Kapitel 12.
Softwareentwicklungsprozessmodelle

Stand: 26.01.2020

Folie 37: Text
Folie 38: Farben
Folie 39: Eingeblendet
Folie 42: Neu eingefügt

Einordnung

- Bisher: **Womit** arbeiten wir?
  - UML, OO-Modellierungsprinzipien, …
- Bisher: **Was** tun wir? → Aktivitäten
  - Anforderungen erfassen, entwerfen, implementieren, testen, ...
  - Versionsverwaltung, Qualitätssicherung, ...
- Nun: **Wie** tun wir es?
  - Wie passt das alles zusammen im Rahmen eines Projektes?
  - Was gehört noch zu einem Projekt außer dem Obigen?
  - Wie organisieren wir ein Projekt?
  - Was ist ein Softwareentwicklungsprozessmodell?
  - Was ist „Agile Softwareentwicklung“?

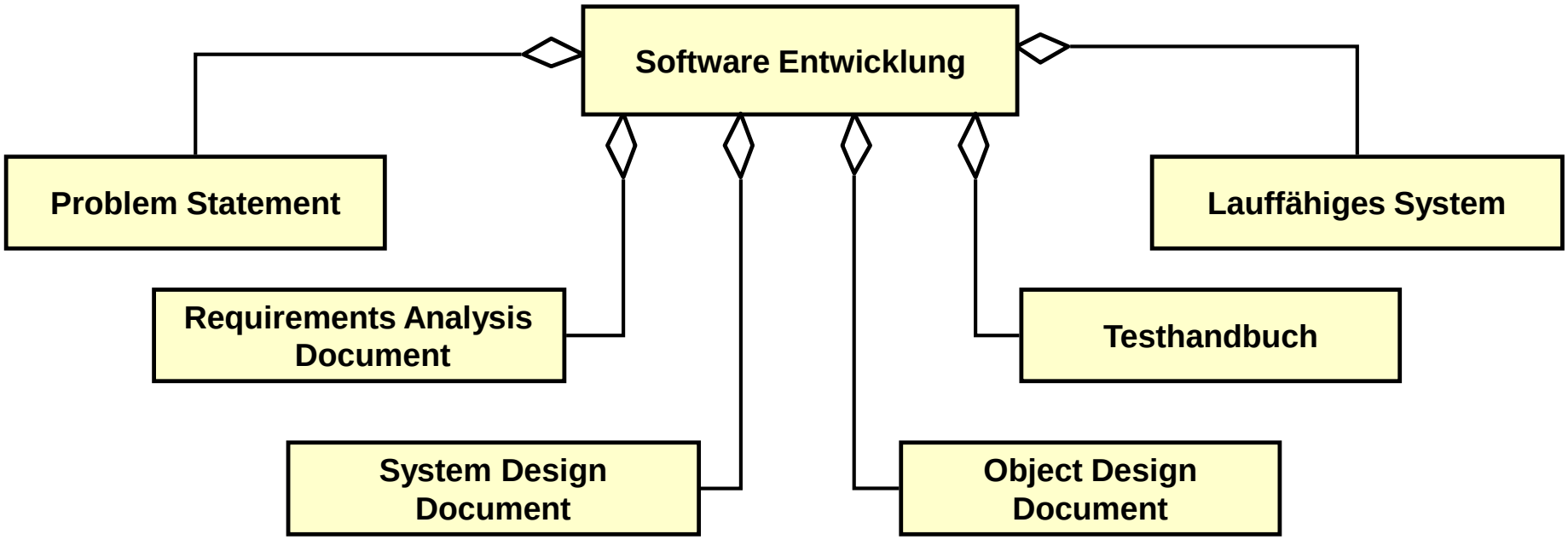
Überblick

- Dieser Foliensatz (Kapitel 12): Software Prozessmodelle
  - Das Wasserfallmodell und seine Probleme
  - Iterative und inkrementelle Prozessmodelle
- Nächste Foliensatz (Kapitel 13): Agile Prozessmodelle
  - Extreme Programming, Scrum, Kanban

Typische Fragen zum Software-Lebenszyklus

- Welche **Aktivitäten** soll ich für das Softwareprojekt auswählen?
- Was sind die **Produkte** der Aktivitäten?
- Welche **Rollen** sind an meinem Projekt notwendig?
  - … und wer nimmt sie ein?
- Was sind die **Abhängigkeiten** zwischen den Aktivitäten?
  - Welche **Aktivitäten** hängen von welchen **Produkten** ab?
  - Hängt das Systemdesign von der Analyse ab?
  - Hängt die Analyse vom Design ab?
- Wie soll ich die Aktivitäten **planen**?
  - Soll die komplette Analyse dem Design vorangehen?
  - Können Analyse und Design parallel ablaufen?
  - Sollten sie iterativ ablaufen?

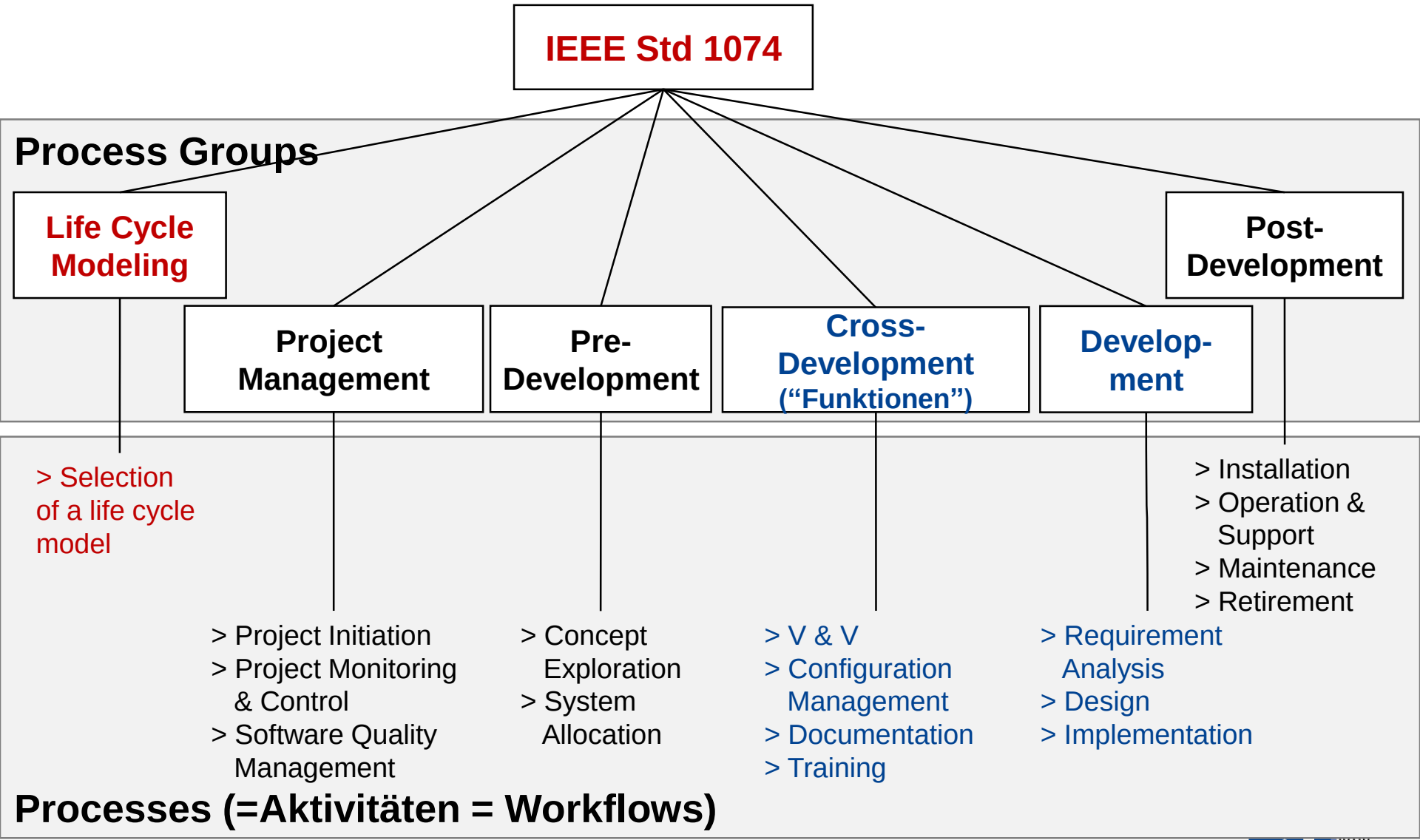
Produkte


Software Entwicklung
Problem Statement
Lauffähiges System
Requirements Analysis Document
Testhandbuch
System Design Document
Object Design Document

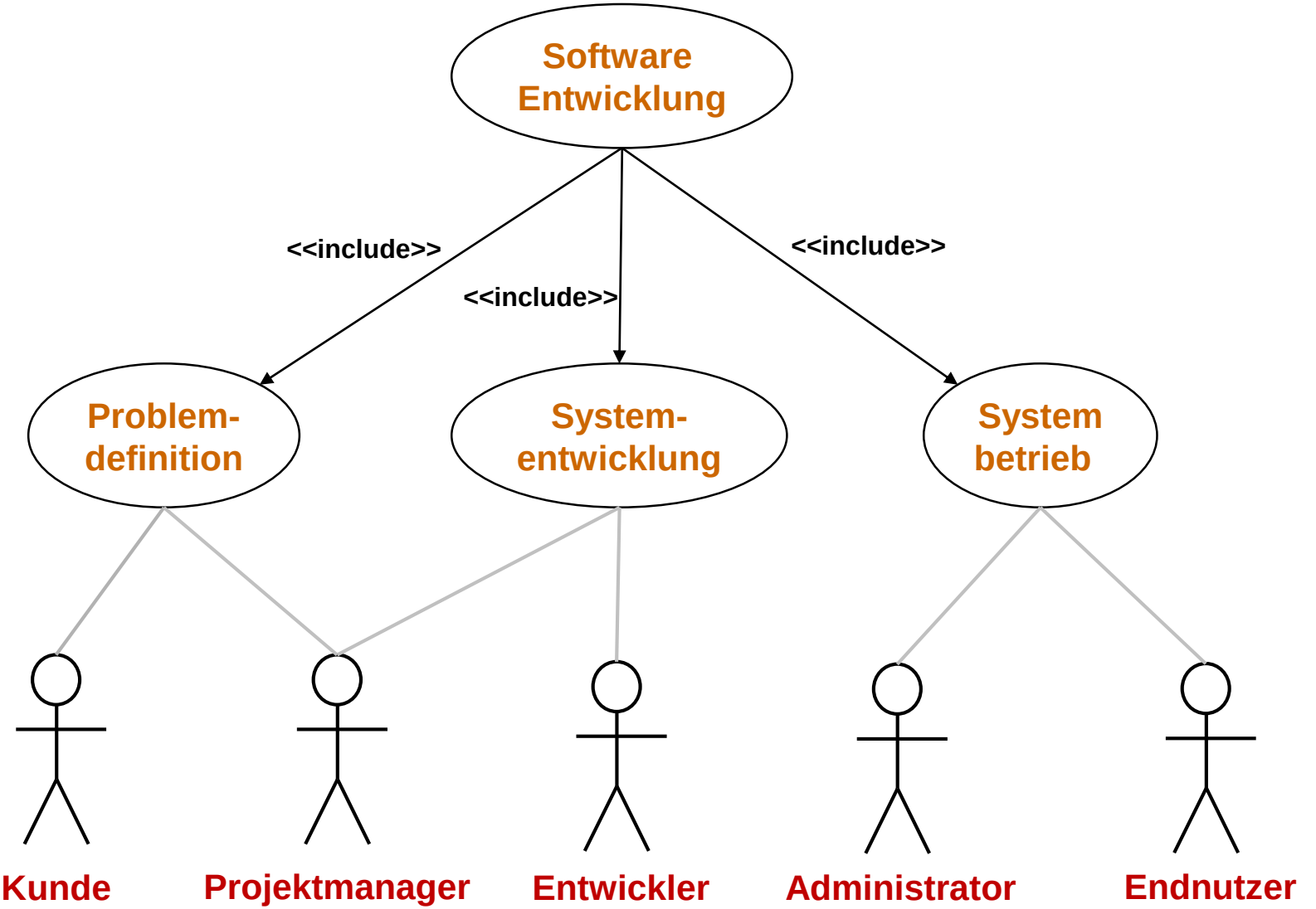
Aktivitäten (Beispiele)

Anforderungsanalyse	Was ist das Problem?
Systementwurf	Was ist die Lösung?
Programmentwurf	Welche Mechanismen liefern die beste Lösung?
Implementierung	Wie setzt sich die Lösung zusammen?
Tests	Ist das Problem gelöst?
Auslieferung	Kann der Benutzer die Lösung verwenden?
Wartung	Sind Verbesserungen notwendig?

IEEE Standard 1074: Standard für den Software-Lebenszyklus



Rollen und Teilprozesse



Prozessgruppen, Prozesse, Aktivitäten und Aufgaben

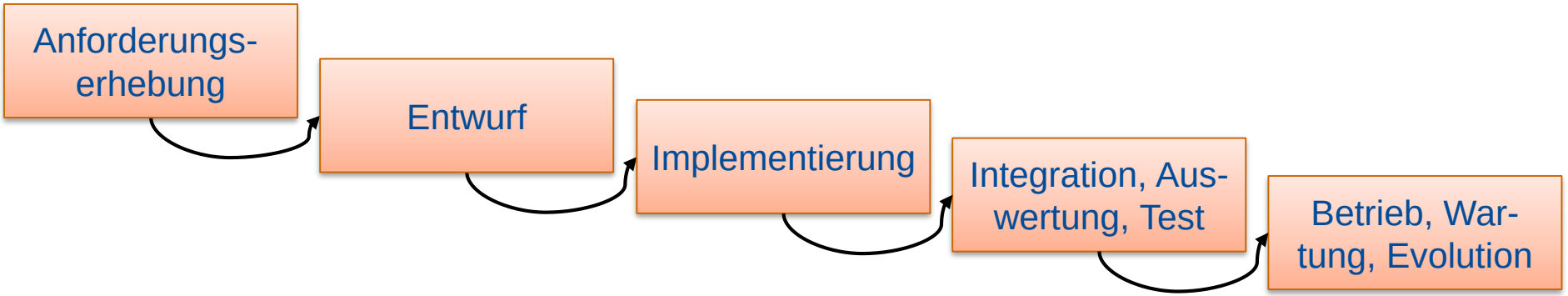
- Prozessgruppe
  - Besteht aus Prozessen
  - Beispiel: “Entwicklungs-Prozessgruppe”
- Prozess
  - Besteht aus Aktivitäten
  - Beispiel: “Entwurfsprozess”
- Aktivität
  - Besteht aus Unteraktivitäten und Aufgaben
  - Beispiel : “Datenbank-Design-Aktivität”
- Aufgabe (“Task”)
  - Atomare Einheit der Aufgabenzuteilung
  - Beispiel : “Bestimme die Tauglichkeit von Oracle DBMS”

Beispiel

- Der Entwurfsprozess ist Teil der Entwicklung
- Der Entwurfsprozess besteht aus den folgenden Aktivitäten
  - Architekturentwurf
  - Datenbankentwurf (falls nötig)
  - Schnittstellenentwurf
  - Falls nötig Auswahl oder Entwicklung von Algorithmen
  - Detailentwurf (= Objektentwurf)
- Die Datenbankentwurf-Aktivität umfasst z.B. folgende Aufgaben
  - Bewertung relationaler Datenbanken
  - Bewertung objektorientierter Datenbanken
  - Kaufempfehlung
  - ....

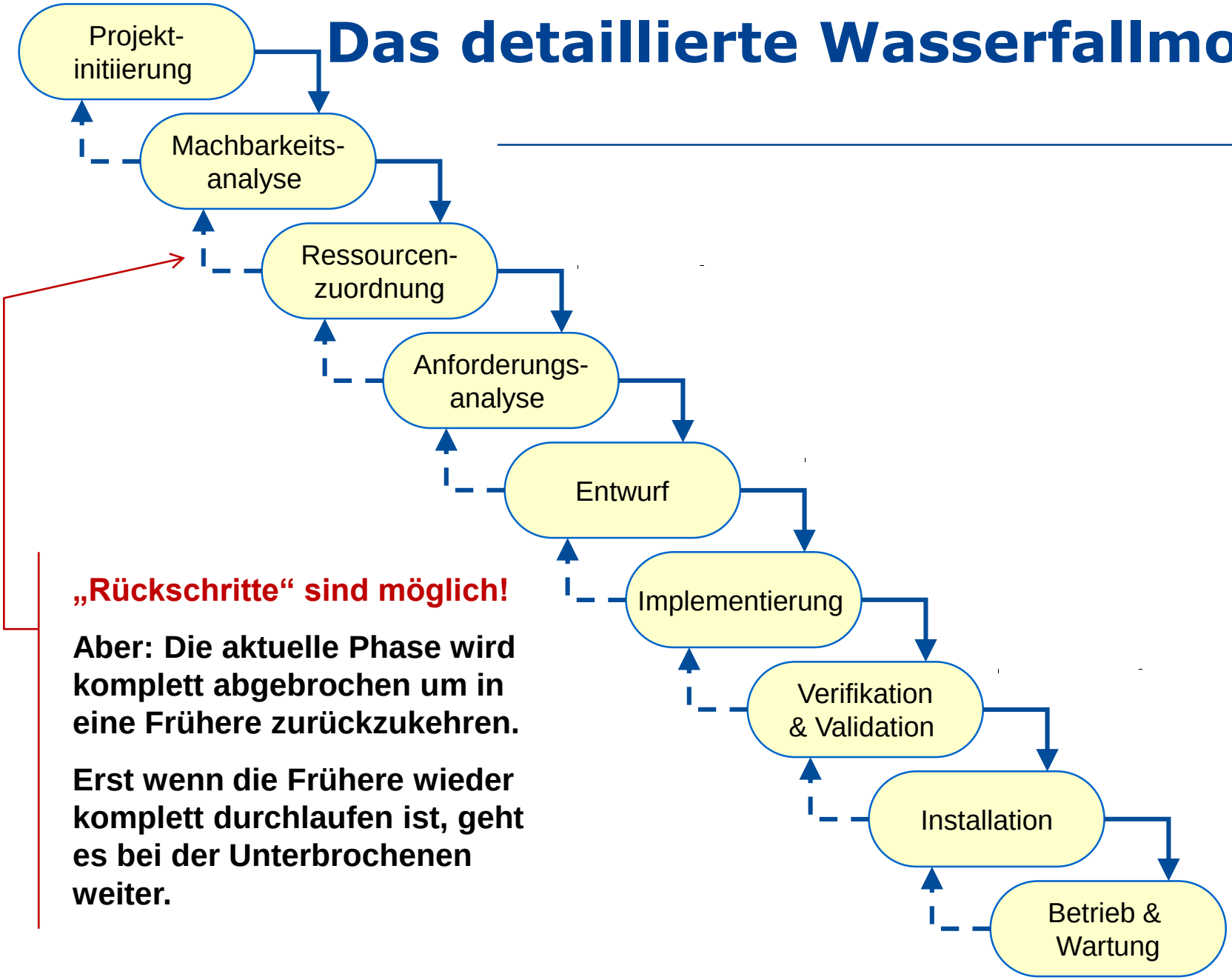

Wasserfallmodell

Wasserfallmodell (1/4)



- Prozess der die sequentielle Ausführung aller Aktivitäten einer Kategorie vor denen der nächsten Kategorie vorschreibt
  - Alle Anforderungen werden erhoben bevor mit dem Entwurf begonnen wird
  - Zwischenergebnisse: Dokumente (Z.B. “Pflichtenheft”).
- Man kann auch zur vorherigen Aktivitätskategorie zurückkehren, wenn man Änderungsbedarf feststellt
  - Das impliziert aber, dass man die Aktivitäten der aktuellen Kategorie ruhen lässt, bis alle vorherigen wieder vollständig aktualisiert sind → kostspielig
  - Vorhandene Probleme werden aufgeschoben, um sie “herum” programmiert

Das detaillierte Wasserfallmodell



„Rückschritte“ sind möglich!

Aber: Die aktuelle Phase wird komplett abgebrochen um in eine Frühere zurückzukehren.

Erst wenn die Frühere wieder komplett durchlaufen ist, geht es bei der Unterbrochenen weiter.

Für und Wider des Wasserfallmodells

- Manager lieben Wasserfallmodelle
  - Nette Meilensteine
  - Kein Rückblick nötig (lineares System), jederzeit nur eine Aktivität
  - Fortschritt leicht zu prüfen : 90% implementiert

Aber, ...

- Softwareentwicklung ist iterativ
  - Während des Designs werden Probleme mit den Anforderungen festgestellt
  - Während der Implementierung werden Design- und Anforderungsprobleme festgestellt
  - Während des Testens werden Implementierungs-, Design-, und Anforderungsfehler gefunden
  - → Spiralmodell
- Systementwicklung ist nicht-linear
  - → Problem-orientiertes Modell


Spiralmodell

Iterativ oder Inkrementell?
Buch-Schreiben-Analogie

- Inkrementelle Entwicklung bedeutet:
  - Ich schreibe Kapitel 1 komplett
  - Ich schreibe Kapitel 2 komplett
  - Ich schreibe Kapitel 3 komplett
  - … bis alle Teile geschrieben sind

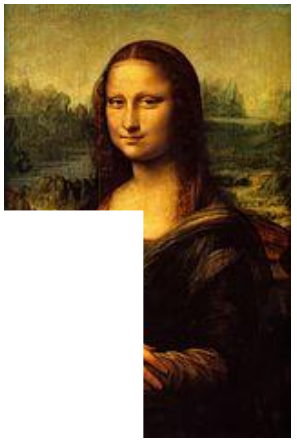
- Iterative Entwicklung bedeutet:
  - Ich schreib alles auf was mir in den Sinn kommt
  - Ich geh es durch und lösch den ganzen Müll, vertiefe das Relevante, verbessere die Struktur
  - Ich geh es wieder durch und erkenne neue Verbesserungsmöglichkeiten
  - Ich gehe es wieder durch …
  - … bis es gut genug ist

from: http://c2.com/cgi/wiki?IterativeVsIncremental

Iterativ oder Inkrementell?
Mona-Lisa-Analogie

Inkrementelle Entwicklung






Iterative Entwicklung




Idee übernommen von: http://agileproductdesign.com/blog/dont_know_what_i_want.html

Iterative Entwicklungsprozesse

- Akzeptieren, dass sich Anforderungen ändern
  - 40% der finalen Anforderungen kamen erst nach der Analysephase
    - ⇨ Ergebnis von Caspers Jones' Auswertung von 8000 Projekten
- Begünstigen vorzeitige Risikoverringerung
  - Aufspalten des Systems in Mini-Projekte
  - Schwerpunkt zuerst auf die riskanten Elemente legen
- Fördern die gemeinsame Beteiligung aller Teilnehmer
- Ermöglichen die Anpassung des Prozesses nach jeder Iteration
  - Was haben wir aus dieser Iteration gelernt?

Das Spiralmodell (Boehm) befasst sich mit Iteration

- Identifiziere die Risiken
- Gib den Risiken Prioritäten
- Entwickle und teste eine Reihe von Prototypen für die einzelnen Risiken
- … in der Reihenfolge fallenden Risikos bzw. fallender Priorität
- Nutze das Wasserfallmodell zur Entwicklung jedes Prototyps (“Zyklus”)
- Wenn ein Risiko erfolgreich beseitigt wurde, bewerte das Ergebnis des Zyklus und plane die nächste Runde
- Wenn ein bestimmtes Risiko nicht beseitigt werden kann, beende das Projekt sofort

Spiralmodell (Boehm 1988)

- Basiert auf den Aktivitäten des Wasserfallmodells
- Fügt mehrere Aktivitäts (-Zyklen) hinzu
  - Ziele definieren
  - Risikoabschätzung und -reduzierung
  - Entwicklung und Bewertung
  - Planen der nächsten Phasen
- Idee
  - Prozess ist wie eine Spirale angeordnet
  - Risikien werden explizit abgeschätzt und während des Prozesses behoben
- → Details in Schaubild auf nächster Folie

Example Spiral Model Project

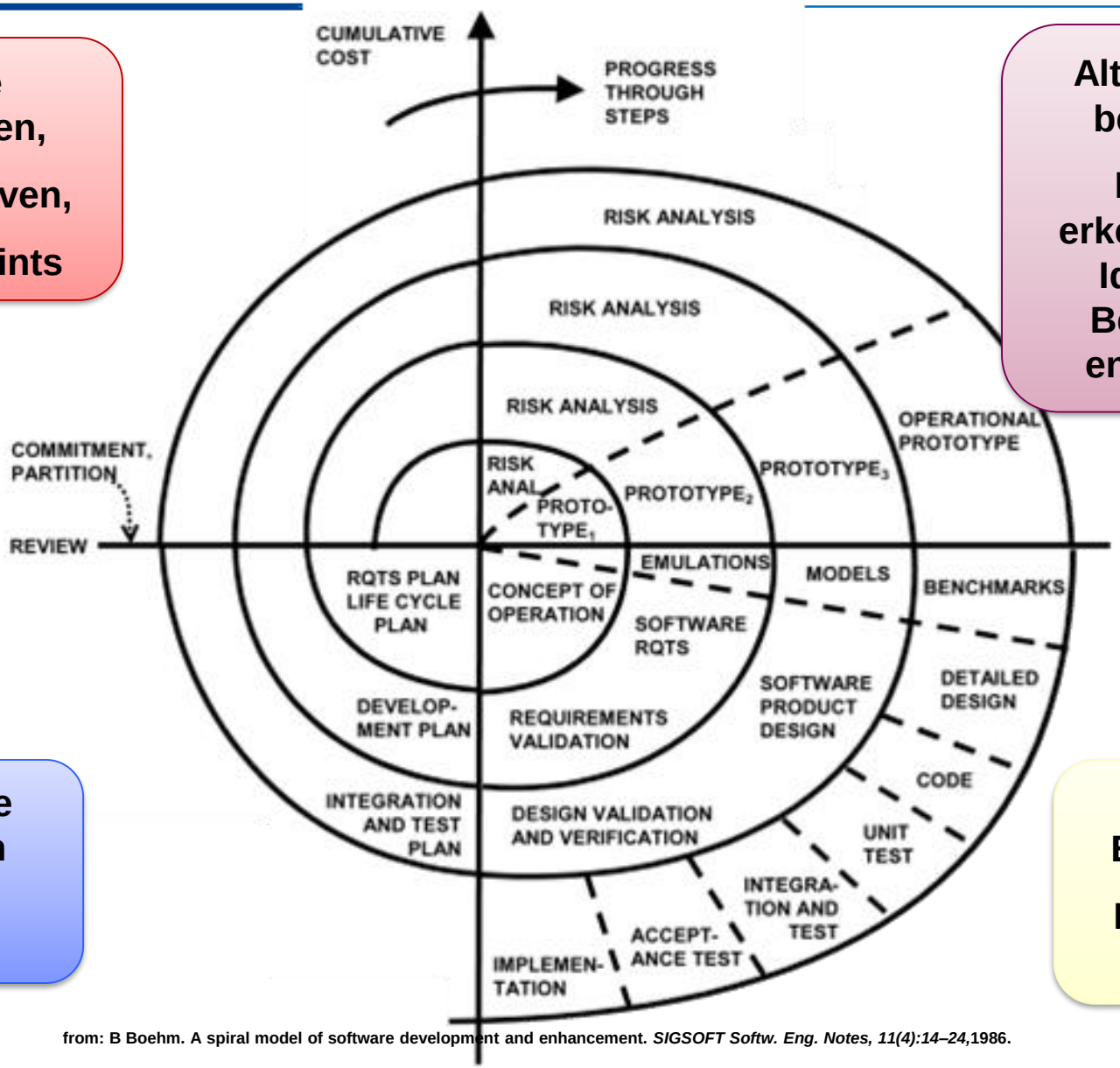
Ziele festlegen,

Alternativen,

Constraints

Alternativen bewerten,

Risiken erkennen und Ideen zur Behebung entwickeln



Nächste Phasen planen

Software Entwickeln,

Ergebnisse bewerten

from: B Boehm. A spiral model of software development and enhancement. *SIGSOFT Softw. Eng. Notes, 11(4):14–24,*1986.

Aktivitäten (“Runden”) in Boehm’s Spiralmodell

- Die folgenden Aktivitäten verteilen sich über die Zyklen der Spirale
  - Rahmenbedingungen
  - Anforderungen
  - Systemdesign
  - Detailiertes Design
  - Implementierung
  - Modultest
  - Integration und Test
  - Akzeptanztest
- Frühere Aktivitäten geschehen in den inneren/initialen Zyklen, spätere in den äußeren Zyklen
  - Siehe Spirale auf der Seite zuvor
- Gehe in jedem Zyklus diese Schritte
  - Bestimme Ziele, Alternativen, Nebenbedingungen
  - Bewerte Alternativen, identifiziere und löse Risiken
  - Entwickle und verifiziere den Prototyp
  - Plane den nächsten Zyklus

Prototypen beim Spiralmodell

- Illustrativer Prototyp
  - Im Dialog mit Kunden Arbeitsabläufe und GUI die sie unterstützt festlegen
    - ⇨ „Interaktionsdesign“
  - „Implementiere“ das Userinterface als „Mock-Up“
    - ⇨ UI-Builder-Tools nur bedingt geeignet, da sie Kreativität einengen auf das was es schon gibt
    - ⇨ Besser: Papier, Stifte, Schere, Kleber, ...
- Funktionaler Prototyp
  - Implementiere ein lauffähiges System mit minimaler Funktionalität
  - Dann füge Funktionalität hinzu
  - Das jeweilige Risiko bestimmt die Reihenfolge
- Explorations-Prototyp
  - Implementiere ein Teil-System, um mehr über die Anforderungen zu lernen
  - Gut für Brüche im Denkmuster

Prototyping (fortgesetzt)

- Revolutionäres Prototyping
  - Ermittle die Praxis beim Kunden mit einer Wegwerf-Version um die Anforderungen richtig zu bestimmen, dann baue das ganze System
    - ⇨ Nachteil: Benutzer müssen einsehen, dass Funktionen des Prototypen teuer in der Implementierung sind
    - ⇨ Benutzer kann enttäuscht sein, weil ein Teil der Funktionalität des Prototyps in der späteren Implementierung nicht machbar ist

- Evolutionäres Prototyping
  - Der Prototyp ist Basis für die Implementierung des finalen Systems
  - Vorteil: Kurze Fertigstellungszeit
  - Nachteil: Kann nur benutzt werden, wenn das Zielsystem in der Sprache des Prototypen konstruiert werden kann

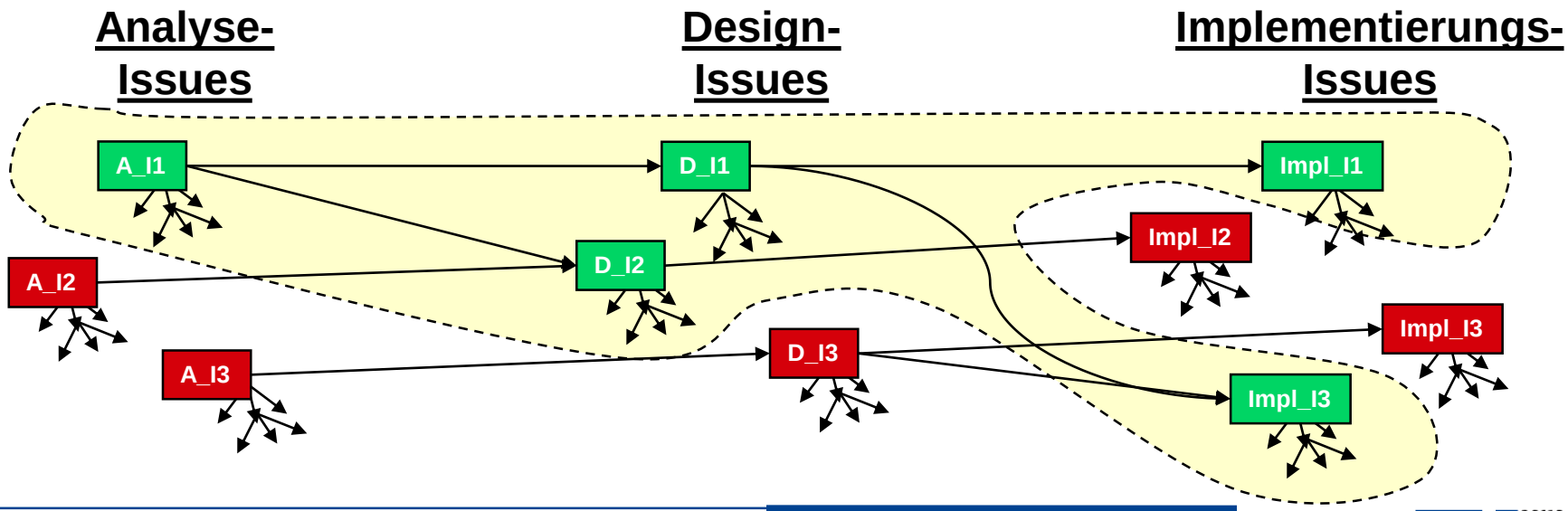
Die Grenzen des Wasserfall- und Spiralmodells

- Keines von beiden befasst sich mit häufigen Änderungen
  - Das Wasserfall unterstellt, dass nach einer Phase alle Probleme in dieser abgeschlossen sind und nicht wieder geöffnet werden können
  - Das Spiralmodell kann mit Änderungen zwischen den Phasen umgehen, aber nicht mit Änderungen während einer Phase

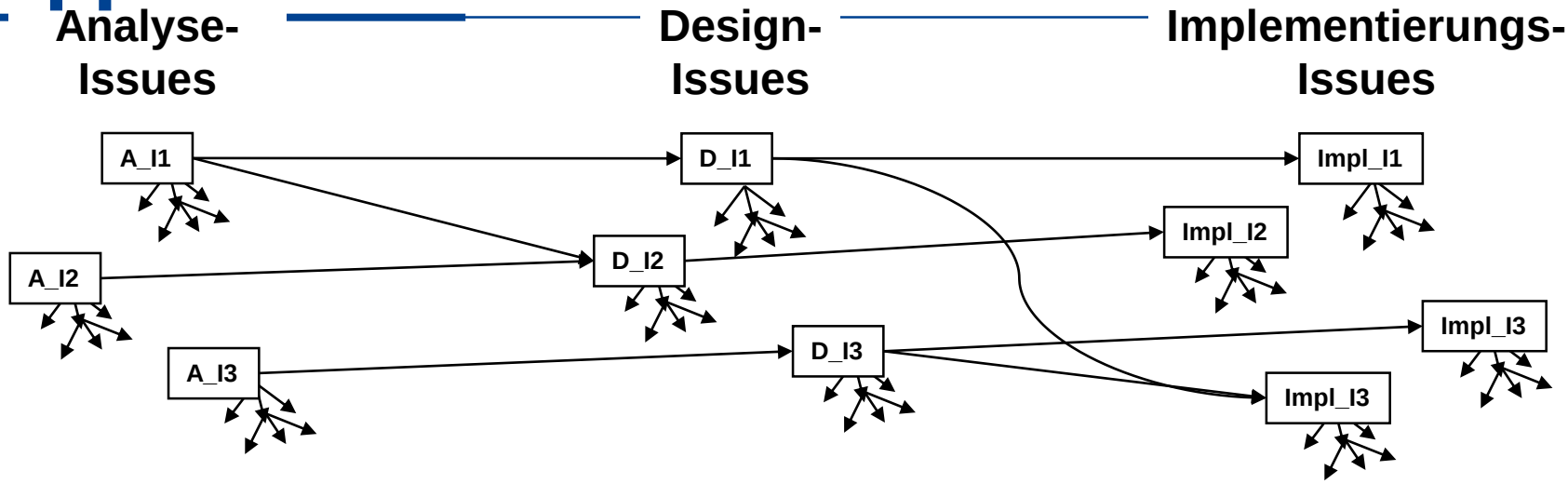
- Was tun, wenn Änderungen häufiger erfolgen? (“Das einzig konstante ist die Änderung”)

Issue-Based Development

- Ein System wird durch eine Sammlung von Problemen beschrieben
  - Probleme sind offen A_I2 oder geschlossen A_I2
  - Geschlossene Probleme haben eine Lösung
  - Geschlossene Probleme können wieder geöffnet werden (Iteration!)
- Probleme haben Abhängigkeiten
- Die geschlossenen Probleme sind die Basis des Systemmodells

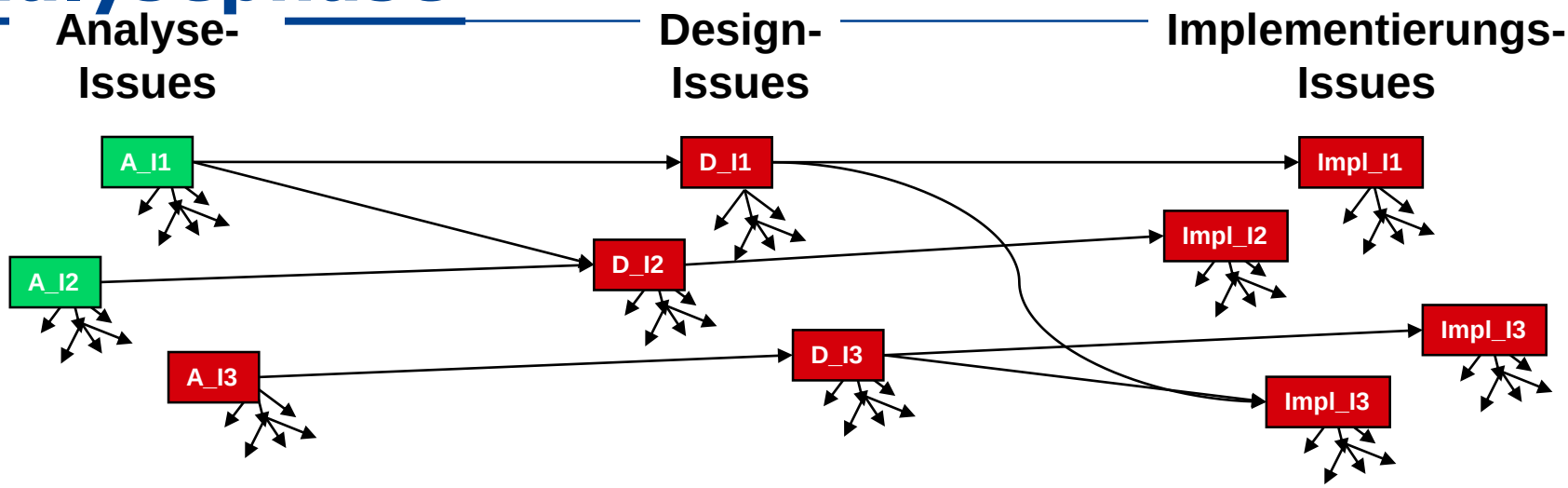


Beispiel: Projekt mit nach Aktivitäten gruppierten Issues



- Legende (auf folgenden Seiten)
  - A_I2 Rot = „offene issues“(noch zu bearbeiten)
  - A_I2 Grün = „geschlossene issues“(erledigt)

Vorgehen im Wasserfallmodell: Analysephase

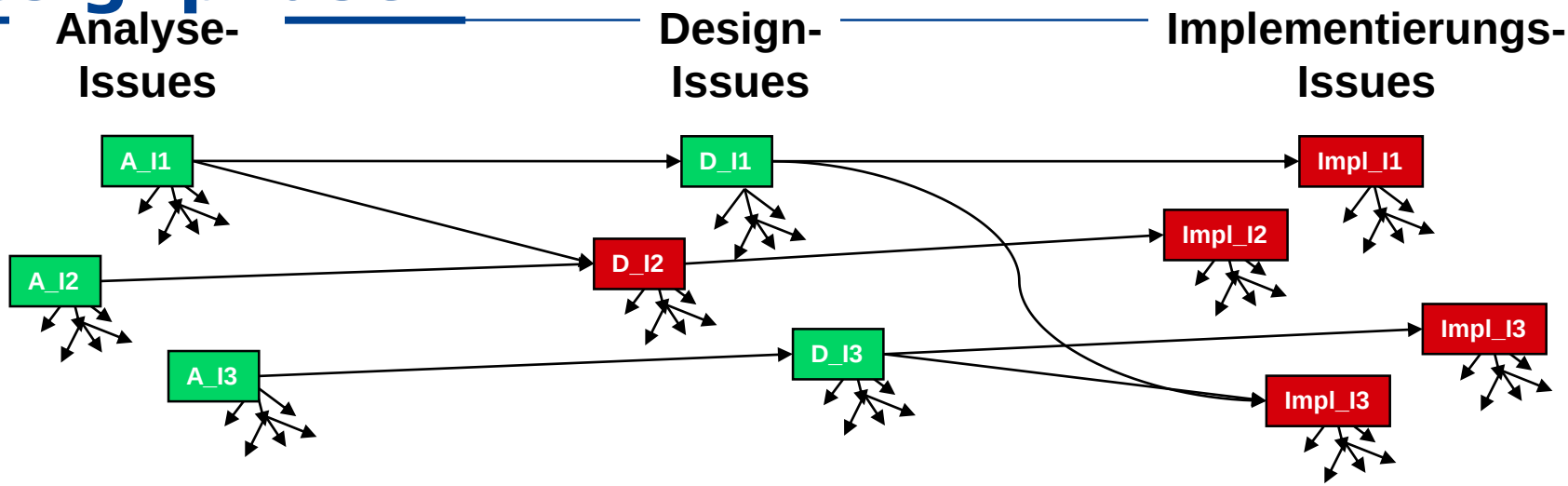


100-0% Offene Analyse-Issues

100% Offene Design-Issues

100% Offene Implem.-Issues

Vorgehen im Wasserfallmodell: Designphase

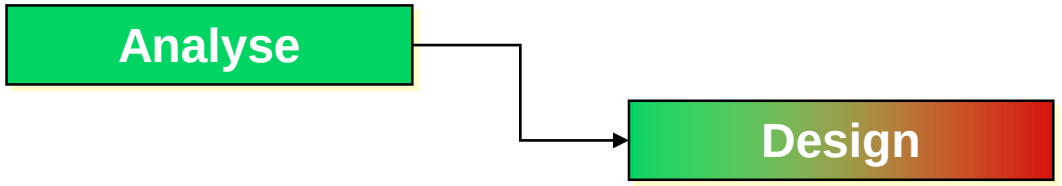


0% Offene Analyse-Issues

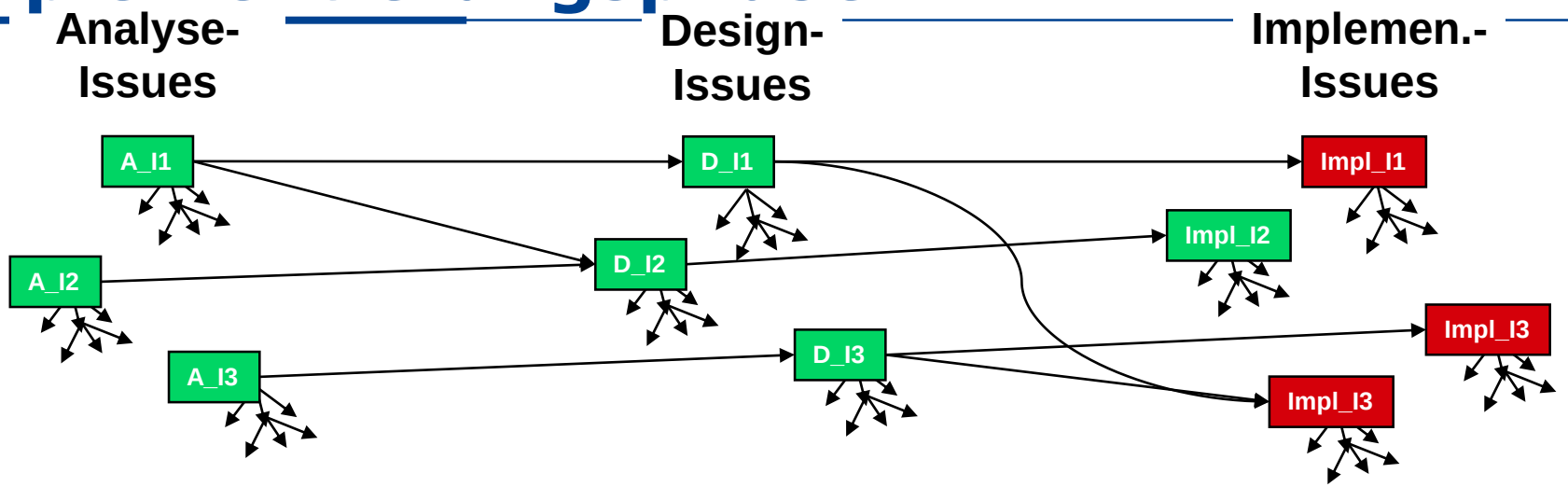
100-0% Offene Design-Issues

100% Offene Implem.-Issues

Wasserfall



Vorgehen im Wasserfallmodell: Implementierungsphase

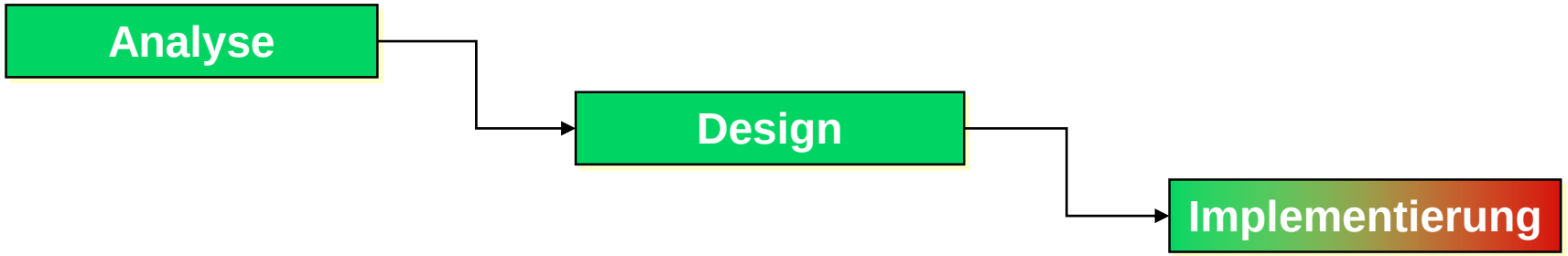


0% Offene Analyse-Issues

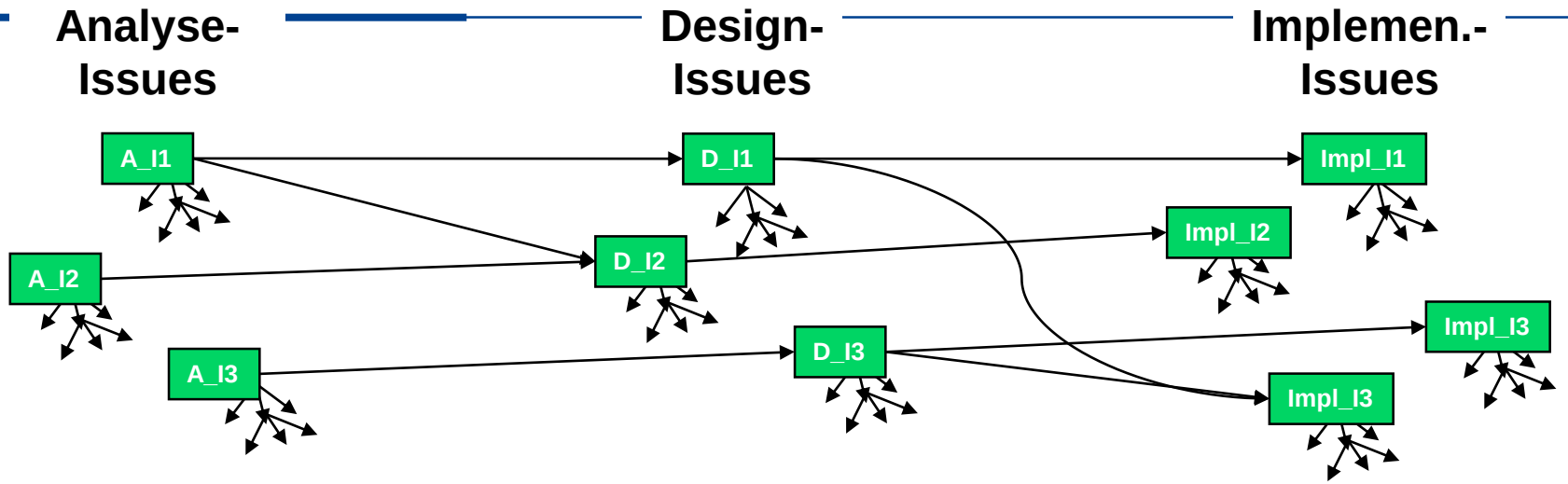
0% Offene Design-Issues

100-0% Offene Implem.-Issues

Wasserfall



Wasserfallmodell: Projekt fertig!

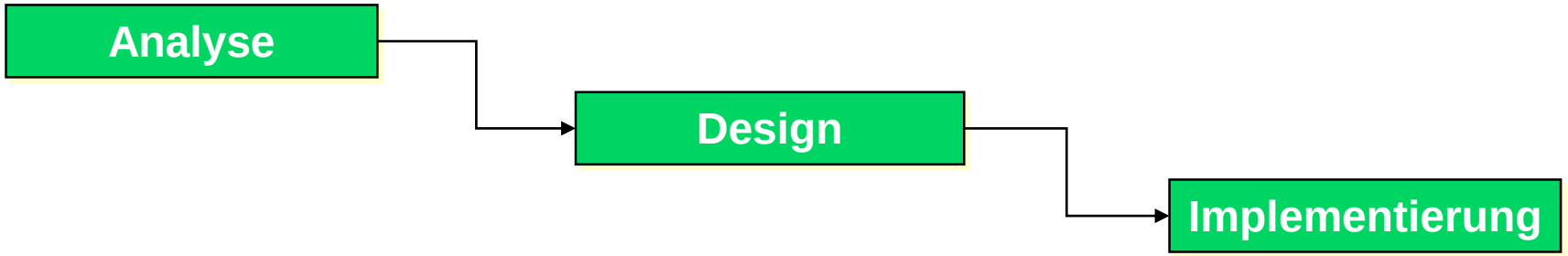


0% Offene Analyse-Issues

0% Offene Design-Issues

0% Offene Implem.-Issues

Wasserfall

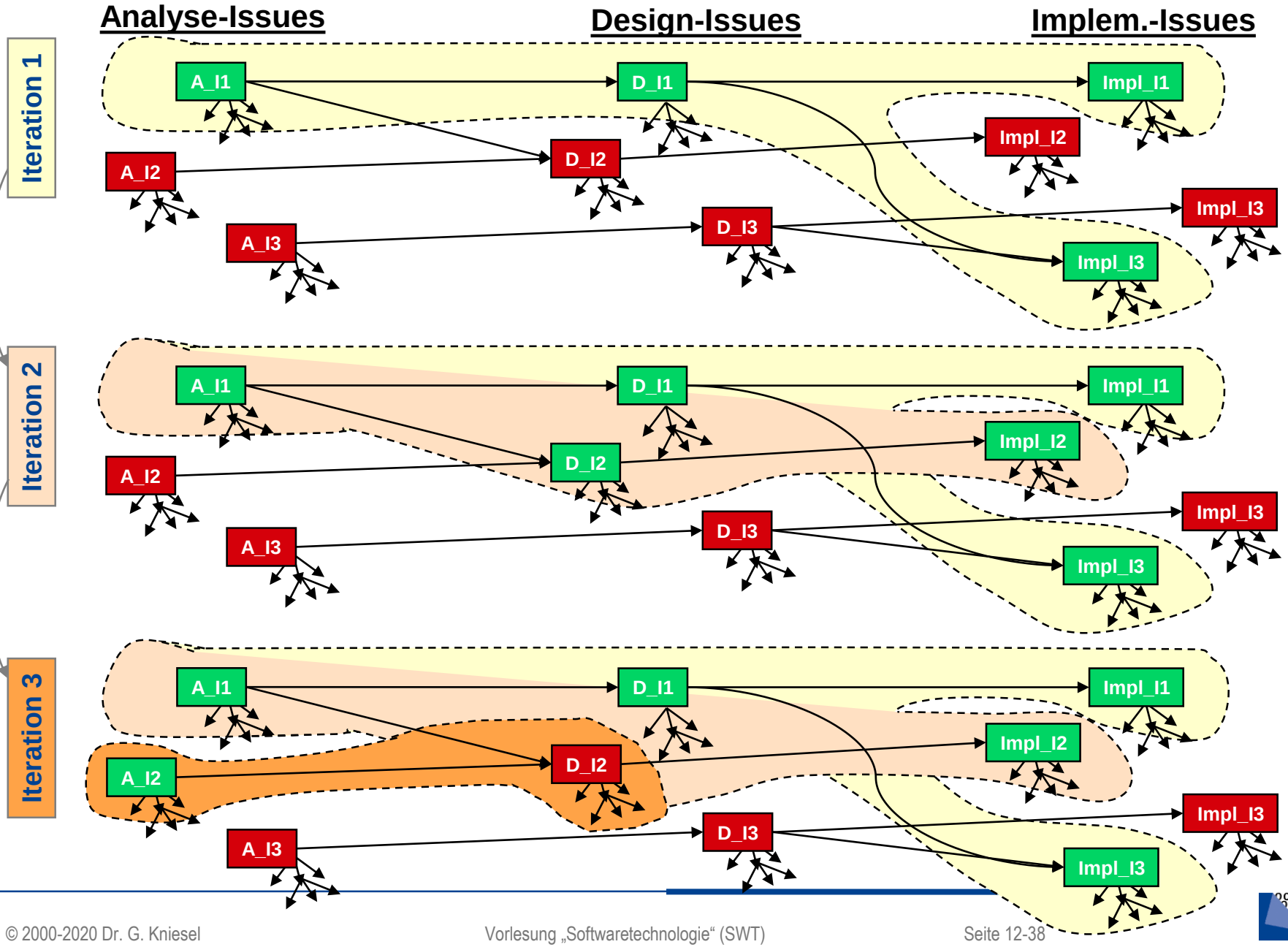


← Bisher: Issue-basierte Illustration eines Wasserfalls (in dem Aktivitäten in Phasen ablaufen)

Nun: „Richtiges" Issue-Based Development →

- Verschiedene Aktivitäten werden entsprechend der Abhängigkeiten hintereinander ausgeführt (nicht wie beim Wasserfallmodell als Phasen die nur aus jeweils einer Aktivität bestehen)
- Iteration 3 zeigt eine Momentaufnahme zu der wegen der Einbeziehung von Anforderungs-Issue A_I2, der Entwurfs-Issue D_I2 neu eröffnet wurde und dieser sich noch in Bearbietung befindet, so dass noch nicht klar ist, ob es weitere, davon abhängige Issues, z.B. Impl_I2 betroffen sind und ebenfalls neu eröffnet werden müssen.

„Issue-Based Development“



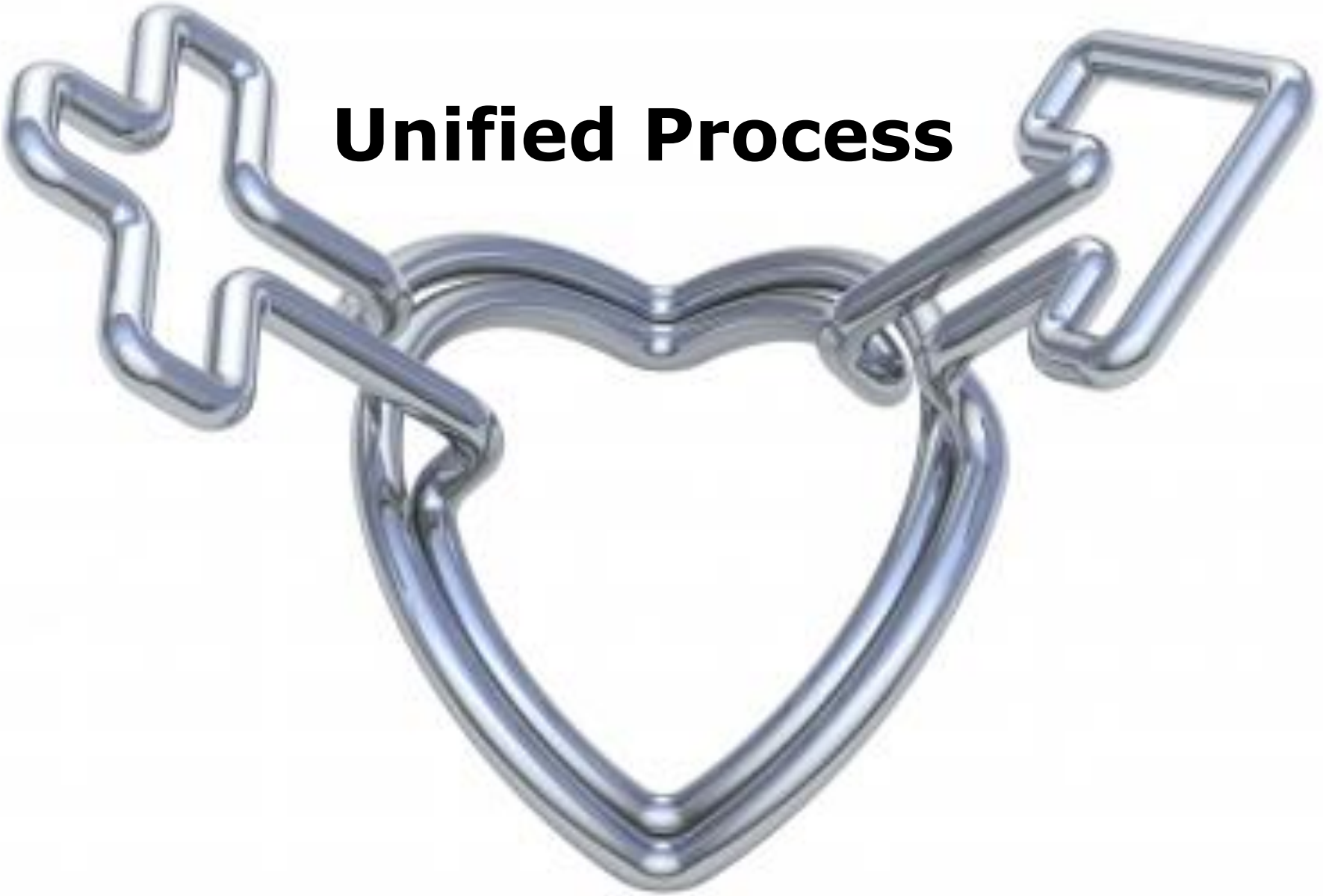
Erläuterungen zur vorherigen Folie

- System entsteht Schrittweise durch Iterationen.
  - Iterationen fassen Issues verschiedener Aktivitäten zusammen.
  - Sie sind problem- und abhängigkeitsorientiert: Alle zusammengefassten Issues sind von einigen „top-level“ Anforderungs-Issues abhängig, deren Umsetzung für den Projektfortschritt einen erkennbaren Mehrwert bietet.
- Jede Iteration produziert einen wohldefinierten neuen Zustand
  - Möglichst einen, der ein funktionierendes Gesamtsystem ergibt
  - ... zu dem Kunden oder Endbenutzer uns Feedback geben können
- Interpretation des Beispiels
  - Iteration 1 implementiert eine Basisversion der für Issue A_I1 gewählten Lösung.
    - ⇨ Dies umfasst die Identifikation, Entscheidung und Umsetzung des Design-Issues D_I1 und der Implementierungs-Issues Impl_1, Impl_3
    - ⇨ Der identifizierte Design-Issue D_I2 wurde zurückgestellt.
  - Iteration 2 ergänzt das System um die Lösung und Implementierung von issue D_I2
  - Iteration 3 widmet sich dem zurückgestellten Analyse-Issue A_I2
    - ⇨ Dabei wird eine Abhängigkeit zu dem schon geschlossenen Issue D_I2 erkannt. Dieser wird wieder geöffnet, neu abgewogen, entschieden, ...

Wann welches Modell verwenden?

- Häufigkeit von Änderungen und Software-Lebenszyklus
  - PT = Projektzeit (project time)
  - MTBC = mittlere Zeit zwischen Änderungen (mean time between changes)
- Änderungen sehr selten (MTBC >> PT):
  - Wasserfallmodell
  - Alle Probleme einer Phase sind vor der nächsten geschlossen
- Änderungen manchmal (MTBC ≅ PT):
  - Boehm's Spiralmodell
  - Änderung während einer Phase kann zur Iteration einer früheren Phase oder der Beendigung des Projektes führen
- Ständige Änderungen (MTBC << PT):
  - Issue-based Development (Concurrent Development Model)
  - Phasen sind nie beendet, laufen alle parallel
    - ⇨ Entscheidung über den Abschluss eines Problems beim Management
    - ⇨ Menge abgeschlossener Probleme ist Basis für das zu entwickelnde System

Unified Process

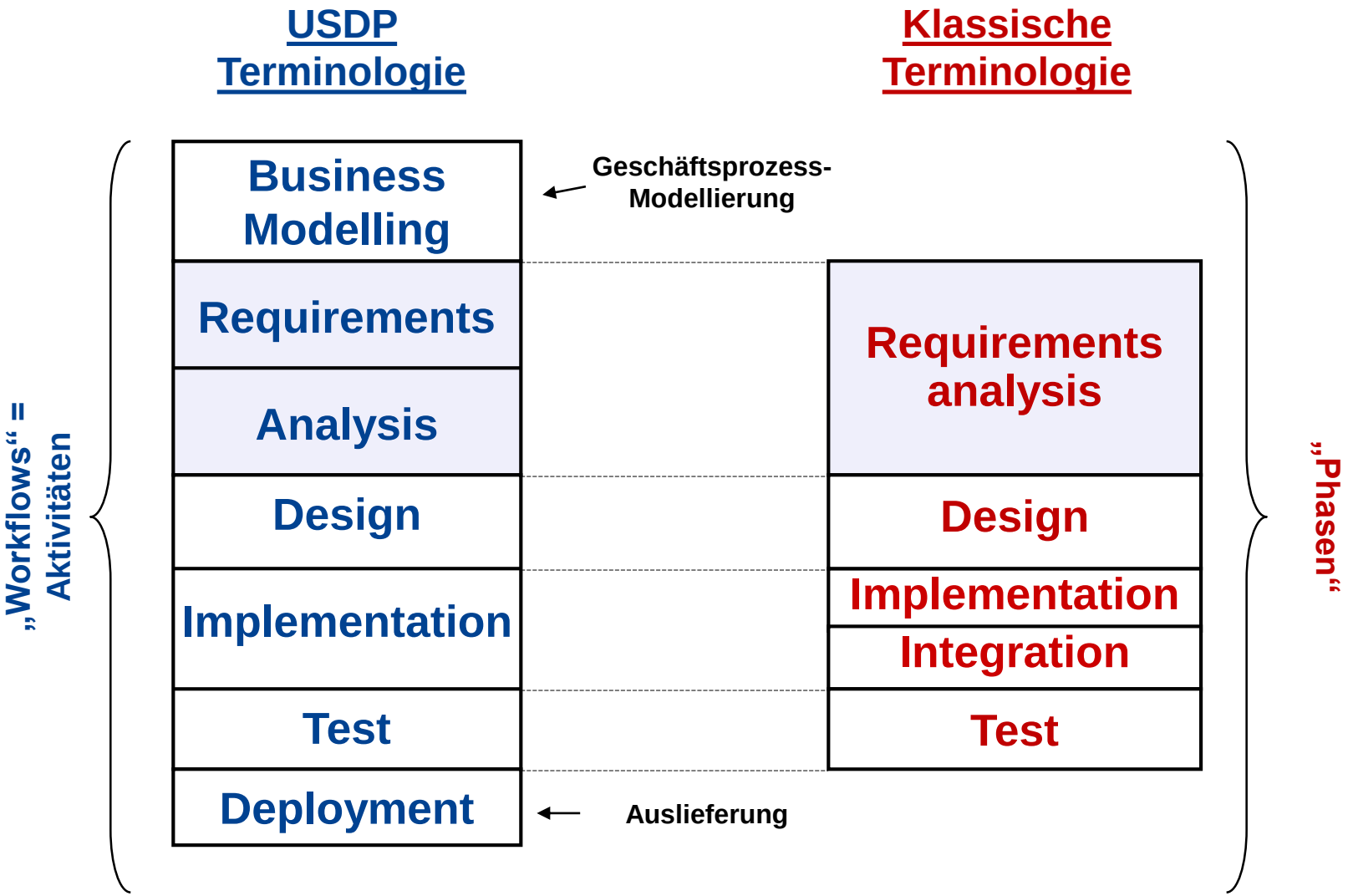


Der „Unified Process“

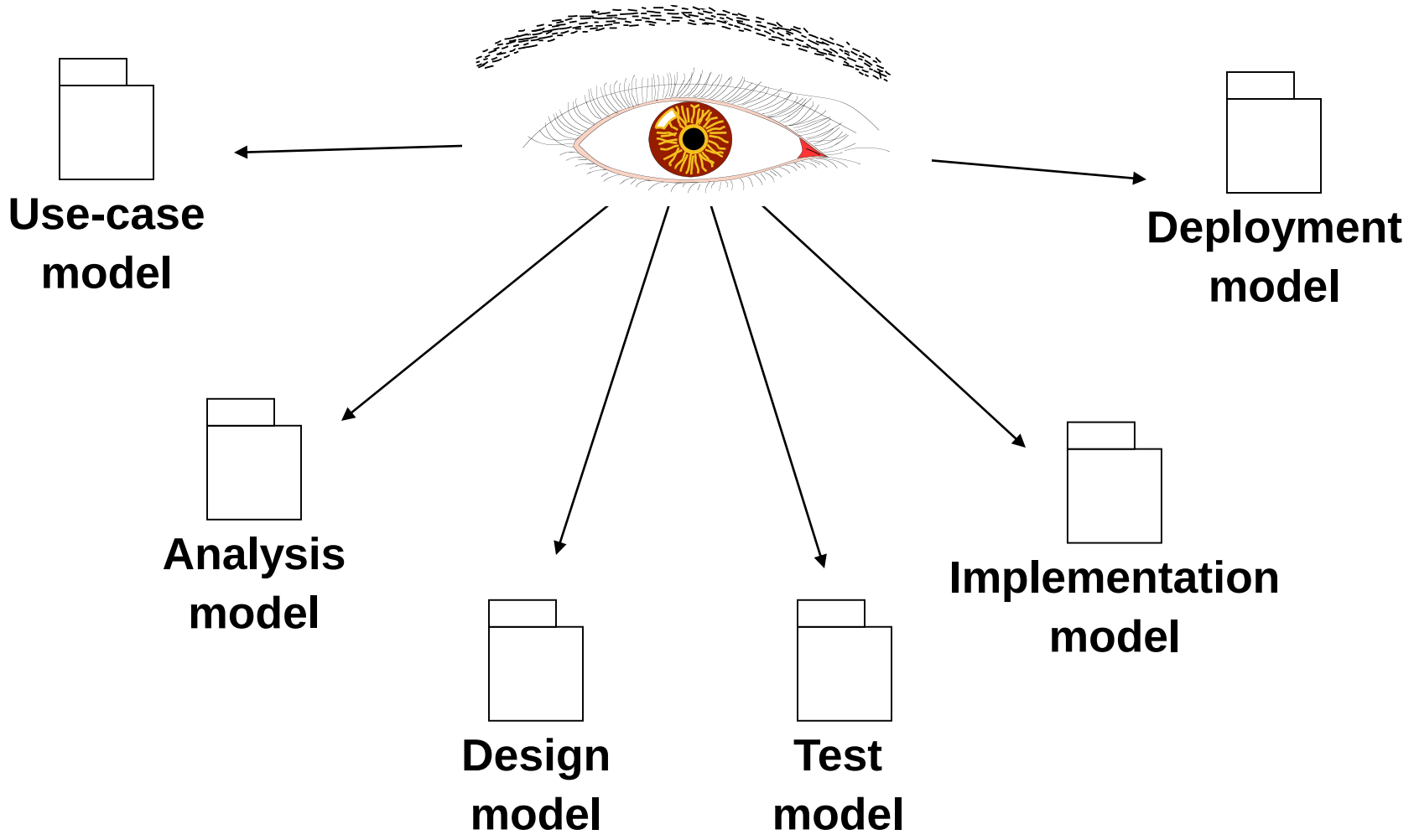
- … ist auf Prozess-Ebene die Entsprechung der Unified Modelling Language (vom gleichen Autorentrio vorangetrieben, vor allem aber Grady Booch)
- … versucht, die vorhandenen Modelle übergeordnet zu beschreiben.

- Merkmale
  - Erweiterung der Aktivitäten und Business Modelling und Deployment (→ Folie 43) was sich in den entsprechenden Notationen der UML wiederspeigelt (→ Folie 44)
  - Iterative, problem- und abhängigkeitsorientierte Vorgehensweise.
  - Klare Definition der Beziehung von „Aktivitäten“ und „Phasen“ bei iterativem und problemorientiertem Vorgehen: Phasen sind dadurch definiert, welchen Aktivitäten *schwerpunktmäßig* durchgeführt werden (→ Folie 45 - 47)

USDP vs. traditionelle Terminologie



Die sechs USDP-Modelle (Sichten der Anwendung)

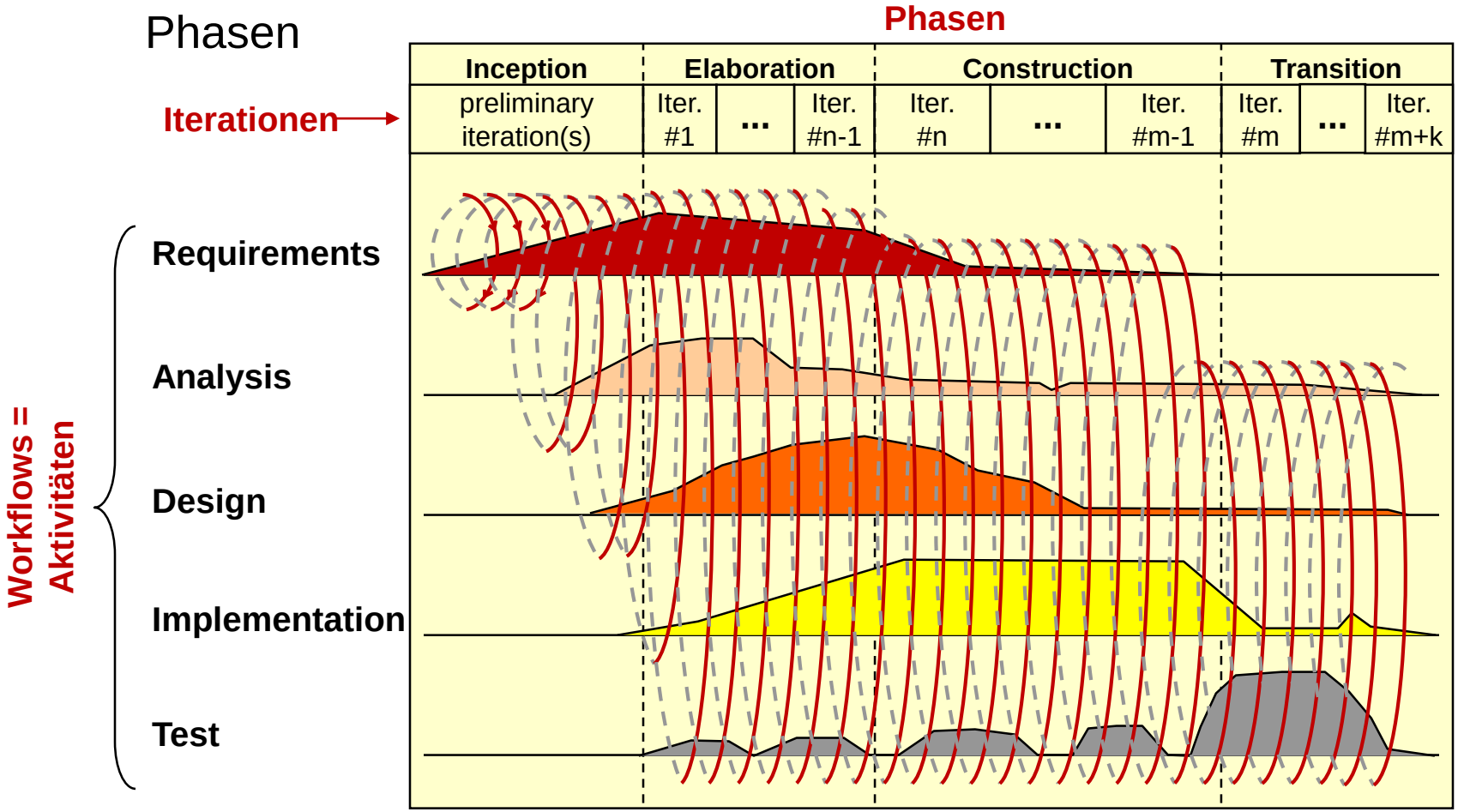


USDP Phasen

- Konzeption (‘Inception’)
  - Umfang des Produktes und dessen Eigenschaften festlegen
  - Machbarkeitsstudien aus wirtschaftlicher Sicht abschließen
  - Die größten Risiken ausschließen
- Ausarbeitung (‘Elaboration’)
  - Möglichst viele Anforderungen erfassen
  - Entwickeln des architektonischen Grundrisses
  - Weitere Risiken ausschließen
  - Abschließend: Kostenschätzung für die nächste Phase
- Konstruktion (‚Construction’)
  - Komplette Entwicklung des Systems
  - Fertig für die Auslieferung an den Kunden
- Inbetriebnahme (‘Transition’)
  - Sicherstellen, dass das Produkt an die User ausgeliefert werden kann
  - User lernen den Umgang mit dem Produkt

Der Unified Software Development Process (USDP)

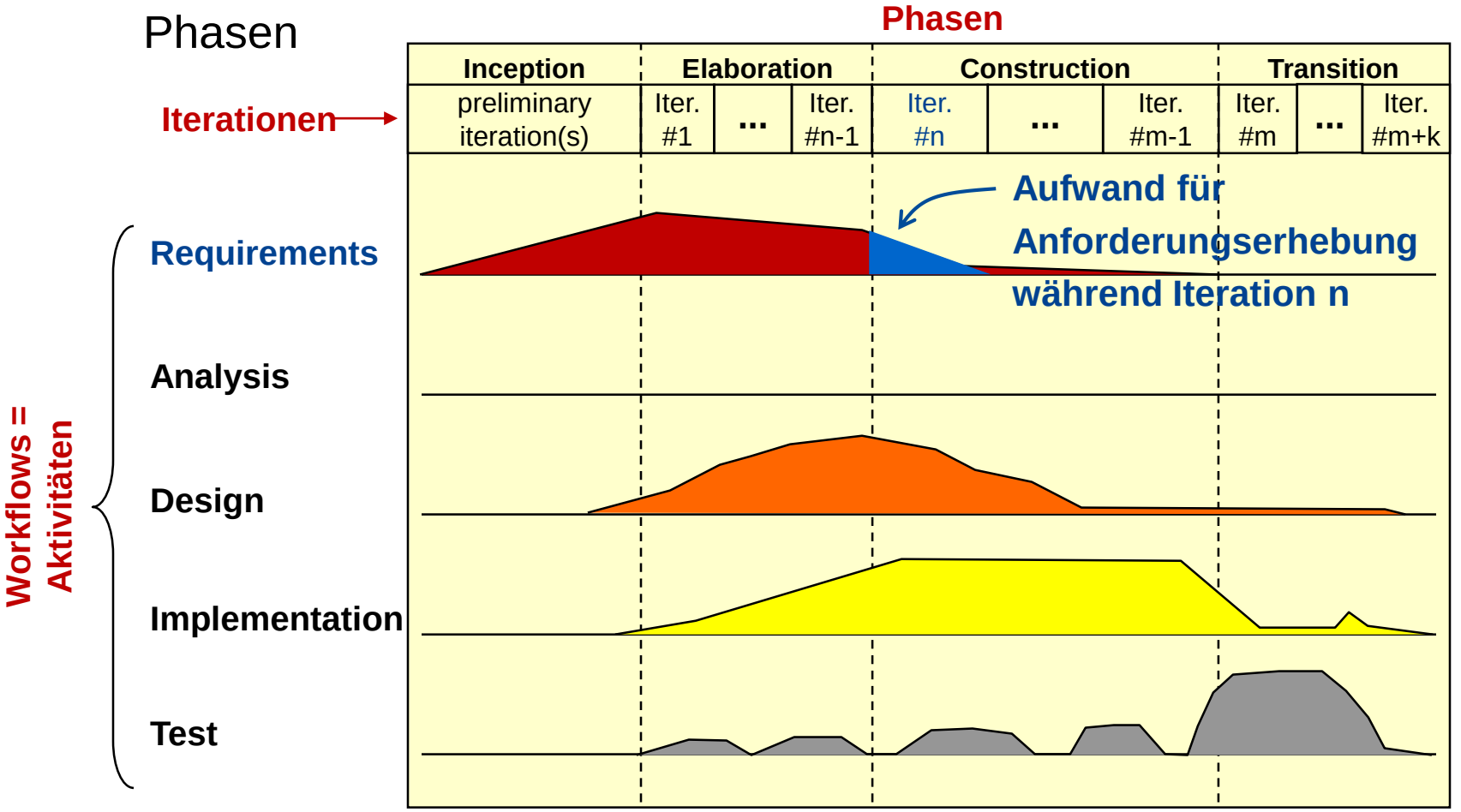
- Problemorientierte Iterationen der Aktivitäten in jeder Phase
- Anteil bestimter Aktivitäten unterschiedlich in den einzelnen Phasen



→ Ivar Jacobsen, Grady Booch, James Rumbaugh: „The Unified Software Development Process“, Addison-Wesley, 1999.

Der Unified Software Development Process (USDP)

- Problemorientierte Iterationen der Aktivitäten in jeder Phase
- Anteil bestimter Aktivitäten unterschiedlich in den einzelnen Phasen



→ Ivar Jacobsen, Grady Booch, James Rumbaugh: „The Unified Software Development Process“, Addison-Wesley, 1999.

Zusammenfassung

- Softwareentwicklungsprozess
  - Organisation und Strukturierung eines Softwareprojektes
- Softwareentwicklungsprozess-Modelle
  - Wasserfall
  - Spiral
  - Issue-Based
  - Unified
- Entwicklung zu immer iterativeren und dynamischeren Prozessen
  - Nächstes Kapitel: „Agile Prozesse“