

# Software Engineering

Mark Keinhörster

FOM  
Hochschule für Ökonomie und Management

7. April 2017

## 1 Projektmanagement

# Voraussetzungen

## Was möchten Sie gerne behandeln?

- 
- 
- 
- 
-

## Was sollten Sie am Ende können?

- Software Engineering als Teildisziplin der Informatik kennen
- Grundpfeiler des Software Engineering kennen
- Vorgehensmodelle der Softwareentwicklung beschreiben und abgrenzen
- Softwarequalität messen und bewerten
- Weiterführende Konzepte verstehen und anwenden

## Projektmanagement

Risikomanagement  
Qualitätsicherung  
Messen/Bewerten

# Einführung

## Aufgabe: Rekursives Take

Implementieren Sie die Methode “take(int n)”, die die ersten n Elemente eines Übergebenen Arrays vom Typ int als neues Array zurückgibt.



- Anforderungen mehrere 100 Seiten lang
- Anforderungen unklar, widersprüchlich, flexibel
- International verteilte Teams
- Mehrere tausend Nutzer in 5 Ländern
- Unterstützung von Chrome, Firefox, IE 6
- 6 Monate Projektlaufzeit, 500.000 LOC



Standish Group (<http://www.standishgroup.com>) veröffentlicht jährlich "Chaos Report".

## Chaos Report 2015

- 19% der betrachteten IT-Projekte scheitern
- 52% der betrachteten IT-Projekte drohen zu scheitern
- 29% der betrachteten IT-Projekte sind erfolgreich

*On June 4, 1996, on its maiden flight, the Ariane-5 was launched and performed perfectly for approximately 40 seconds. Then it began to veer off course. At the direction of the Ariane ground controller, the rocket was destroyed by remote control. . . total cost of the disaster was 500 million dollar.*

- Flugbahn wird durch “Inertial Reference System (SRI)” gemessen, dessen Software teilweise von Ariane-4 übernommen wurde.
- Andere Flugbahndaten erzeugten Überlauf bei Konvertierung von 64-Bit Floating Point in 16-Bit Integer und verursachten Fehlfunktion des SRI-Systems.

- Heartbeat hält TLS-Verbindung am Leben
- Eine Seite schickt beliebig langen Payload, Gegenseite schickt die gleichen Daten wieder zurück
- Indikator für aufrechte Verbindung

# Softwarekatastrophe Heartbleed

Mark Keinhörster

Projektmanagement

Risikomanagement

Qualitätsicherung

Messen/Bewerten

SERVER, ARE YOU STILL THERE?  
IF SO, REPLY "BIRD" (4 LETTERS).



User Olivia from London wants pages about "ha bees in car why". Note: Files for IP 375.381.283.17 are in /tmp/files-3843. User Meg wants these 4 letters: BIRD. There are currently 34 connections open. User Brendan uploaded the fileelffeing (contents: 834ba962e2ce9ff89bd3b-ff89)

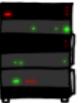


HMM...



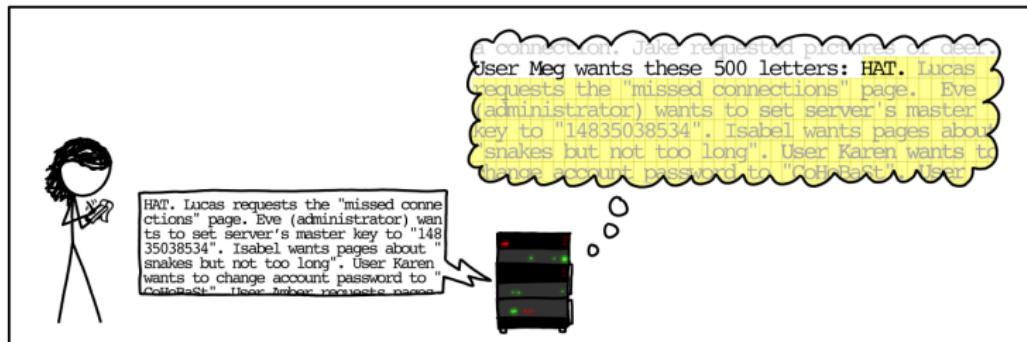
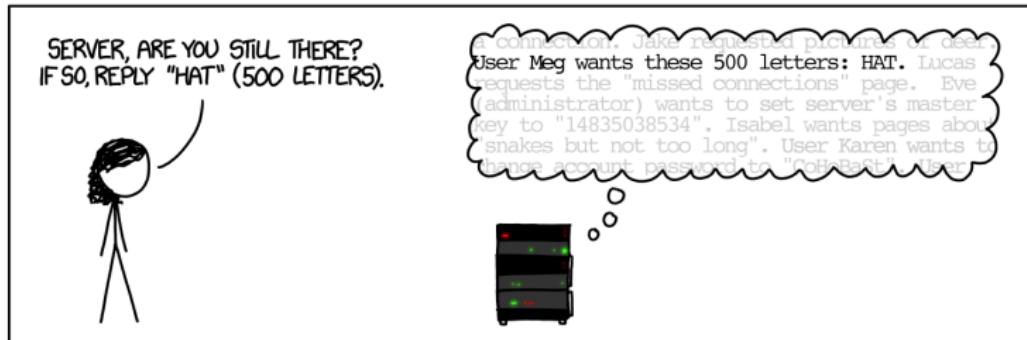
BIRD

User Olivia from London wants pages about ha bees in car why". Note: Files for IP 375.381.283.17 are in /tmp/files-3843. User Meg wants these 4 letters: BIRD. There are currently 34 connections open. User Brendan uploaded the fileelffeing (contents: 834ba962e2ce9ff89bd3b-ff89)



- Prüfung ob Payload der angegebenen Länge entspricht fehlte
- War der Payload kürzer als angegeben wurden Daten aus den darauffolgenden Speicherbereichen kopiert
- Da OpenSSL eine eigenen Speicherverwaltung implementiert waren diese Daten auch aus dem OpenSSL Kontext

# Softwarekatastrophe Heartbleed



## SW-Entwicklung in den 40er und 50er Jahren

- Teure Hardware
- Low-Level Programmierung (Assembler, fast kein OS)
- Von Experten bedient (Entwickler = Nutzer)
- numerisch-naturwissenschaftliche Probleme
- Codierung bekannter, mathematisch fundierter Algorithmen
- Viele Daten, einfache Algorithmen
- Häufig Batch-Systeme
- Fokus auf Effizienz
- Häufig "Wegwerf-Software"

### Projektmanagement

Risikomanagement  
Qualitätssicherung  
Messen/Bewerten

## SW-Entwicklung in den 60er Jahren bis heute

- Preiswerte Hardware mit viel Leistung
- Embedded Hardware die günstig ist und häufig eingesetzt wird
- Nicht-Informatiker nutzen die Software
- Vielzahl von Anwendungsbereichen
- Kritische Anwendungsbereiche wie Finanzsektor etc.
- Systeme sind komplex und interaktiv
- Software teurer als Hardware
- Lange Lebensdauer

## Softwarekrise

- Programme werden immer komplexer
- Passende Programmiersprachen, Methoden, Werkzeuge fehlen

## Folgen

- Kosten für Software steigen
- Softwareprojekte scheitern

## Lösungsansatz

SW-Entwicklung als Ingenieurstätigkeit mit definiertem Vorgehen statt künstlerischer Tätigkeit

## Dijkstra (The Humble Programmer)

*Als es noch keine Rechner gab, war auch das Programmieren noch kein Problem, als es ein paar leistungsschwache Rechner gab, war das Programmieren ein kleines Problem und nun, wo wir gigantische Rechner haben, ist das Programmieren zu einem gigantischen Problem geworden. In diesem Sinne hat die elektronische Industrie kein einziges Problem gelöst, sondern nur neue geschaffen. Sie hat das Problem geschaffen, ihre Produkte zu nutzen.*

## IEEE Definition

Software ist eine Sammlung von Computerprogrammen, Prozeduren, Regeln, zugehöriger Dokumentation und Daten

- Programme sind eine Teilmenge von Software
- SW beeinhaltet Dokumente die verschiedene Abstraktionsschichten für verschiedene Zielgruppen beschreiben

- Kommunikationsprobleme mit Anwender
- SW ist immateriell
- SW ist leicht modifizierbar, Behebung von Fehlern wird unterschätzt
- SW ist nur beobachtbar
- Anforderungen ändern sich regelmäßig
- SW altert über Umgebung, ohne zu verschleißen, dass führt zu immer neuen Erweiterungen und wachsender Komplexität
- Verhalten für Software lässt sich nur schwer beweisen
- ...

- Auslöser für Begriff “Software Engineering” war Softwarekrise von 1968
- Begriff “Software Engineering” wurde 1967 von F.L. Bauer (ehemaliger Prof. in München) im Rahmen einer “Study Group on Computer Science” der NATO geprägt.
- Software wurde erstmals als Industrieprodukt bezeichnet

## Definition IEEE

Software Engineering ist der systematische Ansatz für

- die Entwicklung,
- den Betrieb
- sowie die Wartung

von Software.

## Definition Lehrbuch Software-Technick (Balzert)

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 die Berücksichtigung z.B. von Kosten, Zeit, Qualität.

## Effiziente Entwicklung von messbar qualitativ hochwertiger Software

- Korrektheit und Zuverlässigkeit
- Robustheit
- Effizienz
- Benutzerfreundlichkeit
- Wartbarkeit und Wiederverwendbarkeit

## Qualitätsfaktoren

- Extern (für den Benutzer sichtbar)
- Intern (nur für den Entwickler sichtbar)

Der systematische Ansatz im Software Engineering wird auch als Entwicklungsprozess bezeichnet. Er beeinhaltet eine Reihe von Aktivitäten die zur Entwicklung von Software führen.

- Spezifikation
- Entwicklung
  - Entwurf
  - Implementierung
- Validierung
- Evolution
  - Weiterentwicklung
  - Betrieb

- Wenige LOC
- SW für die eigene Verwendung
- Produkt spezifiziert sich selbst
- Lösung wird direkt entwickelt
- Validierung und Korrekturen am Endprodukt
- 1 Entwickler
- Komplexität gering
- Software besteht aus wenigen Komponenten
- Wenig bis keine Dokumentation nötig
- Keine Planung und Projektstruktur nötig

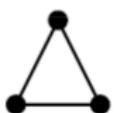
- Viele LOC
- SW für die Verwendung durch Dritte
- Klares Ziel, genaue Spezifikation erforderlich
- Lösung wird in Phasen entwickelt
- Tests in jeder Phase sind unerlässlich
- Produkt wird im Team entwickelt
- Hohe Komplexität macht Strukturierung der SW erforderlich
- Software besteht aus vielen Komponenten
- Dokumentation für den wirtschaftlichen Betrieb der SW erforderlich
- Projektstruktur zwingend erforderlich

Anzahl beteiligte Personen

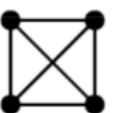
1      2



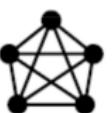
3



4



5



6



Anzahl Kommunikationspfade

0      1



Quantensprung:  
Kommunikation  
wird erforderlich

3

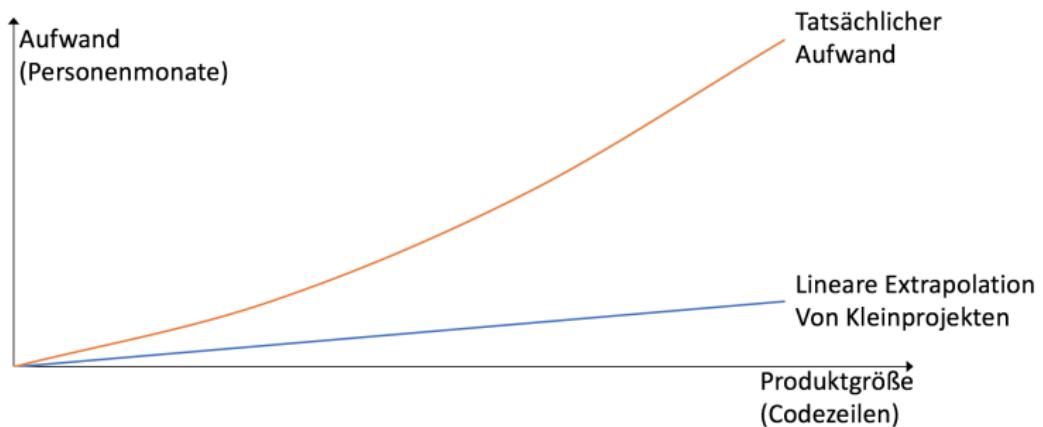


Quantensprung: Zahl der  
Kommunikationspfade über-  
steigt Zahl der Personen

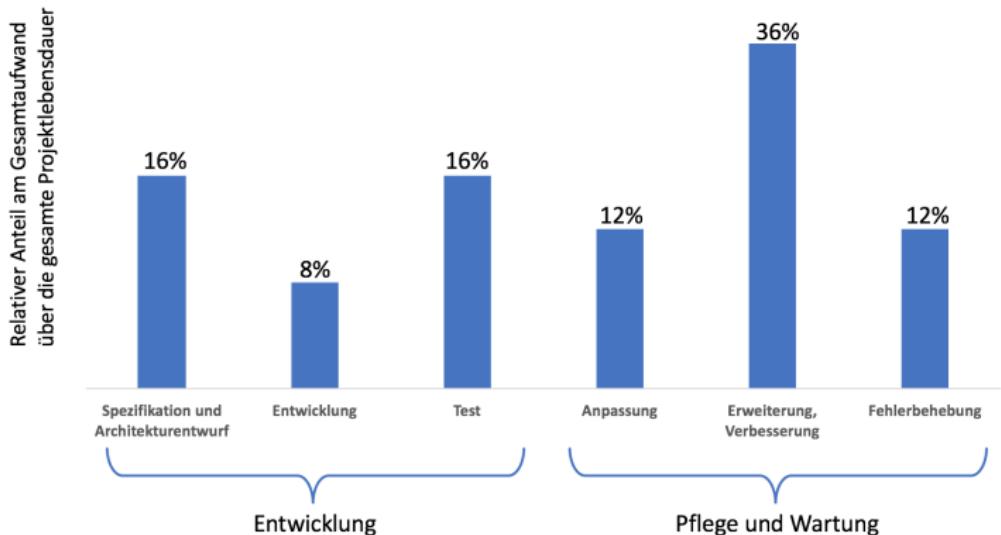
6

10

15



# Aufwände aufgeschlüsselt



Eine Person braucht zum Bau einer 2m langen Brücke 0,5 Tage. Wie lange brauchen 100 Leute für den Bau einer 2km langen Brücke?

- 1** Interpolieren Sie den Aufwand linear
- 2** Warum ist die Berechnung aus Punkt 1 eine Milchmädchenrechnung?

Eine Person braucht zum Bau einer 2m langen Brücke 0,5 Tage. Wie lange brauchen 100 Leute für den Bau einer 2km langen Brücke?

- 1 Aufwand =  $(2000\text{m} / 2\text{m} * 0.5\text{PT}) / 100 \text{ Personen} = 5 \text{ Tage}$
- 2 Mehr Kommunikation, Projekt deutlich Komplexer, Ressourcenbeschaffung, Logistik ...

## Übung 1.2

Eine Kundenbetreuerin im Firmenkundengeschäft einer Bank hat auf Grundlage eines Tabellenkalkulationsprogramms eine kleine persönliche Anwendung geschrieben, die sie bei der Überprüfung der Kredite der von ihr betreuten Firmen unterstützt. Die notwendigen Daten gibt sie jeweils von Hand ein. Der Abteilungsleiter sieht diese Anwendung zufällig, ist davon angetan und beschließt, sie allen Kundenbetreuerinnen und -betreuer zur Verfügung stellen. Die notwendigen Daten sollen jetzt automatisch aus den Datenbanken der Bank übernommen werden. Die Kundenbetreuerin gibt an, für die Entwicklung ihrer Anwendung insgesamt etwa vier Arbeitstage aufgewendet zu haben. Der Abteilungsleiter veranschlagt daher für die Übernahme und die gewünschten Änderungen einen Aufwand von einer Arbeitswoche. Als die geänderte Anwendung endlich zur Zufriedenheit aller Beteiligten läuft, sind jedoch rund acht Arbeitswochen Aufwand investiert. Der Abteilungsleiter erzählt die Geschichte einem befreundeten Berater als Beispiel, dass Informatikprojekte nie ihre Termine einhalten. Darauf meint der Berater trocken, der investierte Aufwand sei völlig realistisch und normal. Begründen Sie warum.

Software  
Engineering

Mark Keinhörster

Projektmanagement

Risikomanagement  
Qualitätssicherung  
Messen/Bewerten

- Die Kundenbetreuerin hat das Fachkonzept ihrer Tabellenkalkulationsanwendung vermutlich schon vor den angegebenen vier Tagen Bearbeitungszeit im Kopf gehabt. Die Fremdentwickler müssen dieses zumindest erst nachvollziehen.
- Die neue Anwendung ist durch die Datenbankanbindung mit den entsprechenden Schnittstellen und Zugriffsrechteproblematiken deutlich komplexer.
- Der Kommunikationsaufwand schon allein von Kundenseite (viele Berater = viele unterschiedliche Meinungen) ist erheblich
- Die neu entstandene "professionelle" Anwendung hat einen erheblich höheren Aufwand für die Validierung als eine eigengenutzte Entwicklung.
- An die Bedienbarkeit (Nutzerschnittstelle) werden bei einer "professionellen" Anwendung erheblich höhere Ansprüche gestellt.

# Vorgehensmodelle

## Definition "Prozess" nach IEEE

Eine Folge von Schritten die zu einem definierten Zweck ausgeführt werden

- Beispielsweise der Softwareentwicklungsprozess
- Um Operationen auf Daten auszuführen

## Definition "Softwareentwicklungsprozess" nach IEEE

Der Prozess bei dem die Bedürfnisse von Nutzern in ein Softwareprodukt übersetzt werden. Der Prozess beeinhaltet

- das Übersetzen der Bedürfnisse in konkrete Anforderungen,
- das Überführen der Anforderungen in einen Entwurf,
- die Implementierung des Entwurfs in Quelltext,
- das Testen des Quelltextes,
- die Installation und den Betrieb der implementierten Software.

- Softwareprozesse variieren je nach Organisation
- kein Prozess ist perfekt
- Folge: Ergebnisse unterscheiden sich situationsbedingt

Geben Sie

- 1** Beispiele für unterschiedliche Softwareprozesse
- 2** Gründe für diese Unterschiede

Warum ist es schwierig Softwareentwicklungsprozesse zu automatisieren?

## Beispiele für unterschiedliche Softwareprozesse

### 1 Planungsgetriebene Prozesse

- 1 Sequenziell
- 2 Nebenläufig
- 3 Inkrementell

### 2 Agile Prozesse

- 3 ...

## Gründe für diese Unterschiede

- 1 Detailgrad der Anforderungen
- 2 Teamstruktur
- 3 Planbarkeit des Softwareprodukts
- 4 Time-to-Market
- 5 Art der Software die entwickelt wird
- 6 Kundentyp
- 7 ...

Warum ist es schwierig Softwareentwicklungsprozesse zu automatisieren?

- 1 Anforderungen oft nicht final
- 2 Komplexe Systeme sehr schwer zu testen
- 3 Große Systeme besitzen viele Schnittstellen
- 4 ...

## Modell

- Abstrakte Abfolge von Schritten
- Dient beliebig vielen Prozessen als Grundlage für konkretes Vorgehen
- Ist ein Muster für eine bestimmte Vorgehensweise

## Prozess = Gegenstand des Modells

- Tatsächlich ausgeführte Abfolge von Schritten
- Jeder Schritt produziert ein konkretes Ergebnis
- Ist das Projekt

## Modell

- Theaterstück
- Musik-CD
- Applikation
- Klasse
- Vorgehensmodell
- Prozessmodell

## Gegenstand

- Aufführung
- Einmalige Wiedergabe
- Ausführung der Applikation
- Objekt
- Projektablauf
- Projekt (inkl. Organisation)

## Abbildungsmerkmal

- Ein Modell ist immer ein Abbild des Originals
- dass
  - Struktur (z.B. Aufbau eines Hauses),
  - Verhalten (z.B. Schiffsmodell im Strömungskanal)
  - oder Funktionsweise (z.B. Modellauto dass fährt) des Originals abbildet.

## Verkürzungsmerkmal

- Es enthält die relevanten Eigenschaften wie
  - detaillierter Skelettaufbau des Menschen für Ärzte
  - oder die Beschreibung der Proportionen des Menschen für Schneider

## Pragmatisches Merkmal

- Es ist zugeschnitten auf den Untersuchungszweck und kann damit unter bestimmten Bedingungen das Original ersetzen

Software  
Engineering

Mark Keinhörster

Projektmanagement

Risikomanagement  
Qualitätssicherung  
Messen/Bewerten

Software wird auf unterschiedliche Arten repräsentiert  
(Software Modelle)

- Spezifikation
- Entwurf
- Diagramme
- Code
- Kennzahlen
- Dokumentation

Abläufe bei der Entwicklung von Software werden durch Vorgehens-/Prozessmodelle beschrieben

# Basismodelle

## Definition Vorgehensmodell

Darstellung, die weitgehend den Softwareentwicklungsprozess beschreibt und prinzipiell auch Analysen des Prozesses gestattet. Ein Vorgehensmodell muss die Prozessschritte und die dabei verwendeten und entwickelten Resultate explizit beschreiben.

Codierung und Bugfixing finden im Wechsel mit Tests statt

- ohne Analyse
- ohne Spezifikation
- ohne Entwurf

Vorteile

- Schnelle Resultate
- Einfacher Ablauf
- Kein Aufwand für Dokumentation und Kommunikation

## Nachteile

- Schlechte Planbarkeit des Projekts
- Arbeit schwer auf mehrere Personen zu verteilen
- Es fehlen die Anforderungen da nicht erhoben
- Programmstruktur leidet durch häufige Nachbesserung
- Oft fehlende Dokumentation
- Hoher Wartungs- und Pflegeaufwand
- Wissen liegt in den Köpfen der Entwickler

Um die Nachteile zu umgehen werden zusätzliche Aktivitäten benötigt



Software  
Engineering

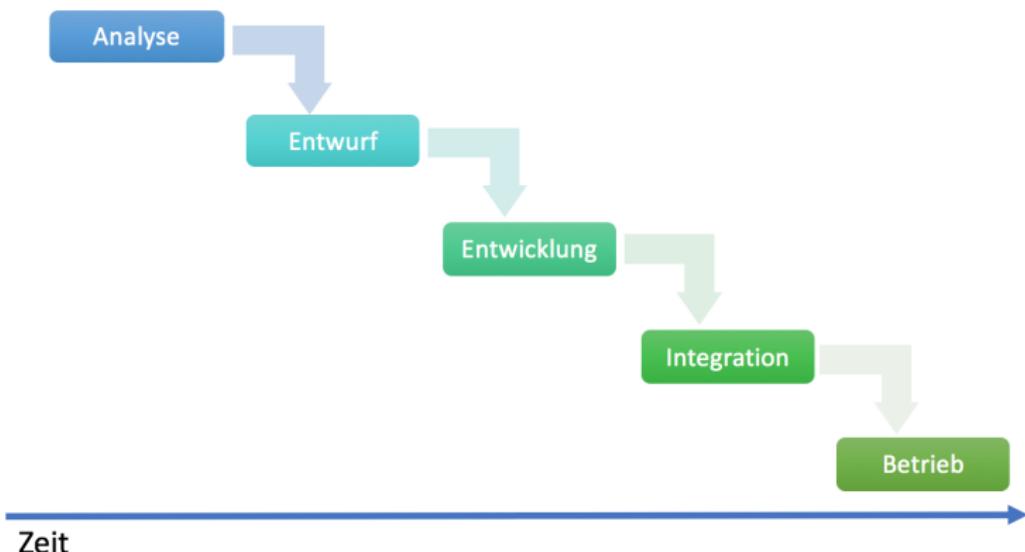
Mark Keinhörster

Projektmanagement

Risikomanagement  
Qualitätssicherung  
Messen/Bewerten

# Sequenzielles Modell

- Softwareentwicklung wird in Aktivitäten (Phasen) gegliedert
- Aktivitäten werden sequenziell durchlaufen
- Nachfolgeaktivität kann erst dann starten, wenn der Vorgänger vollständig abgeschlossen ist



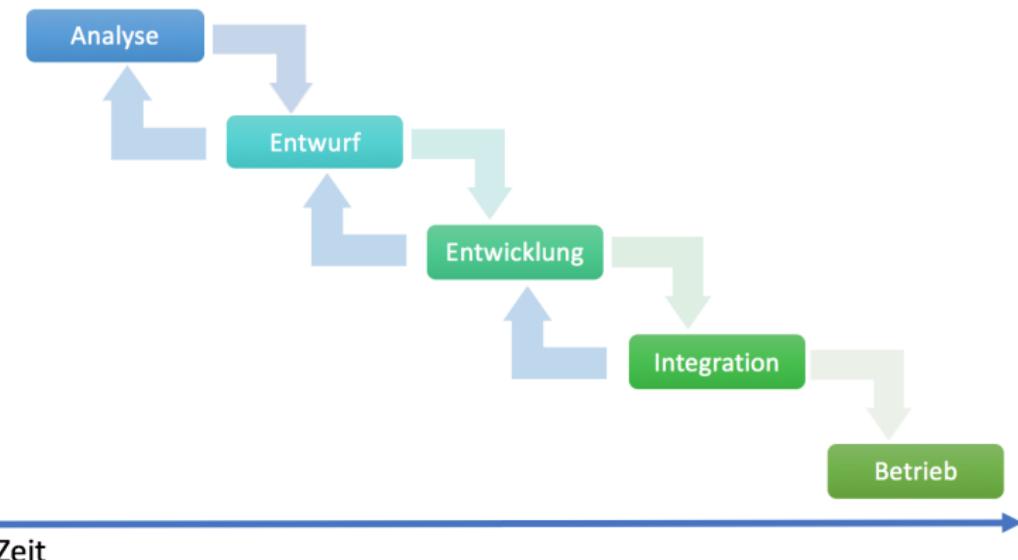
## Vorteile

- leicht verständlich
- geringer Managementaufwand

## Nachteile

- Gesamtdauer = Summe aller Aktivitäten
- Fehlende Rückkopplung zwischen den Aktivitäten
- Wird ein Problem in Folgeaktivität erkannt muss von vorn begonnen werden
- Lauffähiges Produkt erst am Ende des Projekts

- Sequenzielles Modell mit Rückkopplung
- Jede Aktivität wird vollständig ausgeführt
- Am Ende jeder Aktivität steht ein Dokumentation (Dokumentengetriebenes Modell)
- Benutzer nur in der Analyse beteiligt



Software  
Engineering

Mark Keinhörster

Projektmanagement

Risikomanagement

Qualitätsicherung

Messen/Bewerten

## Vorteile

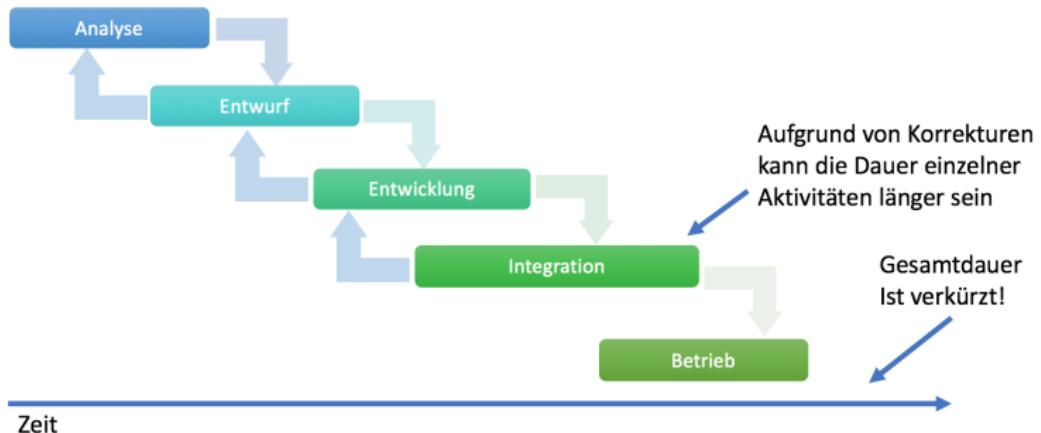
- leicht verständlich
- geringer Managementaufwand
- Aktivitäten gut dokumentiert

## Nachteile

- Es ist nicht immer sinnvoll alle Aktivitäten vollständig auszuführen
- Team ist an die Reihenfolge gebunden
- Die Dokumentation kann wichtiger werden als das eigentliche System
- Es kann nicht flexibel auf Risikofaktoren reagiert werden
- Lauffähiges Produkt erst am Ende des Projekts

# Nebenläufiges Modell

- Durch Überlappungen und Rückkopplungen soll die Gesamtzeit reduziert werden
- Nachfolger beginnen sobald Vorgänger die ersten Informationen bereitgestellt haben
- Die Teams arbeiten parallel
- Nachfolger müssen auf neue Informationen der Vorgänger reagieren



## Vorteile

- Gute Ausnutzung der Zeit
- Frühzeitige Rückmeldung möglich

## Nachteile

- Wichtige Entscheidungen können zu spät getroffen werden
- Hoher Planungs- und Personalaufwand
- Gefahr dass Nachfolger mit unzureichenden Informationen beginnen
- Es kann nicht flexibel auf Risikofaktoren reagiert werden
- Kommunikation zwischen den Teams muss aufrecht erhalten werden

## Sequenzielle Modelle

- Bisher wurde in einem Rutsch ein vollständiges Produkt entwickelt
- Es wurden vor der Implementierung alle Anforderungen im Detail erarbeitet
- Der Kunde ist nur zu Beginn involviert
- Es kann mitunter lange dauern bis der Kunde das Produkt nutzen kann

## Problem

- Zu Beginn sind oftmals nicht alle Anforderungen vorhanden
- Der Kunde sollte bereits früh Feedback geben ob das Produkt in seinem Sinne entwickelt wurde

Software  
Engineering

Mark Keinhorster

Projektmanagement

Risikomanagement  
Qualitätssicherung  
Messen/Bewerten

- Anforderungen werden vollständig erfasst und modelliert
- Produkt wird in Ausbaustufen zerlegt
- Jede Ausbaustufe realisiert einen Teil der Funktionalität
- Kunde bekommt sehr früh eine erste Version
- Erfahrungen fließen in die Folgeversionen mit ein
- Jede Version kann in eigenem Projekt entwickelt werden

Software  
Engineering

Mark Keinhörster

Projektmanagement

Risikomanagement  
Qualitätssicherung  
Messen/Bewerten

## Vorteile

- Kunde erhält früh und in kurzen Abständen produktionsreife Software
- Nicht-Funktionale Anforderungen werden frühzeitig berücksichtigt
- Durch vollständige Anforderungen kann die Applikation von Beginn an gut strukturiert werden

## Nachteile

- Vollständige Anforderungen zu Beginn führen zu einer relativ späten Auslieferung von Version 0
- Modell kann nur verwendet werden, wenn Anforderungen vollständig erfasst sind

- Einsatz wenn zu Beginn nicht alle Anforderungen erfasst werden können
- Es wird mit Kernanforderungen des Kunden angefangen
- Auf dieser Basis wird der Produktkern entwickelt
- Kunde kann früh die erste Version einsetzen
- Aus den Erfahrungen leitet der Kunde weitere Anforderungen ab
- Neue Anforderungen werden in der nächsten Version implementiert
- Zyklus aus praktischer Erprobung und Erweiterung + Verbesserung
- Software wird in Evolutionsstufen entwickelt
- Grundlage der agilen Prozessmodelle

## Vorteile

- Anforderungen müssen zu Beginn nicht vollständig vorliegen
- Kunde kann sehr früh die erste Version einsetzen und bewerten
- Erfahrungen aus Produktiveinsatz können in nächste Version einfließen
- Durch kurze Entwicklungszyklen kann kurzzeitig auf Änderungen reagiert werden

## Nachteile

- Gefahr dass bei Folgeversionen die gesamte Architektur überarbeitet werden muss
- Es besteht die Gefahr dass “evolutionär” nur ein Vorwand für mangelhafte Spezifikation ist

Software  
Engineering

Mark Keinhörster

Projektmanagement

Risikomanagement  
Qualitätssicherung  
Messen/Bewerten

Wann würden Sie evolutionäre Modelle anwenden und wann sequenzielle?

- Bei Projekten mit offenen Fragen (bzgl. Technologie, Domäne, ...)
- Bei Projekten mit unklaren oder sich ändernden Anforderungen
- Bei sehr komplexen Projekten

Entwickeln Sie eine Applikation. Für Analyse, Entwurf und Entwicklung benötigen Sie jeweils 2 Monate. Sie führen eine nebenläufige Entwicklung durch. Dabei wird jeweils 75% der Vorgängerphase überlappt. Aufgrund des Kommunikations- und Änderungsaufwands verlängern sich die Phasen Entwurf und Entwicklung um jeweils 20%. Wie viel Zeit sparen Sie durch die Nebenläufige Entwicklung ein?

Sequenziell = 2M A + 2M E + 2M C = 6 Monate

Nebenläufig = 2M A + 2.4M E (Start 0.5) + 2.4M C  
(Start 1) = ca. 3.5 Monate

Sie arbeiten nun nach dem inkrementellen Vorgehensmodell. Die Analyse benötigt 2 Monate. Sie entwickeln 2 inkrementale (V1 und V2). Für jedes Inkrement benötigen Entwurf und Entwicklung jeweils 1 Monat. Wie viel Zeit wird benötigt um V1, V2 sowie die finale Applikation fertig zu stellen? Wo liegt der Vorteil im Vergleich zur sequenziellen Entwicklung?

$$V1 = 2M A + 1M E + 1M C = 4 \text{ Monate}$$

$$V2 = 1M E + 1M C = 2 \text{ Monate}$$

$$V3 = V1 + V2 = 6 \text{ Monate}$$

## Vorteile

- Kunde kann Applikation bereits nach 4 Monaten nutzen
- Inkremeente können jeweils in eigenen Projekten realisiert werden
- ...

Entwickeln Sie eine Applikation nun nach dem evolutionären Vorgehensmodell. Für Analyse, Entwurf und Entwicklung benötigen Sie jeweils 1 Monat. Sie entwickeln 2 Versionen (V1 und V2). Wie lange dauert die Entwicklungszeit für V1 und V2 sowie für das ganze Produkt? Wo liegt der Vorteil im Vergleich zur sequenziellen Entwicklung?

$$V1 = 1M A + 1M E + 1M C = 3 \text{ Monate}$$

$$V2 = 1M A + 1M E + 1M C = 3 \text{ Monate}$$

$$V3 = V1 + V2 = 6 \text{ Monate}$$

## Vorteile

- Kunde kann Applikation bereits nach 3 Monaten nutzen
- Architektur und Code ist auf die Problemstellung abgestimmt
- ...

- Entwicklung einer Anfangsimplementierung
- Benutzer geben zu dieser (konkreten) Implementierung Feedback
- Äußerungen der Benutzer werden in neuer Version des Systems berücksichtigt
- Die Schritte 2 – 3 werden solange durchgeführt bis ein angemessenes System entstanden ist

## Prototyp

Provisorisches Softwaresystem (Modell), das erstellt wird, um Anforderungen zu klären oder zu veranschaulichen.

## Prototyping

Folge von Prototypen, die bestimmte Systemfunktionen oder -aspekte frühzeitig realisieren oder vortäuschen, so dass der Benutzer:

- den gewünschten Eindruck erhält
- sich mit dem Prototyp auseinandersetzen kann.

## Rapid Prototyping

Art von Prototyping bei dem der Fokus auf der Entwicklung von Prototypen zu Beginn des Softwareprozesses an liegt, um so möglichst von Anfang an Feedback vom Nutzer einzuholen.

Prototypen haben die folgende Verwendung:

- Klärung von Anforderungen oder Entwicklungsproblemen
- Durchführung von Experimenten und Sammlung von Erfahrungen

Dabei wird zwischen verschiedenen Arten von Prototypen unterschieden.

## Demonstrationsprototyp

- Dient der Auftragsakquisition
- Auftraggeber bekommt ersten Eindruck vom Produkt
- Schnelle Entwicklung in frühem Stadium des Entwicklungsprozesses
- Auf Standards und Best Practices bei der Entwicklung wird verzichtet
- Wegwerfprodukt

## Projektmanagement

Risikomanagement  
Qualitätssicherung  
Messen/Bewerten

## Funktionaler Prototyp

- Unterstützt die Anforderungsanalyse
- Modelliert Ausschnitte der Bedienoberfläche und/oder Teile der Funktionalität
- Kann in Architektur bereits dem Zielsystem entsprechen
- Dient dem Nachweis der technischen Machbarkeit
- Auch für Benutzerfeedback hinsichtlich Software-Ergonomie

## Projektmanagement

Risikomanagement  
Qualitätssicherung  
Messen/Bewerten

## Labormuster

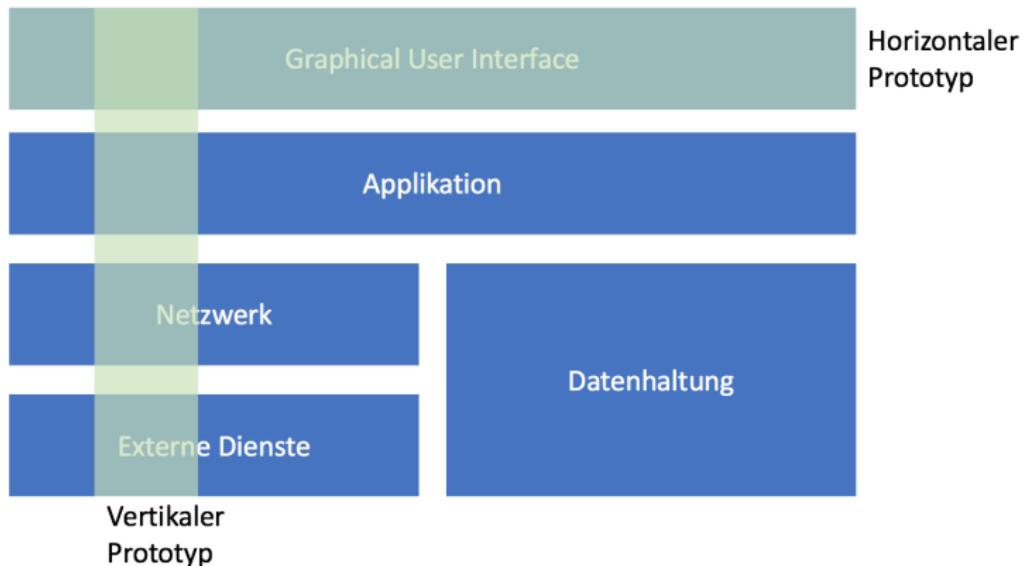
- Modelliert technische Aspekte des Zielsystems
- Experimentiersystem für Entwickler
- Für Machbarkeitsstudien
- In der Regel sind Endnutzer nicht an der Evaluation beteiligt

## Pilotsystem

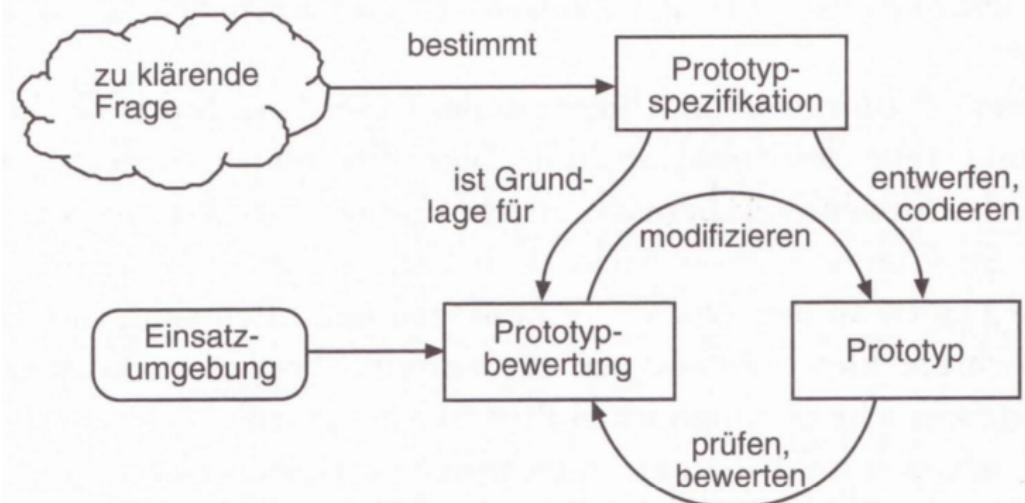
- Realisiert einen abgeschlossenen Bereich des Zielsystems
- Funktionalität und Qualität reichen mindestens für vorübergehenden Produktiveinsatz
- Wird Schrittweise erweitert

Ein Softwaresystem besteht in der Regel aus verschiedenen Komponenten/Schichten. Aus diesem Grund lassen sich Prototypen auch in horizontale oder vertikale Prototypen unterscheiden.

- Horizontale Prototypen realisieren nur eine Schicht des Systems (beispielsweise Nutzeroberfläche oder Datenbankschicht)
- Vertikale Prototypen implementieren einen funktionalen Teilbereich durch alle Schichten hindurch (End-to-End)



# Prototyping Modell - Prozess



- Für sehr interaktive Systeme
- Für kleine/mittlere oder Teile großer Systeme
- Für Systeme mit kurzer Lebensdauer
- Für Systeme, die schlecht vorab planbar sind

Prototyping wird selten als alleinstehendes Vorgehensmodell verwendet. Vielmehr findet es im Rahmen anderer Vorgehensmodelle statt.

Welche Vor- und Nachteile hat die Verwendung von Prototypen?

## Vorteile

- Reduziert das Entwicklungsrisiko
- Missverständnisse in den Anforderungen fallen frühzeitig auf
- Fehlende Anforderungen werden identifiziert wodurch die Planung optimiert wird
- Fördert die Kreativität

## Nachteile

- Die entstehenden Applikationen sind oftmals schlechter strukturiert
- Wegwerf-Prototypen werden of nachträglich als evolutionär entwickeltes System deklariert und produktiv eingesetzt

- Generisches Vorgehensmodell
- Bietet Anleitung um unter Berücksichtigung von spezifischen Risikofaktoren ein auf das Projekt zugeschnittenes Vorgehensmodell zu entwickeln
- Mit Blick auf die Risiken wird für jede Phase ein eigenes Vorgehensmodell verwendet

Für jede Phase werden 4 Schritte durchlaufen.

## Schritt 1:

- 1** Identifikation der Ziele des Teilprodukts (Leistung, Funktionalität...)
- 2** Alternative Möglichkeiten Aufzeigen um das Teilprodukt zu realisieren (Entwurf 1, Entwurf 2, Kauf, Wiederverwendung)
- 3** Randbedingungen beachten (Schnittstellen, Zeit, Budget ...)

## Schritt 2:

- 1 Alternativen evaluieren
- 2 Risiken identifizieren
- 3 Zeichnen sich Risiken ab, werden geeignete Maßnahmen ergriffen um sie zu vermeiden
- 4 Prototypen entwickeln

## Schritt 3:

- 1 Unter Berücksichtigung der verbleibenden Risiken wird geeignetes Vorgehensmodell gewählt
- 2 Entwicklung

## Schritt 4:

- 1 Nächsten Zyklus planen
- 2 Review der vorhergehenden Schritte

Software  
Engineering

Mark Keinhörster

Projektmanagement

Risikomanagement  
Qualitätssicherung  
Messen/Bewerten



- Fläche der Spirale steht für angefallene Kosten
- Winkel der Spirale zeigt Fortschritt der Entwicklung im aktuellen Durchlauf

## Vorteile

- Risiken frühzeitig berücksichtigt
- Modell ist sehr flexibel
- Andere Vorgehensmodelle können integriert werden
- Frühe Erkennung und Eliminierung von Fehlern
- Unterstützt Wiederverwendung da Alternativen berücksichtigt werden
- Anpassung der Entwicklung durch Erfahrungen die im Zyklus t-1 gewonnen wurden

## Projektmanagement

Risikomanagement  
Qualitätssicherung  
Messen/Bewerten

## Nachteile

- Ständiger Wechsel des Vorgehensmodells ist wenig Praktikabel
  - hoher Managementaufwand
  - Mitarbeiter müssen sehr flexibel seinem
  - Detailwissen und Erfahrung verschiedener Modelle sind Voraussetzung
  - Nicht für kleine bis mittlere Projekte geeignet
  - Oftmals fehlende Erfahrung im Risikomanagement
  - Geplant wird über einen Zyklus, End-2-End Planung nicht möglich

## Projektmanagement

Risikomanagement  
Qualitätssicherung  
Messen/Bewerten

Nennen Sie 5 mögliche Gründe warum Projekte  
(insbesondere Softwareprojekte) scheitern.

- Projektziel/Strategie nicht klar kommuniziert oder gar definiert
- Verwendung unausgereifte Technologien
- Kommunikationsdefizite innerhalb des Teams
- Kommunikationsdefizite gegenüber dem Kunden
- Kein klares Projektvorgehen definiert
- Fehlende Transparenz bezüglich des Projektfortschritts
- Schlechte Verteilung von Skills und Erfahrung
- Unternehmenspolitische Hürden
- ...

In einem Softwareprojekt können immer wieder Führungsprobleme auftreten. Was können mögliche Ursachen dafür sein? Betrachten Sie die Frage aus dem Blickwinkel des Projektleiters sowie aus Perspektive des Mitarbeiters.

## Sicht des Projektleiters

- Entwickler benötigen genaue Beschreibungen
- Komplexität einzelner Arbeitspakete für Nicht-Entwickler schwer bewertbar
- ...

## Sicht des Mitarbeiter

- PM bringt oftmals zu wenig Wissen aus der SW-Entwicklung mit
- Auswirkungen von ad-hoc Änderungen werden vom Management oft unterschätzt
- ...

# Monumentale Prozessmodelle

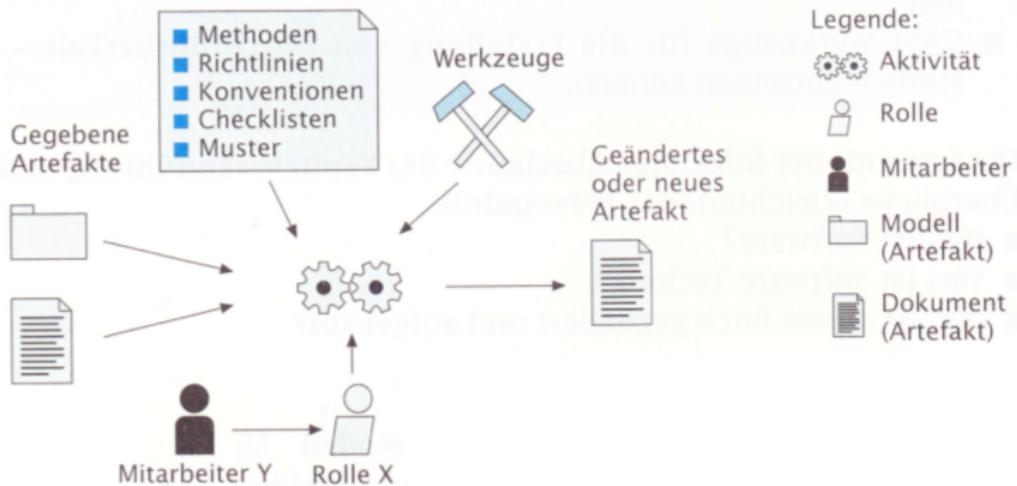
## Definition Prozessmodell

Während Vorgehensmodelle den Kern bilden, ergänzen Prozessmodelle die Vorgehensmodelle um Organisationsstrukturen für Projektmanagement, Qualitätssicherung, Dokumentation sowie Konfigurationsverwaltung.

## Monumental

- Prozessmodelle mit sehr umfangreicher Beschreibung
- Dokumentation von zentraler Bedeutung und umfangreich
- Planung ist fest vorgegeben

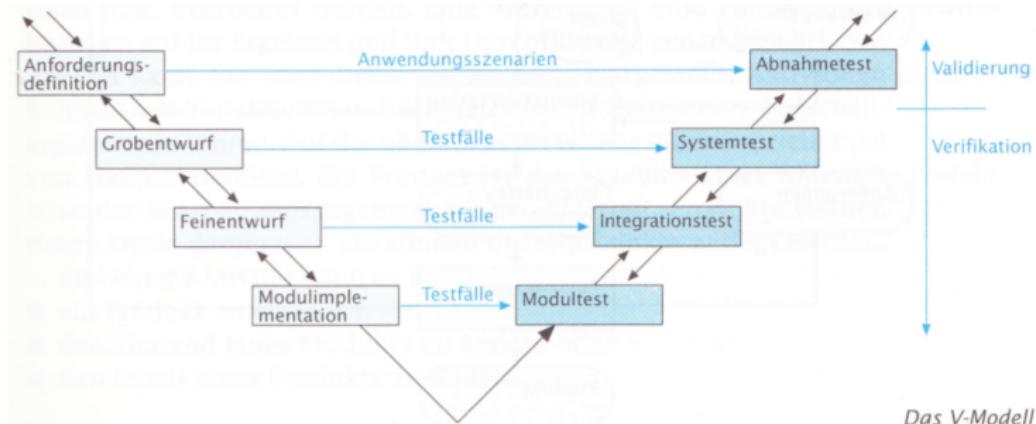
- Aktivitäten werden von Mitarbeitern durchgeführt
- Die Kenntnisse/Fähigkeiten die als Voraussetzung dienen werden durch Rollen beschrieben
- Die Durchführung wird genauer spezifiziert
  - Weitere durchzuführende Aktivität?
  - Rollenzuordnung
  - Zu verwendende Artifakte
  - Zu erstellende Artifakte
  - Zu beachtende Konventionen, Methoden, Richtlinien
  - Einzusetzende Werkzeuge



Projektmanagement

Risikomanagement  
Qualitätssicherung  
Messen/Bewerten

- Erweiterung des Wasserfallmodells
- Integration einer Qualitätssicherung
- Erzeugte Teilprodukte werden Verifikation und Validierung unterzogen
  - Verifikation: Prüfung ob das SW-Produkt mit der Spezifikation übereinstimmt ("Am I building the product right?")
  - Validierung: Eignung des Produkts für den vorgeschriebenen Einsatzzweck ("Am I building the right product?")



- Regelt “wer”, “wann”, “was” in einem Projekt zu tun hat
- Unterscheidet 4 verschiedene Projekttypen zur Abbildung von Auftraggeber und Auftragnehmer
- Charakteristische Projekteigenschaften werden berücksichtigt

Projektrolle	Auftraggeber	Auftragnehmer	Auftraggeber/Auftragnehmer	Auftraggeber/Auftragnehmer
Projekttyp	Systementwicklungsprojekt (AG)	Systementwicklungsprojekt (AN)	Systementwicklungsprojekt (AG/AN)	Einführung und Pflege eines organisationsspezifischen Vorgehensmodells
Projektgegenstand [Projektmerkmal]	HW-System	SW-System	HW- und SW-System / Eingebettetes System	Systemintegration

- Für jeden Typen bietet V-Modell angepasste Varianten an
- Variante bestimmt dann die Durchführungsstrategie



- Projektdurchführungsstrategie legt Reihenfolge der zu erstellenden Produkte und durchzuführenden Aktivitäten fest
- Grundlage für den Projektplan
- Es stehen 11 Strategien zur Auswahl

Für Entwicklungsprozess aus AN-Sicht mit hohem Realisierungsrisiko werden z.B. folgende Strategien vorgeschlagen:

- Inkrementelle Entwicklung
- Komponentenbasierte Entwicklung
- Agile Entwicklung
- ...

- Produkte sind Ergebnisse und Zwischenergebnisse eines Projekts
- 4 Produkte werden für alle Projekte verpflichtend vorgeschlagen:
  - Projekthandbuch
  - Projektplan
  - QS-Handbuch
  - Produktbibliothek
- Jedes Produkt wird durch eine Aktivität erstellt

- Zusammengehörige Produkte und Aktivitäten werden als Disziplinen bezeichnet
- Es werden 13 Disziplinen unterschieden und in die drei Bereiche Projekt, Entwicklung und Organisation unterteilt

Projekt	Entwicklung	Organisation
Planung und Steuerung	Anforderungen und Analyse	Prozessverbesserung
Berichtswesen	Systemelemente	
Konfigurations- und Änderungsmanagement	Systementwurf	
Prüfung	Logische Konzeption	
Ausschreibung / Vertragswesen	Logistikelemente	
Angebot / Vertragswesen	Systemspezifikation	

Tabelle: Aktivitäten der Disziplin “Planung und Steuerung”

## Planung und Steuerung

---

Projektfortschrittsentscheidung herbeiführen

Projekthandbuch erstellen

QS Handbuch erstellen

PM-Infrastruktur einrichten

Schätzung durchführen

Risiken managen

Projekt planen

Arbeitsauftrag vergeben

Kaufm. Projektkalkulation durchführen

- V-Modell XT wird aus Vorgehensbausteinen zusammen gesetzt
- Vorgehensbaustein fasst die für konkrete Aufgabe erforderlichen Disziplinen zusammen (z.B. alle Disziplinen die für Änderungsmanagement)
- Auch die an Produkt mitwirkenden Rollen werden durch Vorgehensbaustein festgelegt
- Den V-Modell Kern bilden folgende Bausteine:
  - Projektmanagement
  - Qualitätssicherung
  - Problem- und Änderungsmanagement
  - Konfigurationsmanagement
- Abhängig von Projektmerkmalen kommen weitere Bausteine dazu

## Vorteile

- Öffentlich zugänglich
- Werkzeuge und Dokumentation existieren
- V-Modell XT kann als Baukasten verwendet und an Projektsituation angepasst werden
- Enthält viele Produktvorlagen

## Nachteile

- Hohe Komplexität erfordert hohen Schulungsaufwand
- Generiert hohe Menge an Artefakten
- Managementaufwand ist hoch

# Agile Prozessmodelle

## Agil

- Prozessmodell mit kleinem bis mittlerem Umfang beschrieben
- Geringer Aufwand für Dokumentation
- Flexible Prozessabläufe
- Teammitglieder, Kunde, Code liegen im Fokus

## Projektmanagement

Risikomanagement  
Qualitätssicherung  
Messen/Bewerten

## Agiles Manifest

- Einzelpersonen und Interaktion > Prozesse und Werkzeuge
- Laufende Systeme > Umfassende Dokumentation
- Zusammenarbeit mit Kunden > Vertragsverhandlungen
- Reaktion auf Änderung > Verfolgen eines Plans

- Formale Aspekte der SW werden auf Minimum reduziert
  - Verzicht auf Dokumentation
  - Fokus auf lauffähigen Code
- Statt auf detailliertem Prozessmodell basiert XP auf
  - Werten
  - Prinzipien
  - Praktiken
- Testfälle (Modul- und Akzeptanztests) ersetzen Spezifikation und Entwurf

## Werte

- Einfachheit: Einfache Lösungen die leicht zu verstehen und schnell zu realisieren sind.
- Kommunikation: Persönlich und direkt, Dokumente für Informationsaustausch zweitrangig
- Feedback: Schnelle Rückmeldung über Ergebnisse an das Team
- Mut: Zur Lücke und zur Kommunikation

## Projektmanagement

Risikomanagement  
Qualitätsicherung  
Messen/Bewerten

## Prinzipien

- Unmittelbares Feedback
- Einfachheit anstreben
- Inkrementelle Veränderung
- Veränderung wollen
- Qualitätsarbeit
- Lernen lehren
- Geringe Anfangsinvestition
- Auf Sieg spielen

## Projektmanagement

Risikomanagement  
Qualitätssicherung  
Messen/Bewerten

- Gezielte Experimente
- Offene, aufrichtige Kommunikation
- Instinkte der Teammitglieder nutzen und nicht dagegen arbeiten
- Verantwortung übernehmen
- An Projektgegebenheiten anpassen
- Mit leichtem Gepäck reisen
- Ehrliches Messen

## Praktiken

- Kunde ist Teil des Teams und vor Ort
- User Stories
- Kurze Releasezyklen
- Akzeptanztests
- Pair Programming
- Testgetriebene Entwicklung
- Gemeinsame Verantwortung

## Projektmanagement

Risikomanagement  
Qualitätssicherung  
Messen/Bewerten

- Continuous Integration
- Nachhaltiges Tempo
- Offene Räume
- Planungsspiel
- Einfacher Entwurf
- Refactoring
- Metapher

## Vorteile

- Testgetrieben
- Kundennah
- Dokumentation besteht aus Code + User Stories und ist damit offen für Änderungen
- Paarweises Programmieren

## Nachteile

- Wiederverwendung oft nicht gegeben da sehr projektspezifischer Code
- Hochqualifizierte Teams gefordert
- Sehr hoher Anspruch an Kommunikation

- Managementkonzept dass nicht nur auf SW-Entwicklung beschränkt ist
- Setzt auf Selbstorganisation der Entwickler
- Ansatz ist empirisch, inkrementell und iterativ
- Basiert auf Transparenz, Überprüfung, Anpassung
- Vorgehensmodell wird durch wenige einfache Regeln definiert (easy to learn, hard to master)

Als agiler Prozess kommt Scrum mit einigen wenigen Dokumenten/Artefakten aus (nach Scrum Guide)

- Product Backlog, enthält alle Anforderungen in Form von User Stories nach Priorität absteigend geordnet
- Sprint Backlog, enthält alle Anforderungen in Form von User Stories die im Sprint umgesetzt werden
- Inkrement, fertiges Produkt als Sprintergebniss
- Definition of Done, gemeinsames Verständnis darüber was es heißt eine Story abgeschlossen zu haben

Daneben existieren außerdem Sprint Ziele, Burndown Charts, Impediment Backlog, Epic Board ...

## Scrum definiert drei Rollen

- Product Owner
  - Kennt fachliche und technische Anforderungen
  - Pflegt, erweitert und priorisiert das Product Backlog
  - Beurteilt die Sprintergebnisse
  - Entwickelt das Produkt weiter, hat eine klare Vision
- Scrum Master
  - Moderator, Coach, Serving Leader
  - Sorgt für die Einhaltung der Scrum-Regeln
  - Hilft bei methodischen Problemen
  - Beseitigt Hindernisse
  - Treibt agile Vorgehensweise über Projektgrenzen hinaus voran
- Projekt Team
  - 4-12 Teammitglieder
  - Organisiert sich selbst
  - Keine Hierarchien
  - Kennt nur Entwickler
  - Unterschiedliche Kompetenzen

Software  
Engineering

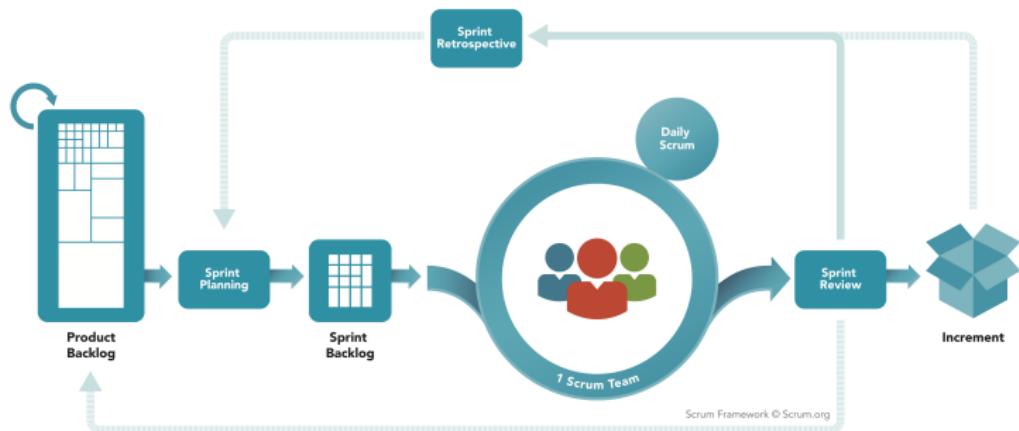
Mark Keinhörster

Projektmanagement

Risikomanagement  
Qualitätssicherung  
Messen/Bewerten

## Scrum definiert eine Reihe von Events

- Daily Scrum, 15 Minuten, 3 Fragen: Was habe ich gestern gemacht? Was mache ich heute? Was blockt mich?
- Sprint Review, Vorstellung der Sprintergebnisse
- Sprint Retrospective, Reflektion des Sprints, was lief gut, was schlecht
- Sprint Planning, erstellen des Sprint-Backlogs, was wird wie erreicht



## Vorteile

- Leichtgewichtig
- Klar definierte Regeln und Abläufe
- Fördert Kreativität durch selbstorganisierte Teams
- Leicht zu lernen

## Nachteile

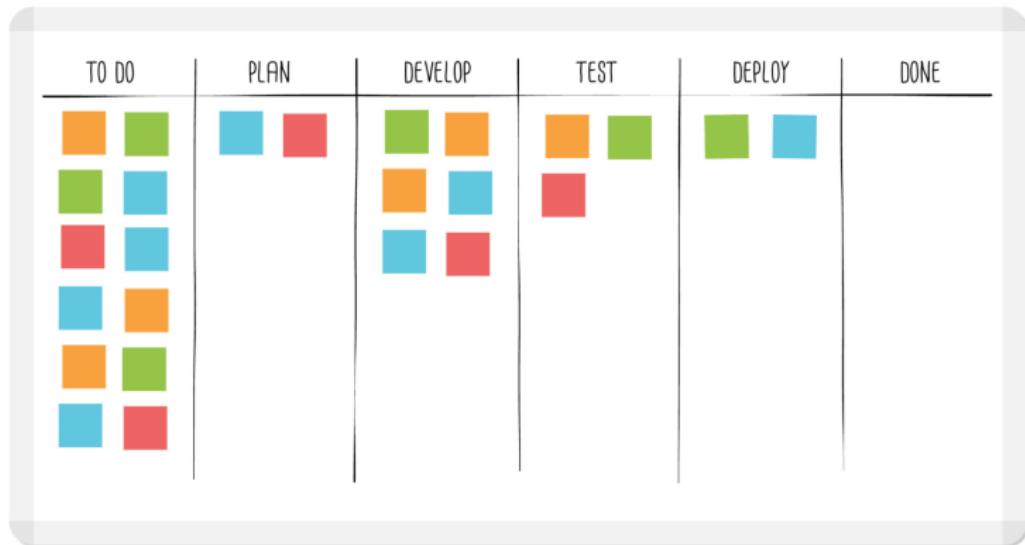
- Wiederverwendung oft nicht gegeben da sehr projektspezifischer Code
- Schwierig wenn Teams weltweit verteilt
- Regelwerk ist unflexibel
- Schwer zu meistern

## Kanban definiert sich durch vier Elemente

- Arbeit wird genommen, nicht gegeben
- Mengen werden limitiert
- Informationen werden veröffentlicht
- Abläufe werden kontinuierlich verbessert

- Kanban-Board ist zentrales Element
- Besteht aus frei definierbaren Spalten
- Visualisiert den Entwicklungsprozess
- User-Stories/Arbeitspakete werden eigenständig vom Team dem Prozess nach abgearbeitet
- Wann eine einzelnen Phase abgeschlossen ist wird in der Definition of Done festgelegt
- Jede Phase hat ein Limit das die maximale Anzahl aufgenommener Aufgaben angibt
- Limit macht Engpässe transparent

# Kanban



■ User Story

■ Defect

■ Task

■ Feature

Metriken dienen als Grundlage der kontinuierlichen Verbesserung

- Cumulative Flow Diagram = Summe aller Aufgaben pro Phase in Abhängigkeit der Zeit
- Anzahl der Tickets die Work-in-Progress sind
- Durchsatz an Tickets pro Woche
- Durchschnittliche Zeit die ein Ticket für gesamten Durchlauf benötigt

## Vorteile

- Leichtgewichtig
- Sehr leicht zu lernen
- Forciert einen ausgeglichenen Arbeitsfluss
- Probleme werden direkt visualisiert

## Nachteile

- Für große Projekte unübersichtlich
- Nicht auf Einzelprojekte/Neuprojekte anwendbar

## Projektmanagement

Risikomanagement  
Qualitätssicherung  
Messen/Bewerten

# Projektmanagement

Es gibt einige, allgemeine Erfolgskriterien die alle Projekte teilen

- Software wird innerhalb der definierten Zeit fertiggestellt
- Kosten liegen innerhalb der Budgetplanung
- Gelieferte Software entspricht der Erwartung des Kunden
- Entwicklungsteam arbeitet kohärent und effizient zusammen

Softwareprojekte haben bestimmte Charakteristika die sie von Projekten in anderen Bereichen unterscheiden

- Software ist intangibel
- Große SW-Projekte sind in Technologie, Vorgehen, Organisation einzigartig
- Prozesse und Vorgehensmodelle sind flexibel und organisationsspezifisch

- Größe der Organisation
- Kundentyp
- Komplexität der Software
- Softwaretyp
- Unternehmenskultur
- Softwareentwicklungsprozesse

- Projektplanung
- Risikomanagement
- Personalführung
- Berichterstattung
- Projektierung

# Risikomanagement

## Risiko

Ein Problem, das noch nicht eingetreten ist, aber wichtige Projektziele oder Projektergebnisse gefährdet, falls es eintritt. Ob es eintreten wird kann nicht sicher vorausgesagt werden.

- In jedem Projekt treten Probleme auf die die Projektziele gefährden
- Risikomanagement versucht mögliche Probleme frühzeitig zu identifizieren und Maßnahmen einzuleiten
- Risikomanagement ist eine kontinuierliche Aktivität und umfasst
  - Identifikation von Risiken
  - Analyse und Bewertung der Risiken
  - Planung von Gegenmaßnahmen
  - Risikoüberwachung

- Es kann zwischen Kernrisiken und projektspezifischen Risiken unterschieden werden.
- Projektspezifische Risiken können beispielsweise im Rahmen eines moderierten Workshops erhoben werden

Überlegen Sie, welche Kernrisiken für einen Großteil der Soll-Ist-Abweichungen in Softwareprojekten verantwortlich ist.

- Unklare bzw. sich laufend ändernde Projektziele und Anforderungen
- Nicht korrekter Zeitplan
- Mitarbeiterfluktuation im Projektteam
- Mangelnde Skills der Mitarbeiter
- Fehlende Unterstützung im Management
- Organisatorische Änderungen
- ...

# Qualitaetssicherung

# Messen und Bewerten