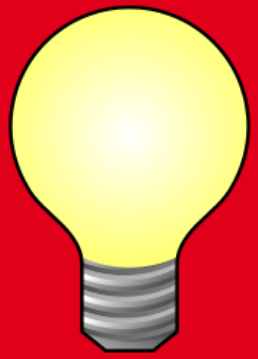
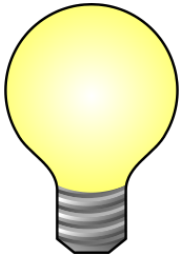
Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

11.02.2021
Programmieren im Grossen VII



Durchführen von Projekten



Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

AGENDA

Einführung ins Thema

Vorgehensmodelle im Detail

Technische Infrastruktur

BSP: Versionsverwaltung

Fazit

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

01
EINFÜHRUNG INS THEMA

Ziel:
Die Eckpunkte des Themas kennenlernen

TYPISCHE FRAGE ZU ANFANG EINES PROJEKTS

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

- Wie gehe ich bei der Softwareentwicklung am besten vor?
  → Suche und nutze ein geeignetes Vorgehensmodell

- Muss ich bei der Softwareentwicklung alles mühsam von Hand machen?
  → Nein, es gibt eine Menge nützlicher Werkzeuge, die einen unterstützen

→ In dieser Vorlesungseinheit besprechen wir:
  – Vorgehensmodelle nochmals im Detail
  – Geeignete Werkzeuge zur Projektunterstützung

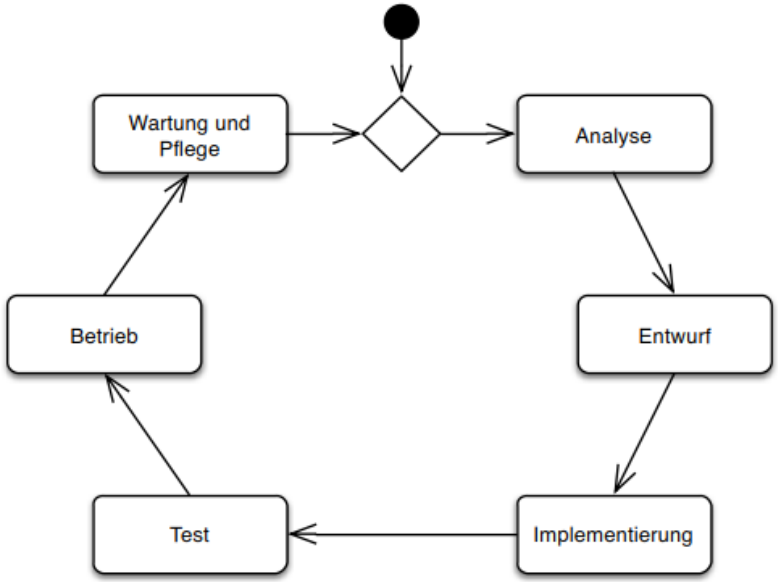
Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

02
Vorgehensmodelle im Detail

Ziel:
Nochmals Details zu den Vorgehensmodellen

BSP FÜR VORGEHENSMODELLE – LEBENSZYKLUS VON SOFTWARE

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

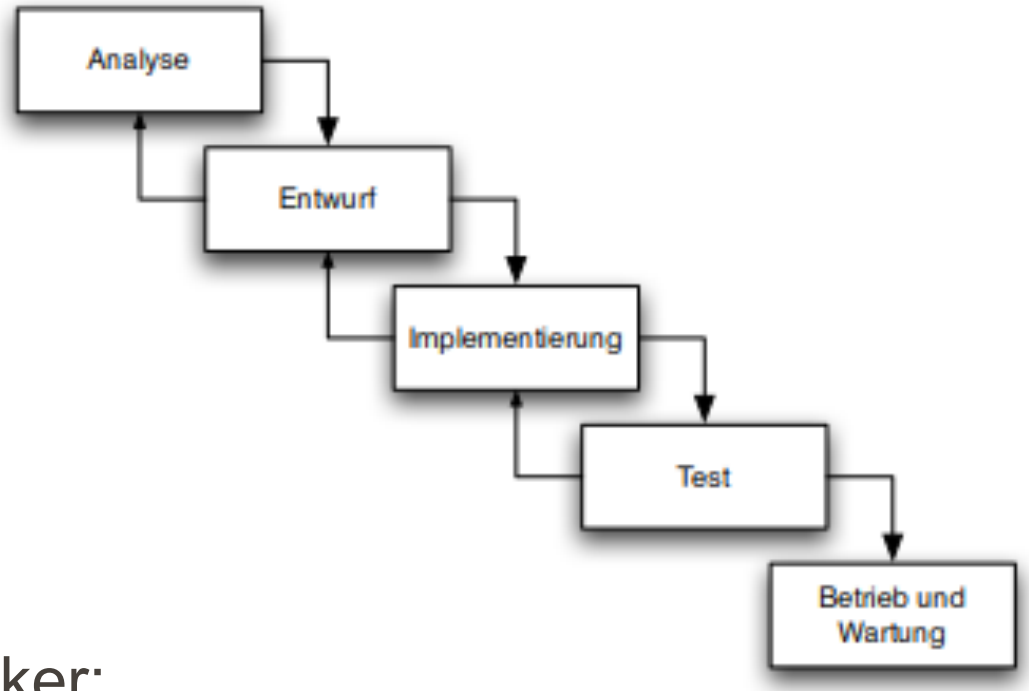


→ Beschreibt die typischen Tätigkeiten bei der SW-Entwicklung

- einfache, klare Struktur
- sehr schlichtes Vorgehensmodell
- wichtigste Aussagen:
  - Software-Entwicklung umfasst 6 grundlegende Tätigkeiten
  - zyklisch
    - („Nach dem Spiel ist vor dem Spiel.“ Sepp Herberger)

BSP FÜR VORGEHENSMODELLE – WASSERFALLMODELL

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim



→ Der Klassiker:

- schwergewichtig & Big Bang Integration
- einfache, klare Struktur
- erlaubt: Rückschritt
- nicht erlaubt: Überschneidung der Phasen

VORGEHENSMODELLE ALLGEMEIN

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

- Ein Vorgehensmodell gibt vor:
  - Reihenfolge des Arbeitsablaufs
  - Aktivitäten und Artefakte
  - Fertigstellungskriterien
  - Verantwortlichkeiten, nötige Kompetenzen und Qualifikationen
  - Standards, Richtlinien, Methoden und Werkzeuge
  - . . .
- Ziemliche Unterschiede jedoch in der Ausführlichkeit:
  - Die Bandbreite reicht von „detaillierten“ (umfassende Liste an Vorgaben) bis hin zu „sehr groben“ Vorgehensmodellen
    - Bsp: „Software Life Cycle“: sehr grob
      (zu allgemein für ein echtes Vorgehensmodell)

GANZ GROBE EINORDNUNG DER VORGEHENSMODELLE

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

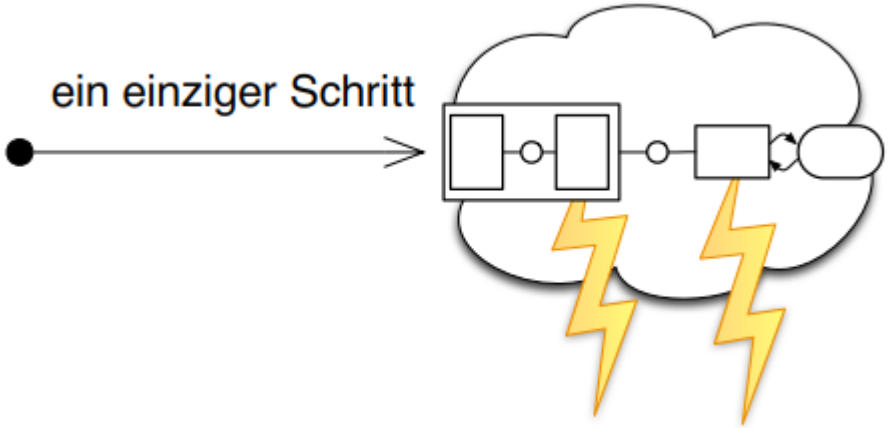
- Schwergewichtig
  - Planung sehr wichtig — spätere Veränderungen unerwünscht
  - Stark dokumenten-orientiert (→ Spezifikationen)
  - Stark ausgeprägtes Phasenkonzept mit Meilensteinen
- Agil / Leichtgewichtig
  - Schnelle Reaktion auf Veränderungen sehr wichtig
  - Konzentration auf das Wesentliche, um Mehrwert für den Kunden herzustellen
    - Eher Menschen-/Kunden-orientiert

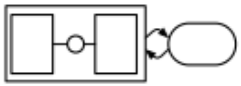
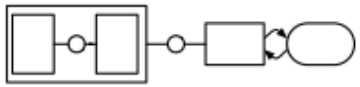
    → Planung + Dokumentation eher nicht so wichtig
- Andere irgendwo dazwischen
  - …

HÄUFIG VERWENDETE SCHLAGWÖRTER

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

- Big Bang-Integration: Alles auf einen Schlag (Knall)

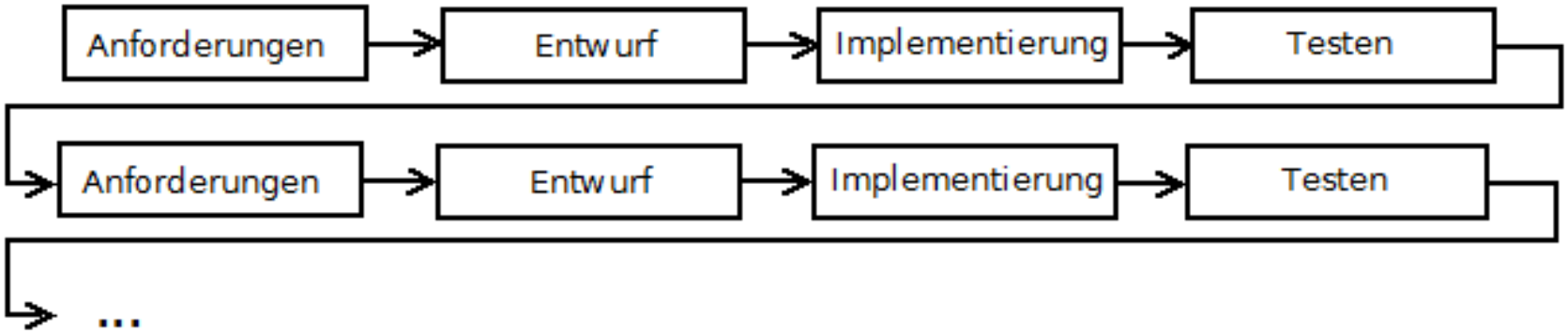


- Inkrementell: Software wächst in klein(er)en Schritten
  - 
  - 
  - 
  - …

HÄUFIG VERWENDETE SCHLAGWÖRTER

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

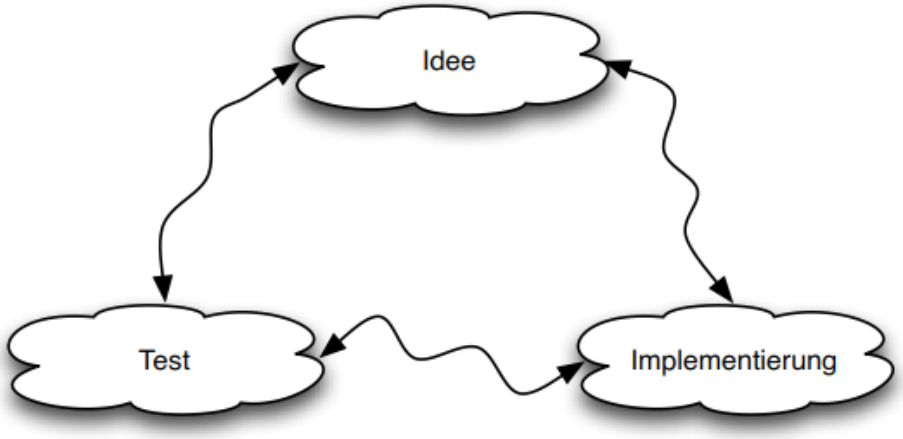
- Iterativ: wiederholte Anwendung einer Vorgehensweise



- Wichtige Bemerkungen:
  - Iterativ ermöglicht inkrementell
  - agil → iterativ und inkrementell + leichtgewichtig
  - Es gibt aber auch schwergewichtige Vorgehensmodelle, die iterativ und inkrementell sind

BSP FÜR VORGEHENSMODELLE – BUILD-AND-FIX-CYCLE

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim



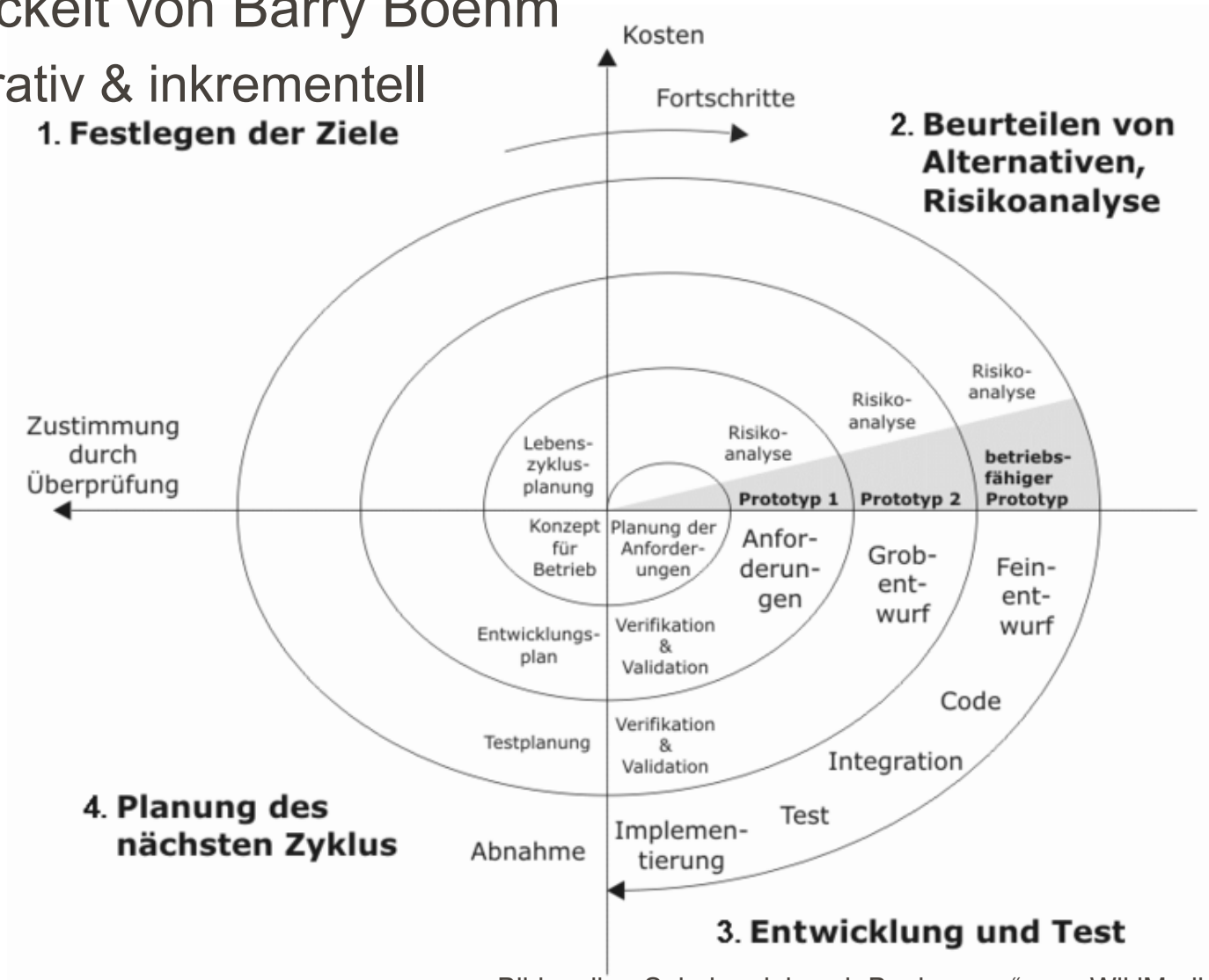
→ Sehr beliebt aber problematisch:

- Streng genommen kein Vorgehensmodell (zu allgemein)
- Verwandte Ansätze/Andere Namen:
  - Versuch und Irrtum (Trial and Error)
  - „Basteln“, „Hacken“, …
- Sehr beliebt
- Manchmal sinnvoll: Neues ausprobieren, Prototypen, ...
- Sonst: problematisch
  → wächst einem schnell über den Kopf

BSP FÜR VORGEHENSMODELLE – SPIRALMODELL

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

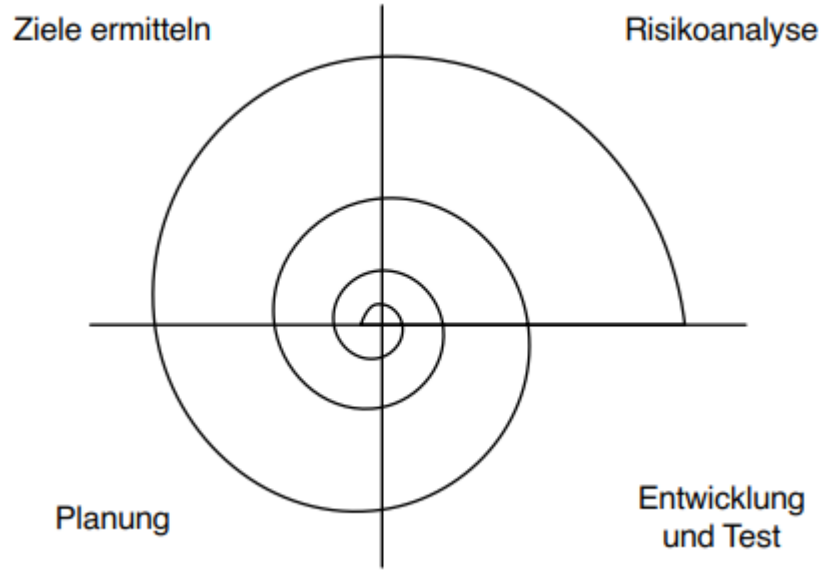
- Entwickelt von Barry Boehm
  - Iterativ & inkrementell



Bildquelle: „Spiralmodel nach Boehm.png“ von WikiMedia Commons

BSP FÜR VORGEHENSMODELLE – SPIRALMODELL

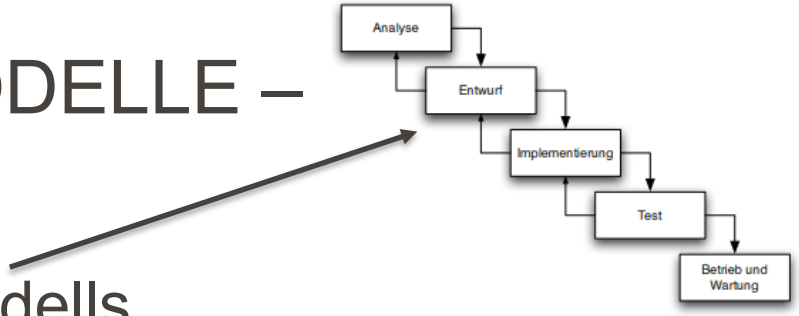
Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim



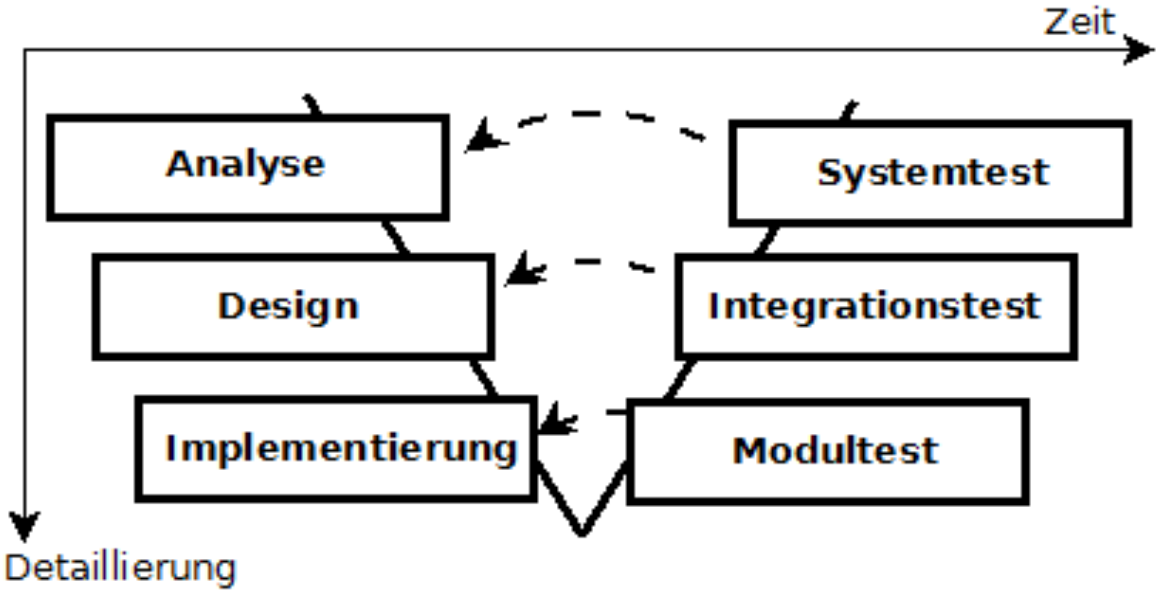
→ Beschreibung des „Prozesses der SW-Entwicklung“:

- Streng genommen kein Vorgehensmodell
  - (eher: Prozessmodell)
- Fokus: Risiken minimieren
- Iterativ & inkrementell
- in jeder Iteration:
  - Einbettung eines (anderen) Vorgehensmodells

BSP FÜR VORGEHENSMODELLE – V-MODELL



- Abwandlung des Wasserfall-Modells
  - Deutsche Erfindung (TU München → siehe Manfred Broy)
  - oft in Deutschland benutzt, z.B. Behörden, Automotive, …



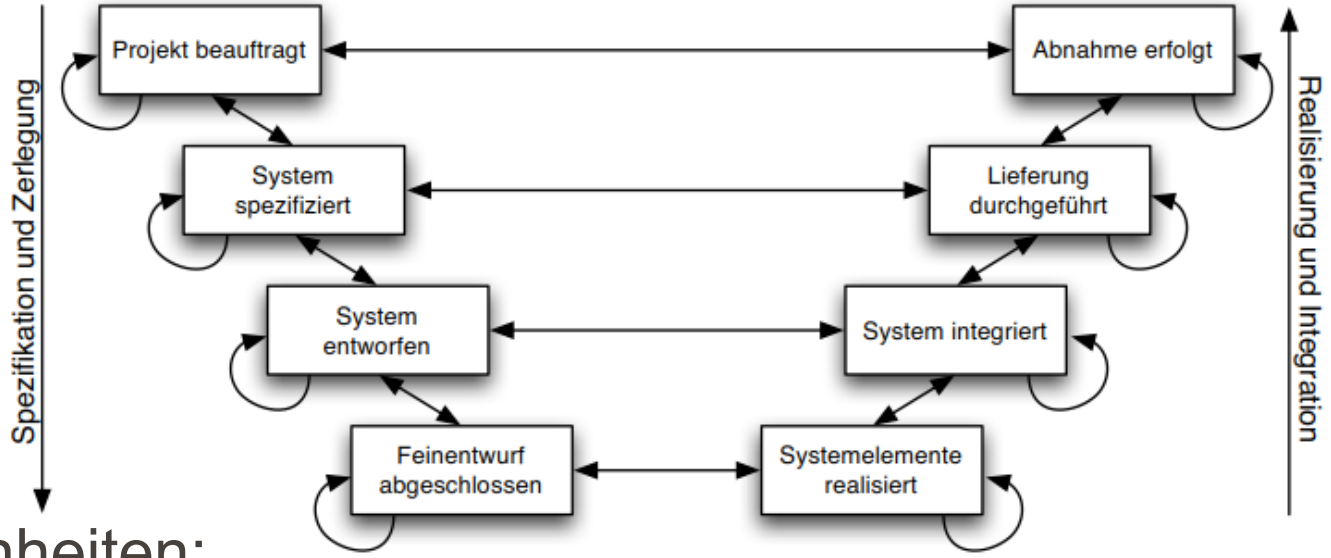
→ Leider sehr starr

→ Zunächst nicht iterativ geplant → Nur ein Zyklus

BSP FÜR VORGEHENSMODELLE – V-MODELL

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

- Auch meilensteinorientierte Betrachtung möglich:



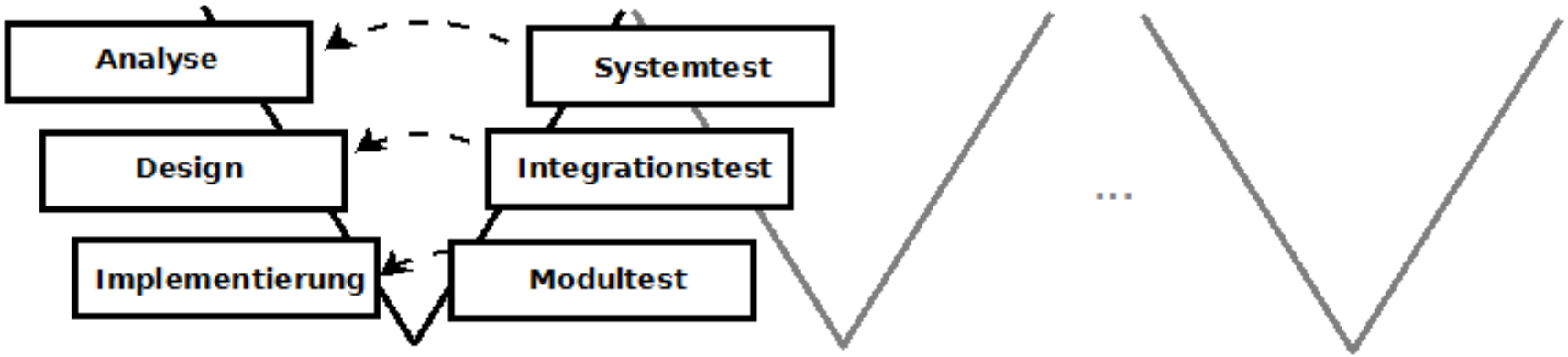
→ Eigenheiten:

- Schwergewichtig & Big Bang-Integration
- Abgewandeltes Wasserfallmodell
- Fokus: Qualitätssicherung
- Sehr komplex/kompliziert
- Teilweise zwingend vorgeschrieben (z.B. Behörden, Automot.)

BSP FÜR VORGEHENSMODELLE – V-MODELL

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

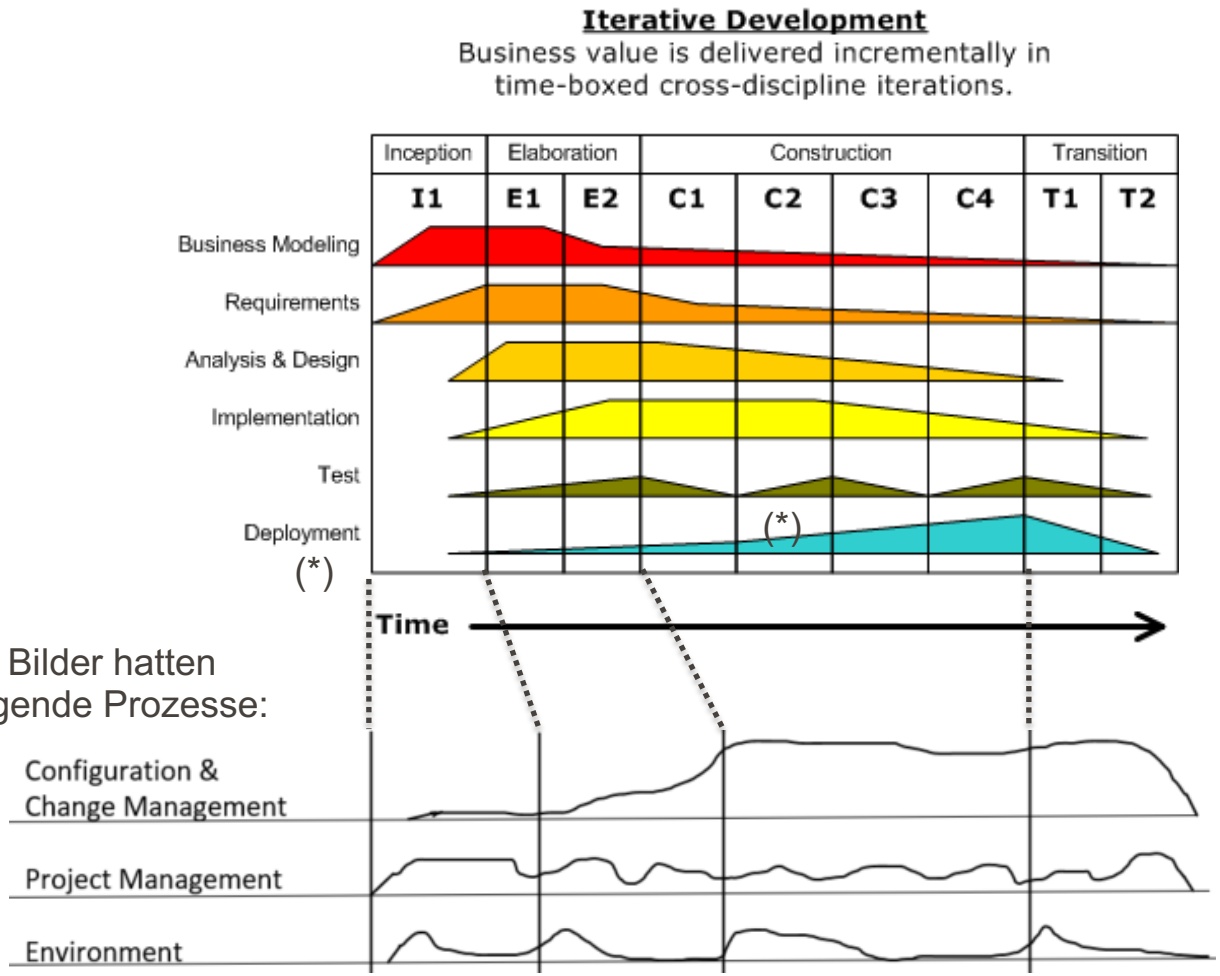
- Weiterentwicklung für iterative Entwicklung:



- Redesign (seit 2005): V-Modell XT
  - Iterativ, inkrementell, wesentlich flexibler und besser skalierbar (Reaktion auf RUP, Spiralmodell & Agile Methoden)

BSP FÜR VORGEHENSMODELLE – (RATIONAL) UNIFIED PROCESS

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

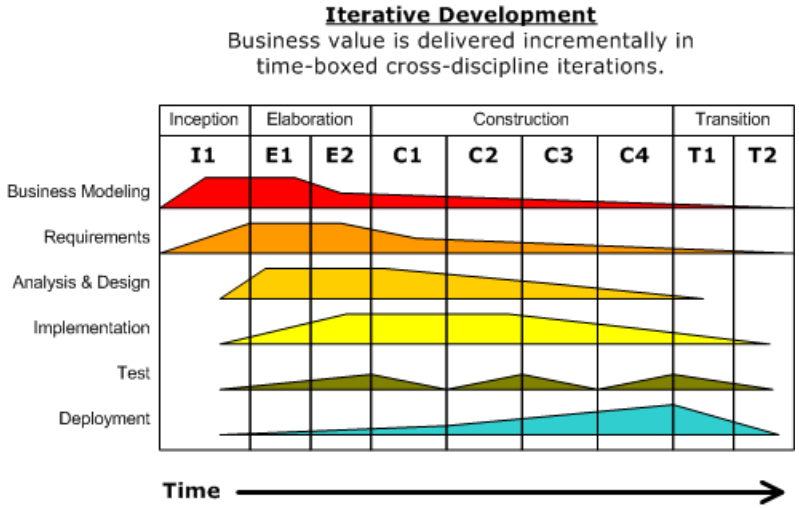


(*) Frühere Bilder hatten noch folgende Prozesse:

Bildquelle: „Development-iterative.png“ von WikiMedia Commons

BSP FÜR VORGEHENSMODELLE – (RATIONAL) UNIFIED PROCESS

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim



→ Charakteristika:

- Schwergewichtig
- Iterativ und inkrementell
  - In jeder Iteration werden hinzugefügt: Weitere Use Cases, weitere Details (z.B. alternative Ablaufschritte), weitere Anwendungs-bestandteile, . . .
- Objektorientiert (UML)
- Anwendungsfall-orientiert
- Architektur-zentriert

Bildquelle: „Development-iterative.png“ von WikiMedia Commons

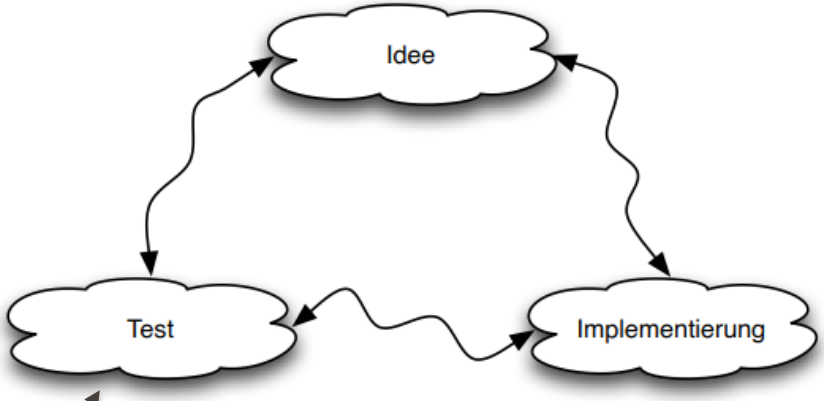
AGILE METHODEN

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

- „Relativ“ neu
  - Abkehr von starren Prozessen
  - Keine richtigen Vorgehensmodelle in klassischen Sinn
  - Eher Sammlung von Prinzipien (Best Practices)

    → Muster-Idee → Prozessmuster (siehe Vorlesung zu Mustern)
  - Siehe auch: **Agiles Manifest**
- Typische Beispiele:
  - eXtreme Programming (XP) → Für Entwicklung
    - Nach Win Vista-Katastrophe hat Microsoft auf XP gesetzt → Win 7
  - SCRUM
    - Eigentlich eher eine Projektmanagementmethode
    - Für Entwicklung kann z.B. XP genutzt werden oder anderes
    - Derzeit richtig „IN“

BUILD-AND-FIX-CYCLE ↔ AGILE METHODEN



→ Agilen Methoden werden gern mit verwechselt

- „Agil“ wird gerne als Schlagwort verwendet, aber es wird dabei „Hacking“ / Build-And-Fix-Cycle gemeint
- Zugegeben: Agile Methoden funktionieren sehr ähnlich
- ABER bei Agilen Methoden gilt:
  - Best Practices greifen wie Zahnräder ineinander
    - Best Practices → Prozessmuster!
    - Z.B.: Kontinuierliches autom. Testen ↔ Refactoring ↔ …

Wirkungen verstärken sich gegenseitig

→ Agile Methoden verlangen **viel Eigendisziplin** des Entwicklers alle Bausteine einzuhalten → sonst zerfällt das „Getriebe“

→ Hier z.B. müssen dann für alle „Ideen“ auch autom. Tests erstellt werden, es müssen Refactorings stattfinden, …

→ Hängt sehr stark von „guten“ Leuten ab

BSP FÜR AGILE METHODEN – EXTREME PROGRAMMING (XP)

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

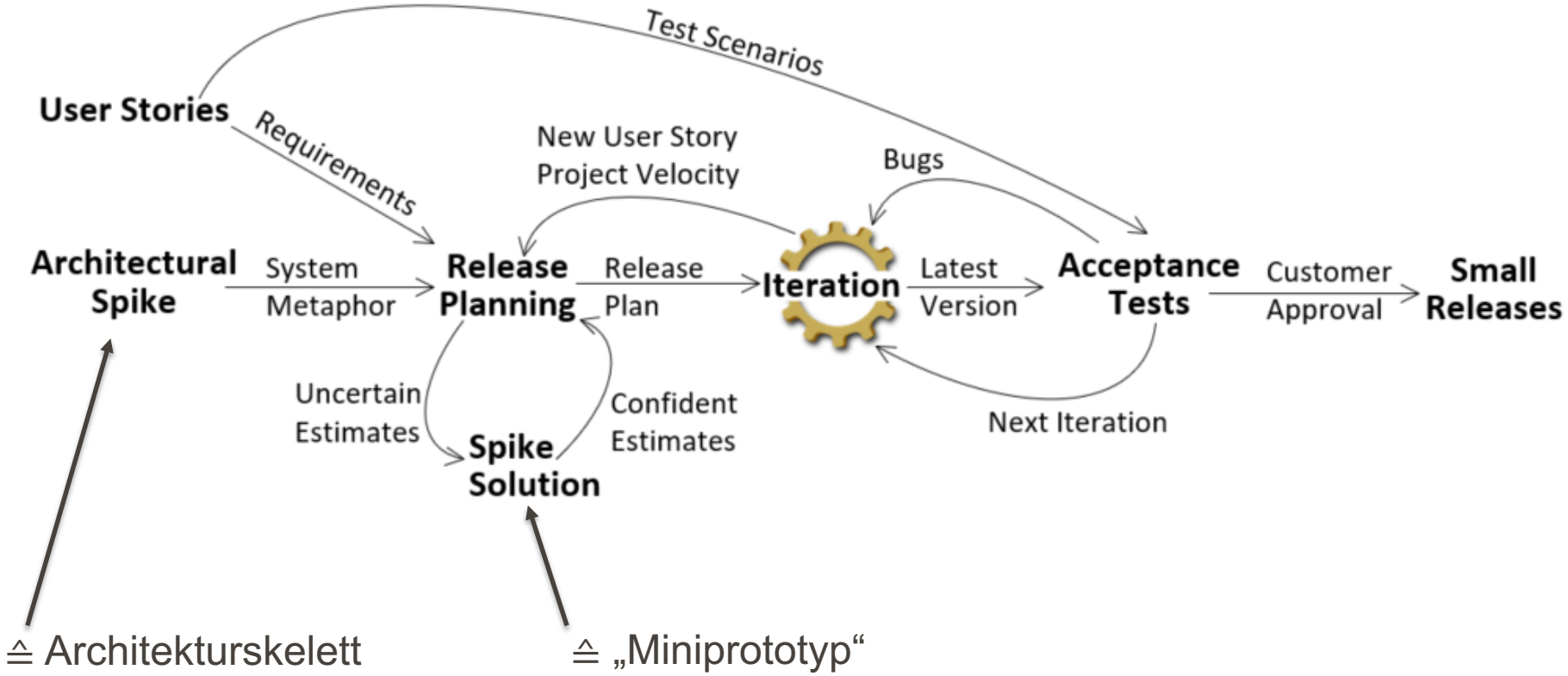
- Häufiges Missverständnis:
  - „extreme“ ≙ extrem schlampige Programmierung / „Hacking“
- Eigentliche Bedeutung von „eXtreme“
  - „Take *best practices* to the extreme!“

  → Best Practice-Gedanke leben
- 12 Prinzipien (Best Practices):
  - Testen ist gut → Test-Driven Development
    - (Testfirst & Testautom.)
  - Frühes Integrieren ist gut → Continuous Integration
  - Code Review ist gut → Pair Programming
  - Kommunikation mit Kunde wichtig → Kunde ist Teil des Teams
  - . . .

BSP FÜR AGILE METHODEN – EXTREME PROGRAMMING (XP)

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

- „Artefakt- & Prozessmodell“:



Quelle: http://www.extremeprogramming.org/map/project.html

BSP FÜR AGILE METHODEN – EXTREME PROGRAMMING (XP)

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

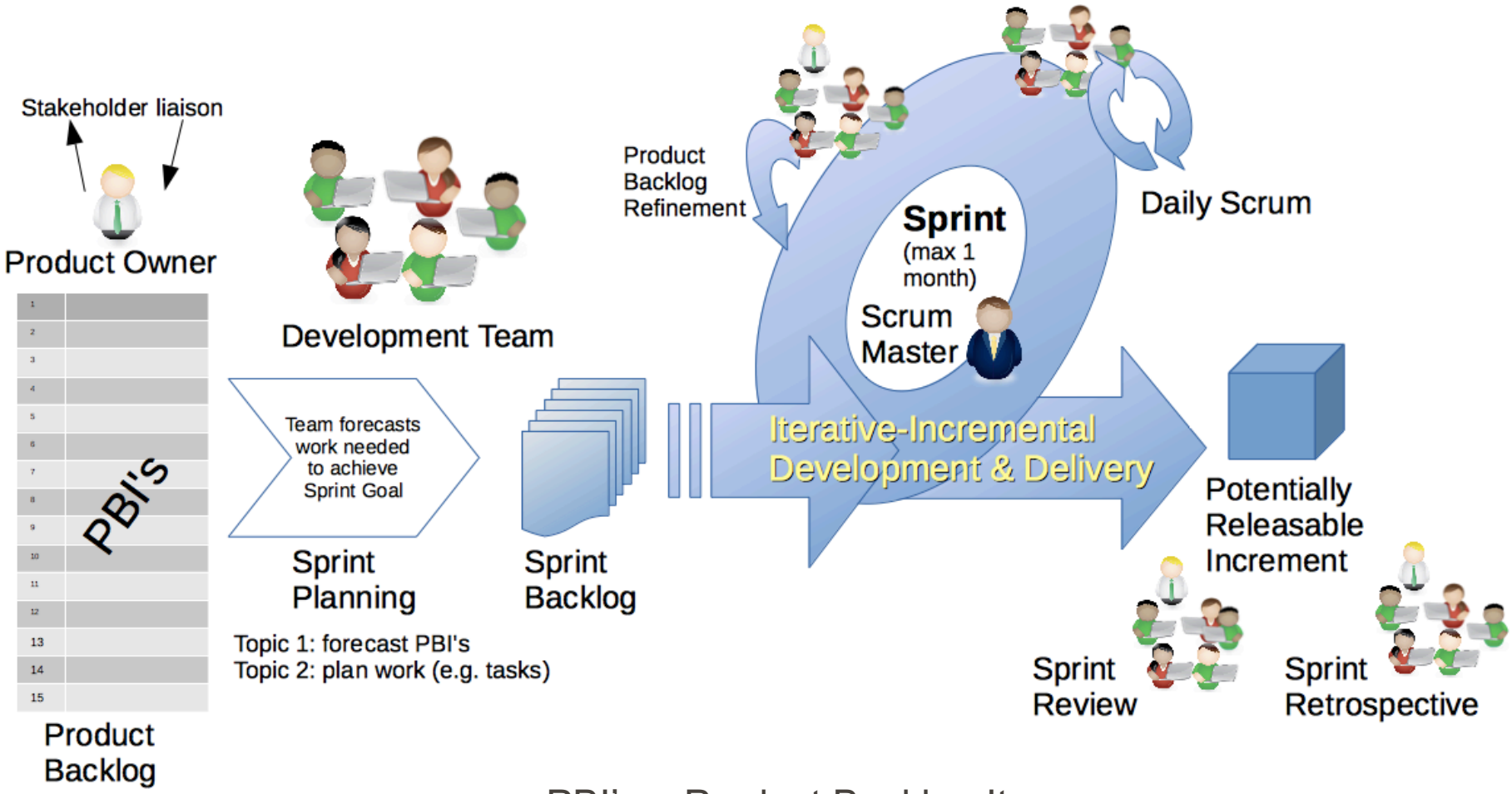
- Anforderungen an das Team:
  - **hohes Maß an Eigendisziplin**
  - geeignete Qualifikation → gute Leute essentiell!
- Weitere Merkmale von eXtreme Programming
  - Agil (erste richtig populäre agile Methode)
  - Anforderungen werden nur kurz als sog. User Stories skizziert
  - Veränderung von Anforderungen ist kein Problem (“Embrace Change”)
  - Iterativ und inkrementell → jede Iteration als „Time Box“
    - Evolutionärer Prototyp wird zum System

    → Immer lauffähiges System zur Hand

    → Möglichst alle Tests werden automat. und immer auf “grün”
  - Kurzer aber vollständiger Entwicklungszyklus → Greifbares Ergebnis
  - Aktivitäten: coding, testing, listening, designing
  - Definierte Rollen

BSP FÜR AGILE METHODEN – SCRUM

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim



PBI's = Product Backlog Items
(z.B.: User Stories, Epics, Change Reqests, Bugs, …)

Quelle: Wikimedia Commons (https://commons.wikimedia.org/wiki/File:Scrum_Framework.png, Autor: Dr Ian Mitchell)

BSP FÜR AGILE METHODEN – SCRUM

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

- Weitere Agile Methode → derzeit sehr „IN“ (Projekt-management)
  - Eigentl. kein Vorgehensmodell, sondern PM-methode
  - SCRUM kann mit anderen Meth. (z.B. XP) kombiniert werden

- Konzepte:
  - Kurzer Entwicklungszyklus (Sprint) mit Zwischenetappen
    - 4 Wochen: 1 Sprint (ein Inkrement/Entwicklungszyklus)
    - 1 Woche: Weekly Scrum
    - 1 Tag: Daily Scrum (Meeting zum Lösen der Probleme des Tags)
  - Anforderungen werden als User Stories nur kurz skizziert
  - Sammlung im Backlog → Priorisierung der User Stories
  - Zu Beginn eines Sprints werden die wichtigsten User Stories aus dem Backlog genommen und in den Sprint Backlog übernommen und geplant

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

03
Technische Infrastruktur

Ziel:
Geeignete Werkzeuge zur Unterstützung in Projekten kennen und nutzen

TECHNISCHE INFRASTRUKTUR – WOFÜR BRAUCHEN WIR DAS?

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

- Was meint technische Infrastruktur?
  - Sammlung an Werkzeugen (Programmen), die bei der SW-entwicklung helfen
  - Arbeiten hoffentlich so zusammen, dass Sie eine wirkliche Infrastruktur bilden

    → Möglichst keine Brüche
- Wofür ist das wichtig?
  - Effizientes Arbeiten & Strukturierung
  - Kollaboratives Arbeiten im Team

  → Bewältigung der aus den verschiedenen Artefakten und der daran arbeitenden Menschen entstehenden Komplexität

TECHNISCHE INFRASTRUKTUR

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

- Die wichtigsten Werkzeuge in einem Entwicklungsprojekt:
  - Integrierte Entwicklungsumgebung (Eclipse IDE, NetBeans, …)
  - Testautomatisierung (JUnit, NUnit, PHPUnit, . . .)
  - Versionsverwaltung (Subversion (svn), Git (derzeit sehr populär))
    → Wichtig für Konfigurationsmanagement (siehe folgendes Kap.)
  - Build-Automatisierung (make, ant, gradle, …) → DevOps
  - Continuous Integration Server (Cruise Control, Trac)
  - Issue-Tracking (Bugzilla, Mantis, . . .), Backlog-Verwaltung
  - Projektmanagement (Open Workbench, MS Project, . . .)
- Empfehlung:
  - Nicht zu viele **neue** Werkzeuge auf ein Mal
  - Lieber nach und nach einführen

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

04
BSP: Versionsverwaltung

Ziel:
Nutzen von Versionsverwaltungssystemen kennenlernen
→ Konfigurationsmanagement kennenlernen

VERSIONSVERWALTUNGS-WERKZEUGE

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

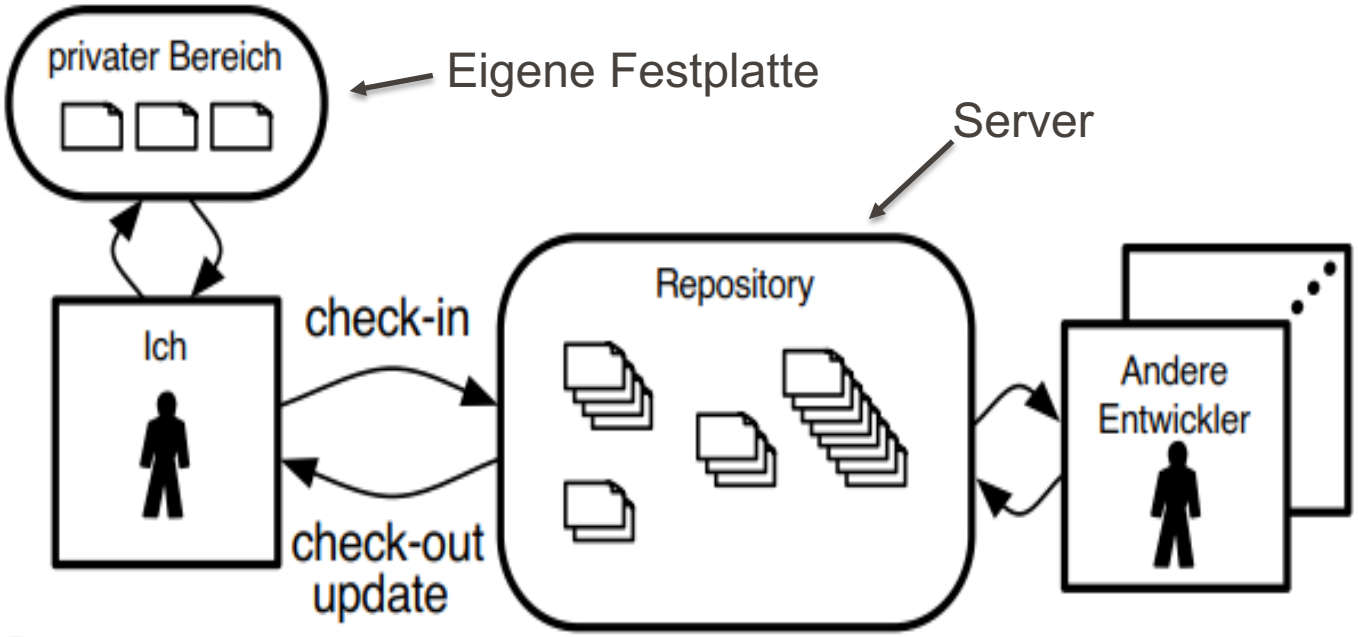
- Wofür brauchen wir eine Versionsverwaltung?
  → Verwaltung aller Artefakte mit den verschiedenen Versionen
  - Backup-Restore
    - Wiederherstellen eines Standes, falls etwas kaputt gegangen ist
  - Nachvollziehbarkeit von Veränderungen
    - Stand der Datei X vor dem letzten Release
  - Parallele Entwicklung ermöglichen
  - Rekonstruktion vorheriger Konfigurationen
    - z.B. Quellcode-Dateien für Binaries, die mit Release XY geliefert wurden.

→ Versionsverwaltung ist **essentieller Bestandteil** einer **professionellen** Softwareentwicklung
  – Nur Amateure & Bastler arbeiten ohne!!

VERSIONSVERWALTUNGS-WERKZEUGE

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

- Bekannte Versionsverwaltungen:
  - Subversion (SVN) – Open Source, immer noch oft verwendet
  - Git – Open Source + neuere –revolutionäre– Konzepte
- Grober Aufbau einer „klassischen“ Versionsverwaltung:



VERSIONSVERWALTUNGS-WERKZEUGE

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

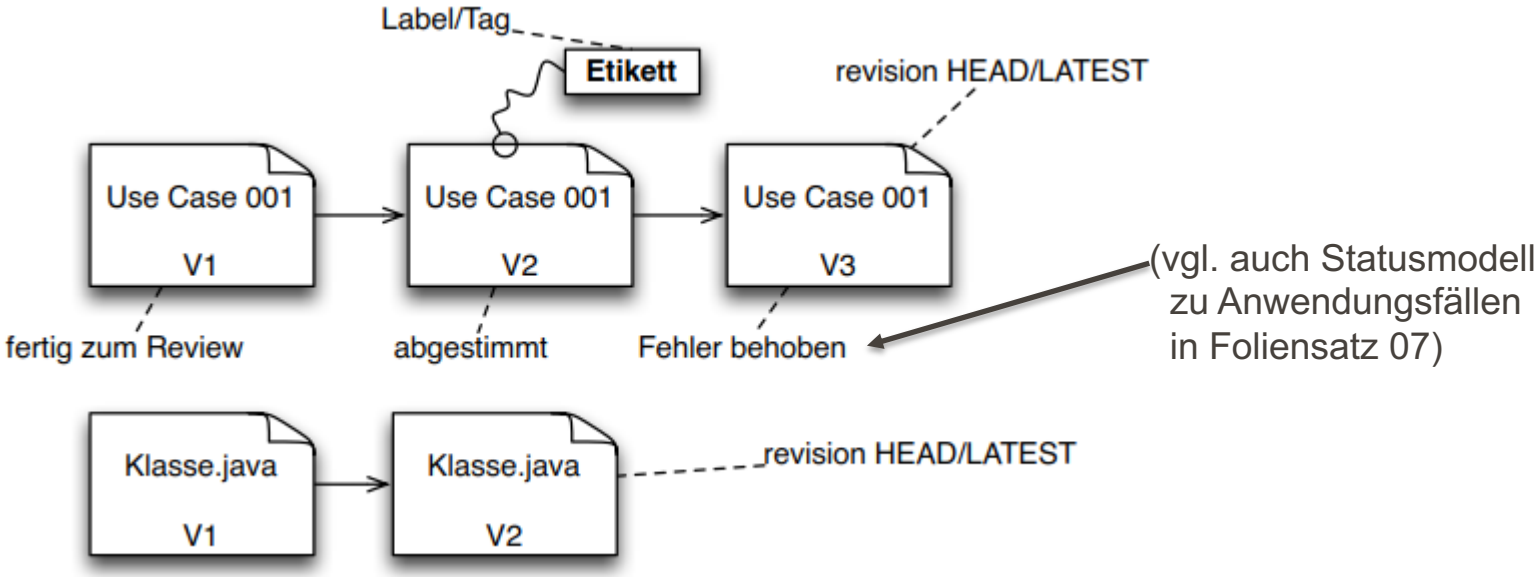
- Grundlegende Anforderungen:
  - Transparenz der Änderungen
    - Veränderung: Wer? Wann? Was?
  - Simultaner Zugriff + Konfliktlösung
  - Rekonstruktion
    - jederzeit beliebige Revision eines beliebigen Artefakts
    - jederzeit beliebige Konfiguration (alle Dateien zu best. Stand)
  - Baselining / Labeling
    - Einfrieren von Ständen verschiedener Dateien zueinander (z.B. alle Stände der Dateien, die bei einem Release betroffen sind)

      → Bekommen ein sog. Label

WIE FUNKTIONIERT DIE VERSIONIERUNG VON DATEIEN?

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

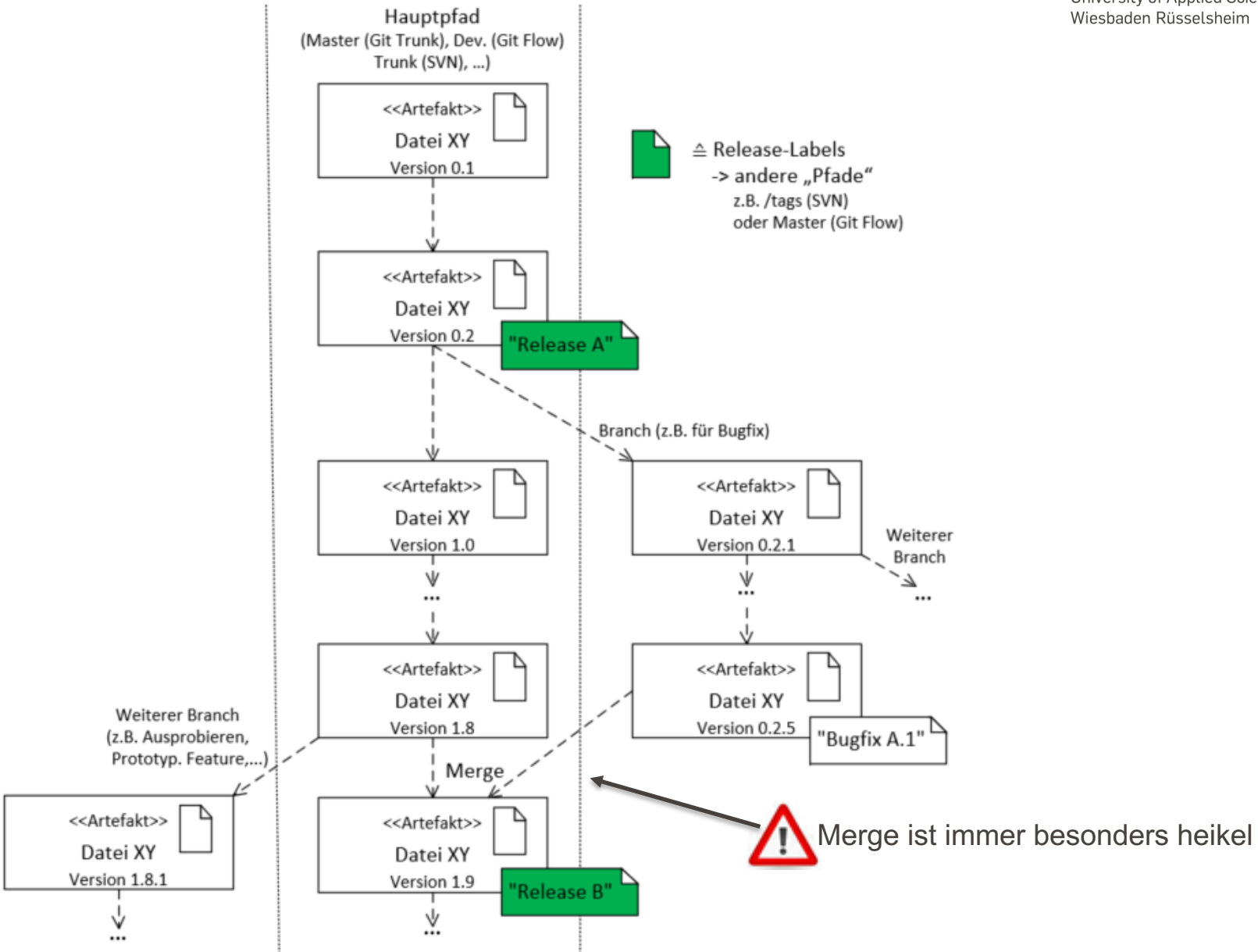
- Für jede Datei / Verz. wird ein Repräsentant (Head) angelegt
- Für jede Version der Datei / Verz. → sog. Revision erzeugt
  - Die Revision enthält die Datei in der jeweiligen Version
  - Daraus entsteht dann jeweils ein Historienbaum für jede Datei:



  - Jede Revision hat eine Revisionsnummer und kann z.B. durch Labels/Tags weiter gekennzeichnet werden

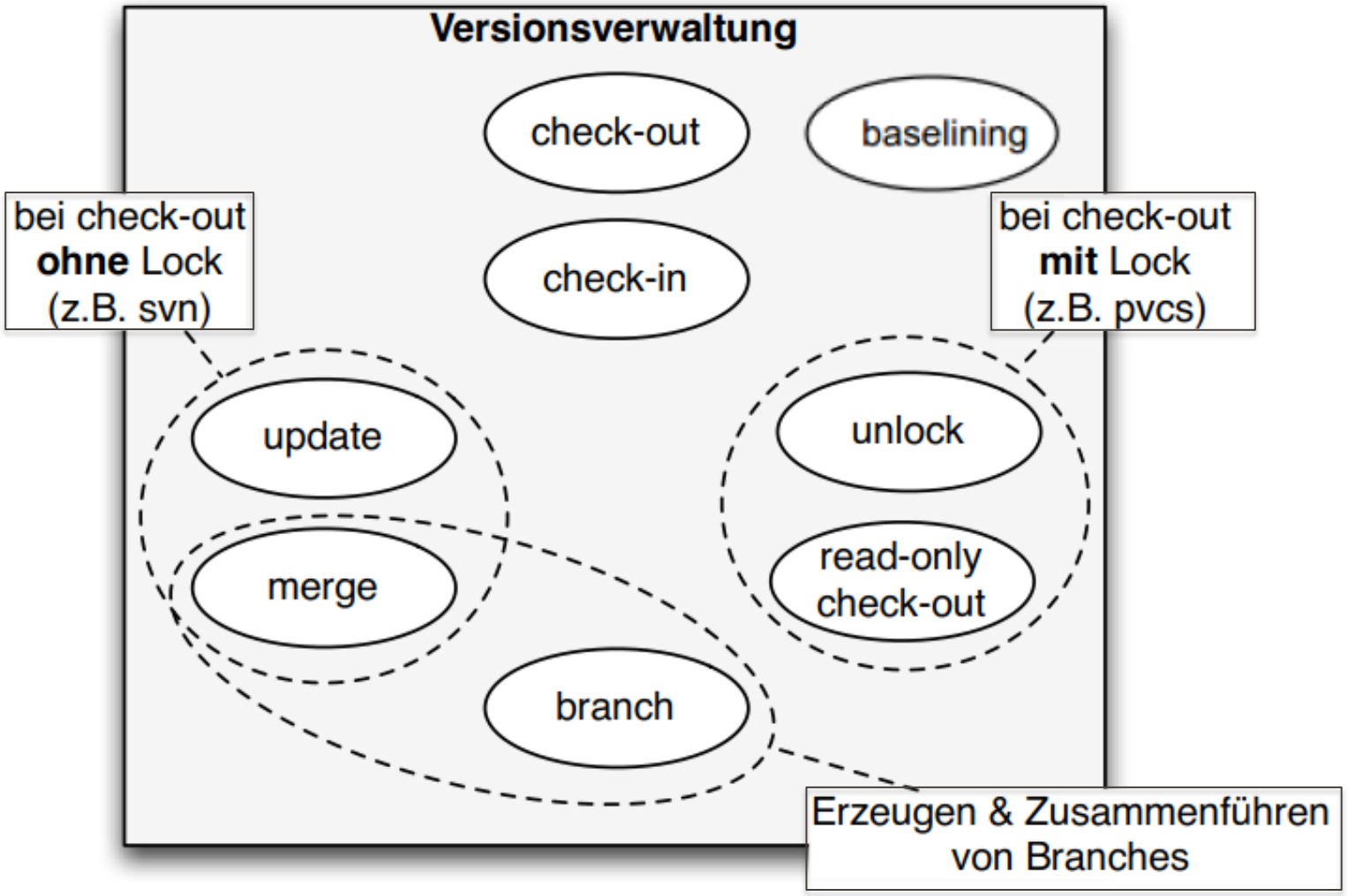
VERSIONSVERWALTUNG – BSP. FÜR VERSIONSBAUM:

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim



VERSIONSVERWALTUNG – GRUNDLEGENDE USE CASES

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim



KONFIGURATIONSMANAGEMENT

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

- Was sollte versioniert werden?

  → Alle Dateien, die mit einem Projekt zu tun haben

  - Code, Tests, Anforderungen, Designdokumente, Sonstige Texte, Möglichst alle Entwicklungswerkzeuge (haben auch Versionen!)

- Zu bestimmten Zeiten müssen alle Dateistände zueinander eingefroren werden → Baselines
  - Zu definierten Meilensteinen (meist vor und nach dem Review, …)
  - Zu bestimmten Meilensteinen müssen evtl. nur bestimmte Dateien zueinander versioniert werden (sog. Konfiguration)

- Zu freigegebenen Releases müssen evtl. noch Bugfixes, Patches, Service Packs nachgereicht werden

  → Branching erforderlich → Verzweigung der Entwicklung

  → Branches müssen später wieder in den Hauptzweig gemergt werden

KONFIGURATIONSMANAGEMENT

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

→ Puh, das ist ja doch ganz schön kompliziert!

→ Deshalb gibt es in größeren Projekten meist mindestens einen Verantwortlichen, der hier den Überblick behält

→ Prozess wird als Konfigurationsmanagement bezeichnet

- Typische Tätigkeiten des Konfigurationsmanagements:
  - Konfigurationsmanagementplan entwickeln
    - Welche Dateien müssen versioniert werden?
    - Wann müssen welche Baselines gezogen werden?
    - Wie verläuft ein Branching-Prozess?
    - …
  - Baselining durchführen, Probleme in der Versionsverwaltung lösen
  - Bei Branches den Rückmerge sicherstellen
  - …

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

05
Fazit

Ziel:
Was haben wir damit gewonnen?

WAS HABEN WIR GELERNT?

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

- Vorgehensmodelle im Detail
  - Big Bang, iterativ, inkrementell
  - Schwergewichtig ↔ leichtgewichtig
  - Build-and-Fix-Cycle, Software Life Cycle, Spiralmodell
  - Wasserfall, V-Modell, RUP
  - Agile Methoden: eXtreme Programming, SCRUM
  - Lastenheft, Pflichtenheft
- Technische Infrastruktur zur Softwareentwicklung
  - Die wichtigsten Werkzeugarten
  - Im Fokus: Versionsverwaltung & Konfigurationsmanagement

GESAMTÜBERBLICK ÜBER DAS IN SWT GELERNTE:

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

- Modellierungssprachen
  1. UML
     - Struktur- und Verhaltens-Diagramme
     - Einsatzzweck (Analyse — Entwurf — Implementierung)
  2. FMC-Blockdiagramme
- Programmieren im Großen
  - Analyse (Anwendungsfälle, Fachmodell, GUI-Entwurf)
  - Entwurf (Grobentwurf, Feinentwurf, Muster)
  - Test
- (Programmieren im Kleinen

(Programmiermethoden)

  - Doku (Design by Contract, Refactoring,)
  - JUnit-Test)

LITERATUR

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

- Vorgehensmodelle & Werkzeuge für die SW-Entwicklung
  - Kleuker: Grundkurs Software-Engineering mit UML. [http://dx.doi.org/10.1007/978-3-8348-9843-2]
  - Balzert: Lehrbuch der Softwaretechnik. 2008 [BF 000 103].
  - Hoffmann: Software-Qualität. 2008 [http://dx.doi.org/10.1007/978-3-540-76323-9]
  - Van Vliet: Software Engineering, Wiley 2008.

- (R)UP
  - Kleuker: Grundkurs Software-Engineering mit UML [http://dx.doi.org/10.1007/978-3-8348-9843-2]
  - Zuser et al: Software-Engineering mit UML und dem Unified Process. [BF 500 92].
  - C. Larman: Applying UML and Patterns [30 BF 500 78].

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

AUF GEHT‘S!!

SELBER MACHEN UND LERNEN!!