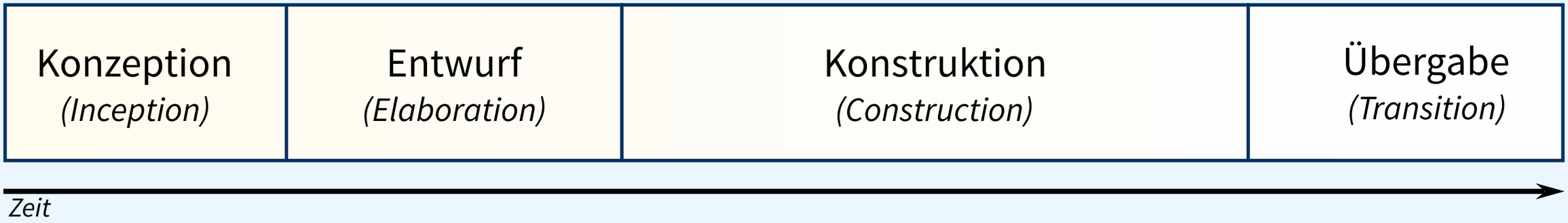
- Die Entwicklung von RUP ging mit der von UML einher
- Hybrides Vorgehensmodell
  - umfasst Elemente der generischen, phasenorientierten Prozessmodelle
  - inkrementelle Auslieferung
  - Entwicklung von Prototypen
- Beschreibung von drei Perspektiven:
  - dynamische Perspektive zeigt die Entwicklungsphasen im Zeitverlauf
  - statische Perspektive zeigt die Prozessaktivitäten, die ausgeführt werden
  - praktische Perspektive schlägt bewährte Techniken zur Ausführung der Aktivitäten vor

Fließende Phasen-Übergänge (1)



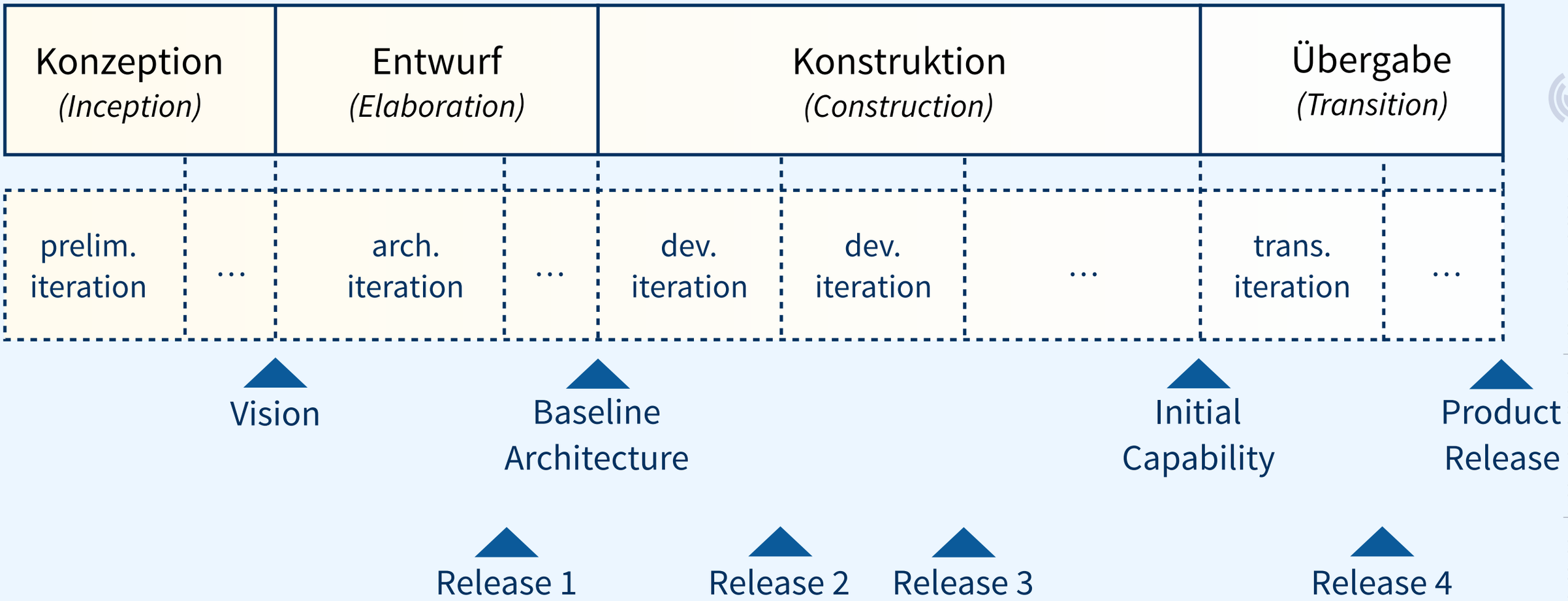
- Konzeption
  - Anwendungsszenarien entwickeln und deren Relevanz bewerten
  - Systemkontext klären
  - Projektumfang definieren
- Entwurf
  - Architekturrahmen festlegen
  - Projektplan entwickeln
  - größte Projektrisiken identifizieren
  - Use Cases erstellen

Fließende Phasen-Übergänge (2)



- Konstruktion
  - System entwerfen, programmieren und testen
  - Systemkomponenten integrieren
  - Funktionierendes System mit entsprechender Dokumentation abliefern
- Übergabe
  - System von der Entwicklung an die Benutzer übergeben
  - System in seiner realen Arbeitsumgebung installieren

Phasen und Iterationen

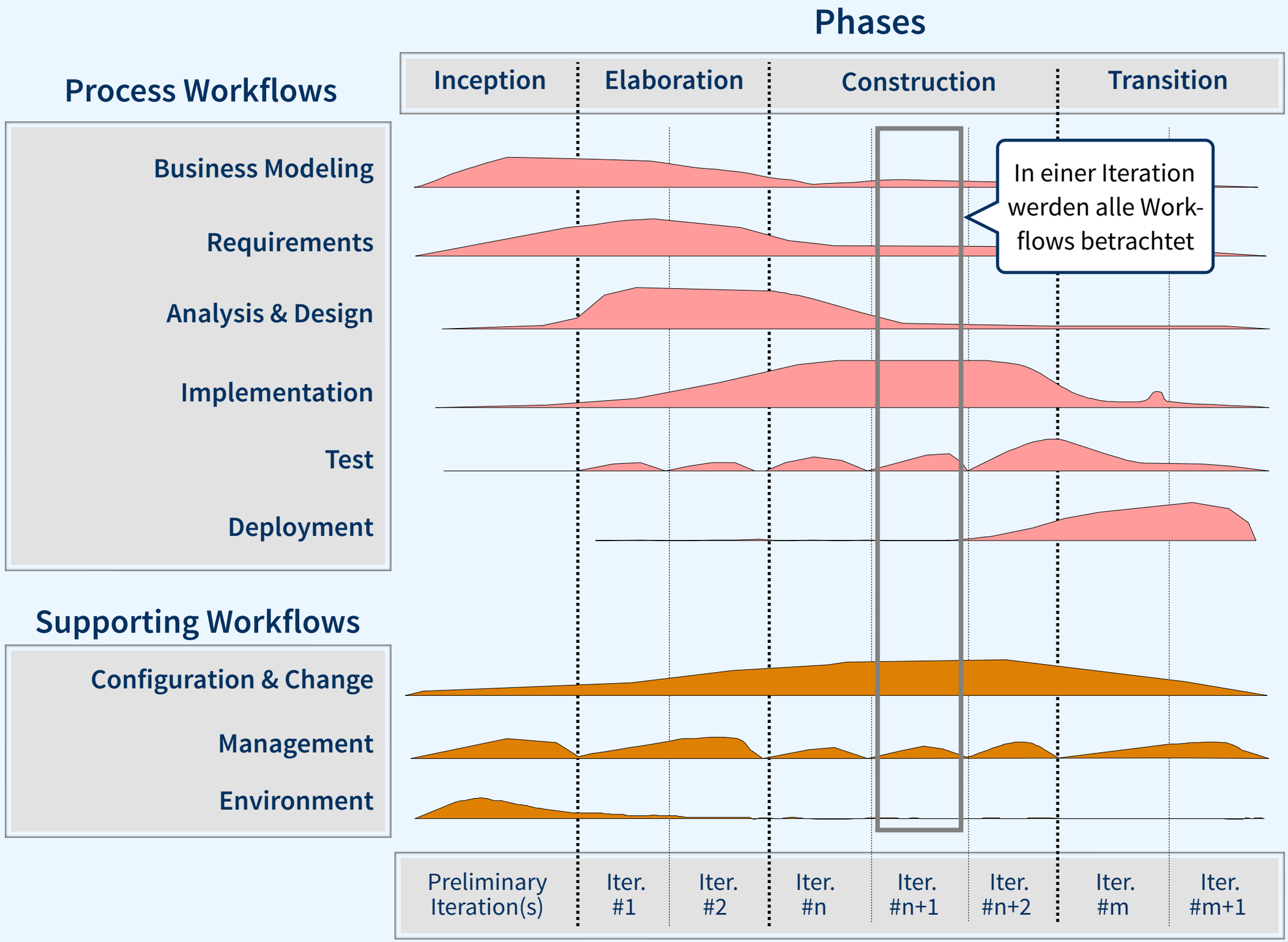


Eine Iteration ist eine Folge von Aktivitäten, die ein mess- und bewertbares Ergebnis (z.B. ein Release) produzieren!

Statische Perspektive

- Statische Perspektive: Aktivitäten während des Entwicklungsprozesses
- 6 Arbeitsabläufe *(Process Workflows)*:
  - Geschäftsprozessmodellierung *(Business Modeling)*
  - Anforderungsanalyse *(Requirements)*
  - Analyse und Entwurf *(Analysis und Design)*
  - Implementierung *(Implementation)*
  - Tests *(Test)*
  - Auslieferung *(Deployment)*
- 3 unterstützende Arbeitsabläufe *(Supporting Workflows)*:
  - Konfigurations- und Änderungsmanagement *(Configuration and Change)*
  - Projektmanagement *(Project Management)*
  - Infrastruktur *(Environment)*

Phasen, Workflows und Iterations



Agile Softwareentwicklung

Agile Methoden – Gemeinsamkeiten

- Software wird nicht als Einheit entwickelt, sondern als Folge von Inkrementen, in denen jeweils neue Systemfunktionalität hinzugefügt wird
- Die Prozesse für Spezifikation, Entwicklung, Implementierung und Verifikation erfolgen verzahnt
- Inkrementelle Entwicklung, bei der die Inkremente klein sind und alle zwei bis drei Wochen neue Versionen erstellt und dem Kunden zur Verfügung gestellt werden
- Kontinuierliche Kommunikation mit dem Kunden

Manifest für Agile Softwareentwicklung

Wir erschließen bessere Wege, Software zu entwickeln,
indem wir es selbst tun und anderen dabei helfen.
Durch diese Tätigkeit haben wir diese Werte zu schätzen gelernt:

Individuen und Interaktionen mehr als Prozesse und Werkzeuge
Funktionierende Software mehr als umfassende Dokumentation
Zusammenarbeit mit dem Kunden mehr als Vertragsverhandlung
Reagieren auf Veränderung mehr als das Befolgen eines Plans

Das heißt, obwohl wir die Werte auf der rechten Seite wichtig finden,
schätzen wir die Werte auf der linken Seite höher ein.

http://www.agilemanifesto.org/iso/de/

Die 12 Prinzipien agiler Methoden

1. Kontinuierliche Auslieferung von Software
2. Offen für Anforderungsänderungen während der Erstellung
3. Kurze Iterationen
4. Enge Zusammenarbeit zwischen Entwicklung und Domänenexperten
5. Mitarbeitermotivation fördern
6. Direkte Kommunikation zwischen alle Projektbeteiligten
7. Lauffähige Software als Maß für den Projektfortschritt
8. „Nachhaltige“ Entwicklung
9. Technische Versiertheit und gutes Design
10. Einfachheit der Lösung
11. Teamarbeit und hohe Projektdynamik
12. Reflexion und Bewertung des Vorgehens

Agiler Entwicklungsprozess

Extreme Programming (XP)

Grundprinzipien

- Direkte Kommunikation zwischen allen Projektbeteiligten, auch mit dem Auftraggeber und Kunden
- Iteratives, inkrementelles Vorgehen
- Kleine Releases: jeweils »überschaubare« Menge von neuen Funktionalitäten
- Kontinuierliche Systemintegration
- **Testfunktionalität und -automatisierung**
- **Refactoring**
- **Programmieren in Zweier-Teams („Pair Programming“)**

Testfunktionalität

- Zur Erstellung des Anwendungssystems gehört gleichzeitig die Bereitstellung von Testfunktionen
- Häufig gekoppelt mit dem »Test-First« Ansatz (TDD):
  - vor dem Hinzufügen einer neuen Funktionalität wird zunächst eine Testfunktion geschrieben
  - diese legt das gewünschte Verhalten fest
  - danach erfolgt die Implementierung der geforderten Funktionalität
- Einbeziehung
  - des Kunden bei der Testerstellung
  - des Anwenders beim Testen
- Einbettung der Testfunktionen in eine automatisierte Testumgebung

Refactoring

- Die Umstrukturierung eines Programms ohne Änderung der Funktionalität ist wesentlicher Bestandteil beim Extreme Programming
- Designänderungen werden vor dem Einbau neuer Funktionen durchgeführt
- Ziele
  - Vereinfachung des Programmcodes: Zusammenlegen von Klassen, Generalisieren, Ändern von Vererbungsstrukturen, …
  - Steigerung der Lesbarkeit: Umbenennung von Variablen, Dokumentation anpassen, Einfügen von Schnittstellen, …
  - Verbesserte Wartbarkeit des Systems
- Literatur
  - Martin Fowler, Refactoring – Improving the Design of Existing Code. Addison-Wesley, 1999

Programmieren in kleinen Teams

- Enge Zusammenarbeit beim Programmieren
  - Programmcode wird von zwei Personen am Rechner geschrieben (»Pair Programming«)
  - einer tippt, der andere achtet auf Fehler, behält den Überblick, usw.
  - die Programmierer/-innen unterhalten sich dabei
  - bei Bedarf werden die Rollen gewechselt
- »Collective Ownership«
  - jeder ist verantwortlich für das ganze System
  - die Konstellation von Paaren wechselt oft
  - ersetzt formellere Code-Review Techniken

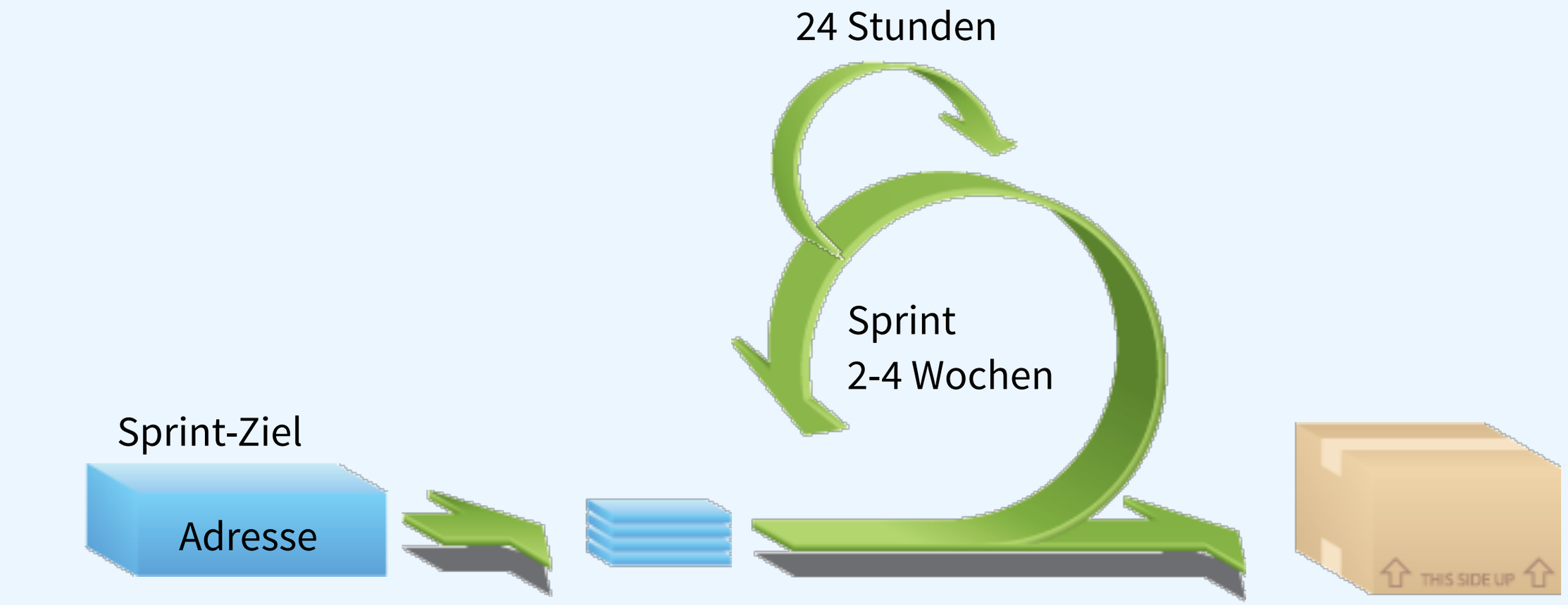
Extreme Programming: Kritische Aspekte

- Architektur:
  - die Architektur soll aus dem »Nichts« entstehen
  - es können Design-Entscheidungen getroffen werden, die in der gegenwärtigen Situation sinnvoll sind; aus Sicht des gesamten Systems jedoch nicht optimal sind
  - das Refactoring solcher verbesserungsbedürftiger Strukturen kann letztendlich teuer werden
- Enge Einbeziehung des Kunden nicht immer möglich:
  - der Repräsentant des Kunden sollte dem Projekt vor Ort zur Verfügung stehen
  - kennt die fachliche Domäne und versteht die Sprache des Projektteams

Agiler Entwicklungsprozess

Scrum

Scrum Prozess - Warenkorb



Quelle: Mountain Goat Software, LLC

Scrum - Terminologie (1)

- Produkt-Vision
  - enthält die grundlegende Idee
  - wird vom Produkt-Owner konzipiert
- Produkt-Backlog
  - Liste der Backlog-Items, d.h. Produkt-Funktionalitäten
  - wird vom Produkt-Owner entweder allein oder mit Hilfe der Projekt-Mitglieder erarbeitet
  - Festlegung der Backlog-Items nach Wichtigkeit, d.h. dem zu erwartenden finanziellen Gewinn der jeweiligen Funktionalität

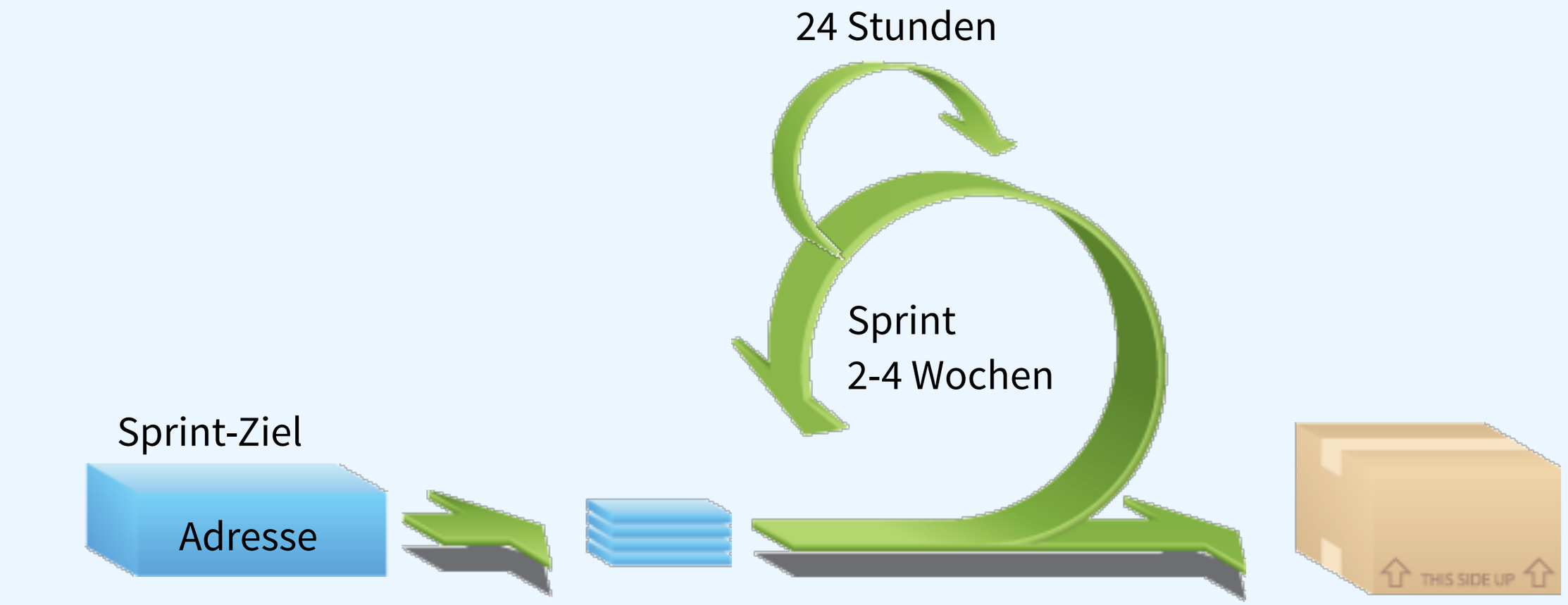
Scrum - Terminologie (2)

- Scrum-Team
  - besteht aus den Personen, die notwendig sind, um die Backlog-Items in »usable Software« zu verwandeln (5-9 Personen)
  - schätzt den Umfang der Backlog-Items und des Produkt-Backlogs (→ Grundlage zur Genehmigung durch den Auftraggeber)
- Sprint Planning Meeting 1
  - Produkt-Owner, Team, Management und Anwender legen die Ziele des Sprints gemeinsam fest
  - Identifikation der Backlog-Items, die das Team in diesem Sprint liefern wird
- Sprint Planning Meeting 2
  - Festlegung der Umsetzungsstrategie, vergleichbar einem Design-Workshop
  - Architekturdefinition

Scrum - Terminologie (3)

- Sprint
  - klar abgegrenzte zeitliche Intervalle
  - am Ende eines „*n* day Sprints“ muss das Team Software liefern, die in einem Zustand ist, die ausgeliefert werden könnte (usable Software)
- Tägliche Abstimmung beim „Daily Scrum“ Meeting:
  - » Koordination und Feedback in 15 Minuten
  - » Was habe ich seit dem letzten Daily Scrum erreicht?
  - » Was will ich bis zum nächsten Daily Scrum erreichen?
  - » Welche Hindernisse sind mir dabei im Weg?

Scrum Prozess (reviewed)



Quelle: Mountain Goat Software, LLC

Zusammenfassung

Auswahl eines geeigneten Prozessmodells

- Zur Beurteilung der prinzipiellen Eignung eines Prozessmodells für ein Entwicklungsvorhaben sollten verschiedene Faktoren betrachtet werden
- Eignung …
  - für unklare, sich ändernde (Kunden-)Anforderungen, neue Marktsegmente bzw. Anwendungsgebiete
  - für technisches Neuland: neue Systemarchitektur, neue Techniken und Methoden
  - zur Risikominimierung bzgl. Machbarkeit, Kosten, Termine, kritische Performance-Anforderungen, …
  - für große, komplexe Systeme
  - zur adäquaten Kommunikation mit dem Kunden und Auftraggeber
  - zur Fortschrittskontrolle für das Management

Danke

Prof. Dr. Stefan Betermieux | Fakultät Informatik | Hochschule Furtwangen