Softwaretechnik C - Softwaremanagement

Einführung und Motivation

Fachhochschule Dortmund
University of Applied Sciences

Zur Person

Fachhochschule Dortmund
University of Applied Sciences

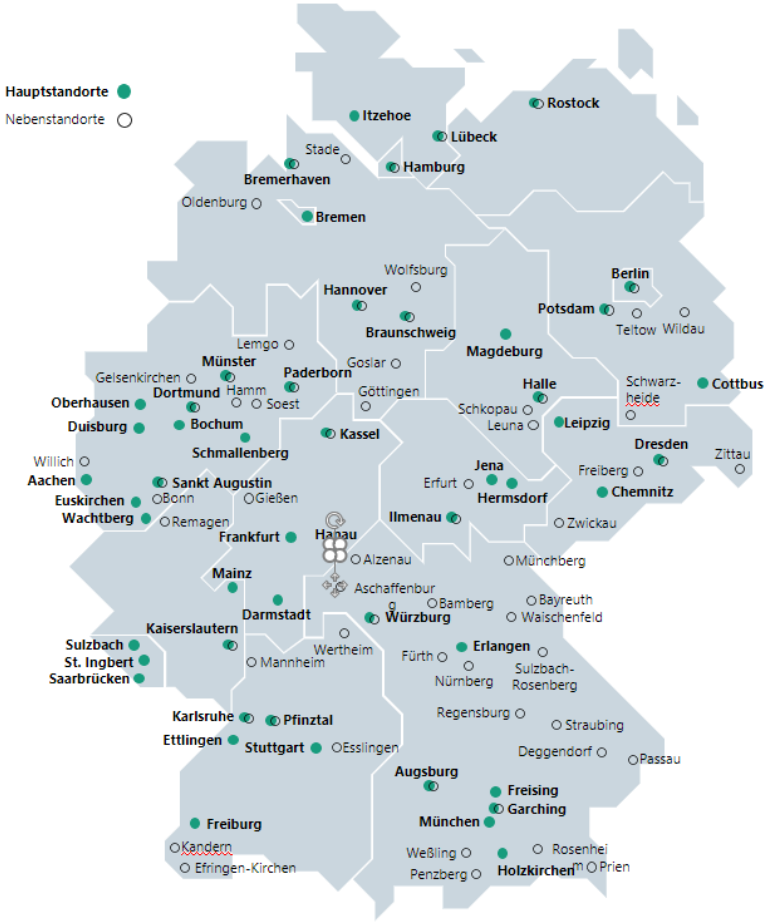
Marcel Klötgen M.Sc.

Ausbildung	2003 – 2006: B.Sc. Medizinische Informatik an der FH Dortmund 2006 – 2008: M.Sc. Medizinische Informatik an der FH Dortmund
Beruflicher Werdegang	2009 – 2018: Softwareentwickler / Entwicklungsleiter / Scrum Master bei der CompuGroup Medical AG / CGM Clinical Seit 2018: Wissenschaftlicher Mitarbeiter / Gruppenleiter bei Fraunhofer Institut für Software- und Systemtechnik ISST
Schwer-punkte	Gesundheitsdateninfrastrukturen & Datenräume im Gesundheitswesen Standards und Prozesse der primären und sekundären sowie sektorübergreifenden Verwendung von Daten im Gesundheitswesen Team- und Projektmanagement von IT-Projekten

Fraunhofer Institut für Software- und Systemtechnik ISST

- Die **Fraunhofer-Gesellschaft** mit Sitz in Deutschland ist die weltweit führende Organisation für anwendungsorientierte Forschung
  - **76** Institute und Forschungseinrichtungen
  - **30.000** Mitarbeiterinnen und Mitarbeiter
  - **2,9 Mrd. Euro** Forschungsvolumen

- Fraunhofer ISST
  - Kernkompetenz: Data Spaces
    - Standards und Architekturen für erfolgreiche Datenökosysteme
    - Technologien für Datenräume erforschen und entwickeln
  - 3 Abteilungen: Gesundheitswesen, Logistik, Datenwirtschaft
  - 118 Mitarbeitende (Stand 12/2020)



Fraunhofer Institut für Software- und Systemtechnik ISST

- Die **Fraunhofer-Gesellschaft** mit Sitz in Deutschland ist die weltweit führende Organisation für anwendungsorientierte Forschung
  - **76** Institute und Forschungseinrichtungen
  - **30.000** Mitarbeiterinnen und Mitarbeiter
  - **2,9 Mrd. Euro** Forschungsvolumen

- Fraunhofer ISST
  - Kernkompetenz: Data Spaces
    - Standards und Architekturen für erfolgreiche Datenökosysteme
    - Technologien für Datenräume erforschen und entwickeln
  - 3 Abteilungen: Gesundheitswesen, Logistik, Datenwirtschaft
  - 118 Mitarbeitende (Stand 12/2020)

Fachhochschule Dortmund
University of Applied Sciences

Kontaktdaten

Marcel Klötgen M.Sc.

Lehrveranstaltung	Softwaretechnik C (Softwaremanagement)
Email	marcel.kloetgen@fh-dortmund.de
Raum	Online via Cisco WebEx
Sprechstunde	Nach Vereinbarung

Agenda

- Organisatorisches
- Motivation
- Einführung in Vorgehens- und Prozessmodelle

- **Organisatorisches**
- Motivation
- Einführung in Vorgehens- und Prozessmodelle

Nächste Vorlesung 03.10.2023

- Aufgrund des **Feiertags am 03.10.2023** entfällt die Vorlesung in Präsenz.
- Stattdessen wird eine **Aufzeichnung (mp4) der LV in ILIAS** bereitgestellt.
- Das Praktikum am 04.10.2023 wird dann – neben der Zeit zur Aufgabenbearbeitung – zusätzlich zur **Diskussion und Beantwortung Ihrer Fragen** genutzt.

Fachhochschule Dortmund
University of Applied Sciences

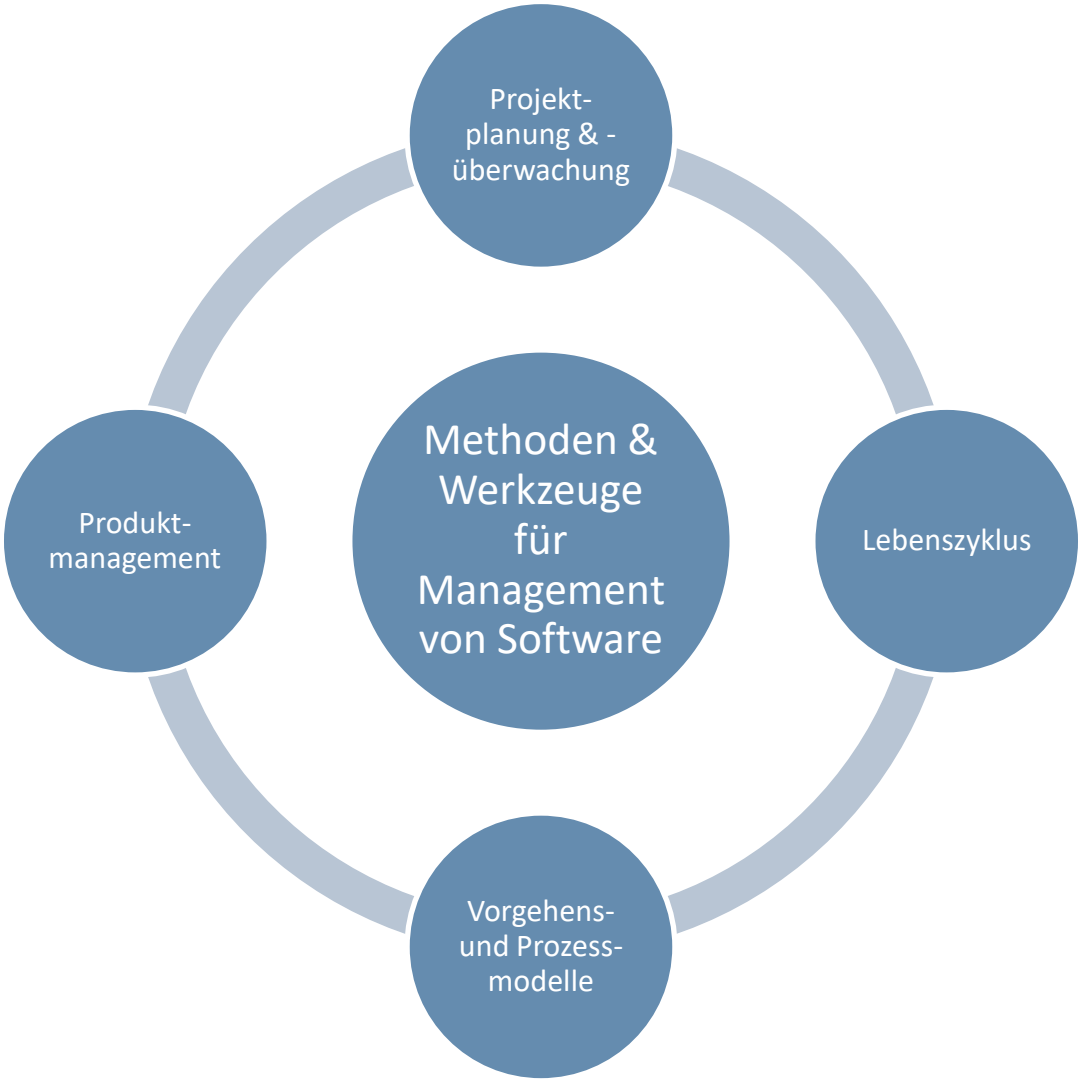
Lehrveranstaltung

Name der Lehrveranstaltung	**Softwaretechnik C (Softwaremanagement)**
Modulnummer	45261
Zielgruppe	Informatik-Studierende (5. Semester)
Vorlesung	Dienstags, 16:00 Uhr – 17:30 Uhr Raum A.E.02
Praktikum	Mittwochs, 16:00 Uhr – 18:30 Uhr Cisco WebEx // Raum C.1.30 **Zugangsdaten sind im ILIAS-Kurs zu finden**
ILIAS-Kurs	Softwaretechnik C Softwaremanagement
ILIAS-Kurspasswort	**swtc23_24**

Kultur der LV

- **Aktive Mitarbeit** erwünscht
  - Diskurs
  - Fragen
  - Wortmeldungen
- Teilnahme am **Praktikum für praktische Anwendung der Lehrinhalte**
- **Vor- und Nachbereitung** der LV
- Nachfragen, Ergänzungen oder Vorschläge gern auch per Email oder im Gespräch

Ziele der LV


Projekt-planung & -überwachung
Produkt-management
Methoden & Werkzeuge für Management von Software
Lebenszyklus
Vorgehens- und Prozess-modelle

Inhalte der LV (2/2)

Konkrete Inhaltssegmente der LV:

- Vorgehens- und Prozessmodelle der Softwaretechnik
- Anforderungsmanagement
- Qualitätsmanagement
- Konfigurationsmanagement
- Risikomanagement
  - Risikomanagementprozess und Planung von Gegenmaßnahmen
- Produktmanagement
  - Anforderungen, Änderungen, Lieferantenvereinbarungen
- Prozessverbesserung

Inhalte des Praktikums

- Erstellung von und Arbeit mit **thematisch relevanten Inhalten**
- Übungen
- Methoden

Prüfungsleistung

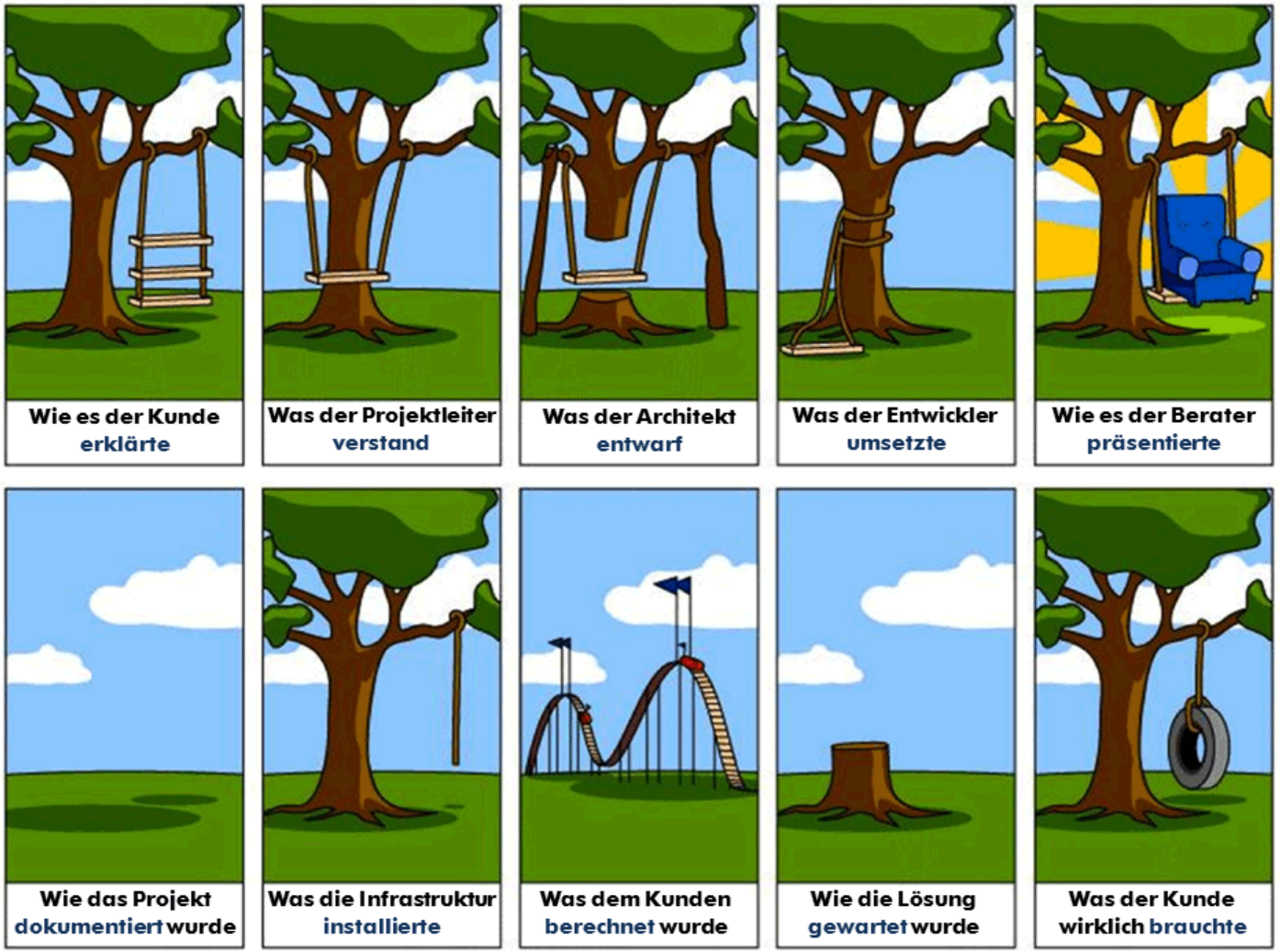
- Klausur
  - Details werden noch bekanntgegeben
- Präsentation der Praktikumsergebnisse
  - Bonuspunkte für Klausur

Literatur

- Balzert, H. (2008): Lehrbuch der Softwaretechnik, Softwaremanagement, 2. Auflage, Heidelberg: Spektrum Akademischer Verlag.
- Sommerville, I. (2012): Software Engineering, 9. aktualisierte Auflage, München: Pearson Studium.
- Spitczok, N., Vollmer, G., Weber-Schäfer, U. (2014): Pragmatisches IT-Projektmanagement, 2. akt. u. überarb. Auflage, Heidelberg: d.punkt-Verlag
- Winkelhofer, G. (2005): Management- und Projektmethoden, 3. Auflage, Berlin, Heidelberg: Springer-Verlag.

- Organisatorisches
- **Motivation**
- Einführung in Vorgehens- und Prozessmodelle

Warum Softwaremanagement?



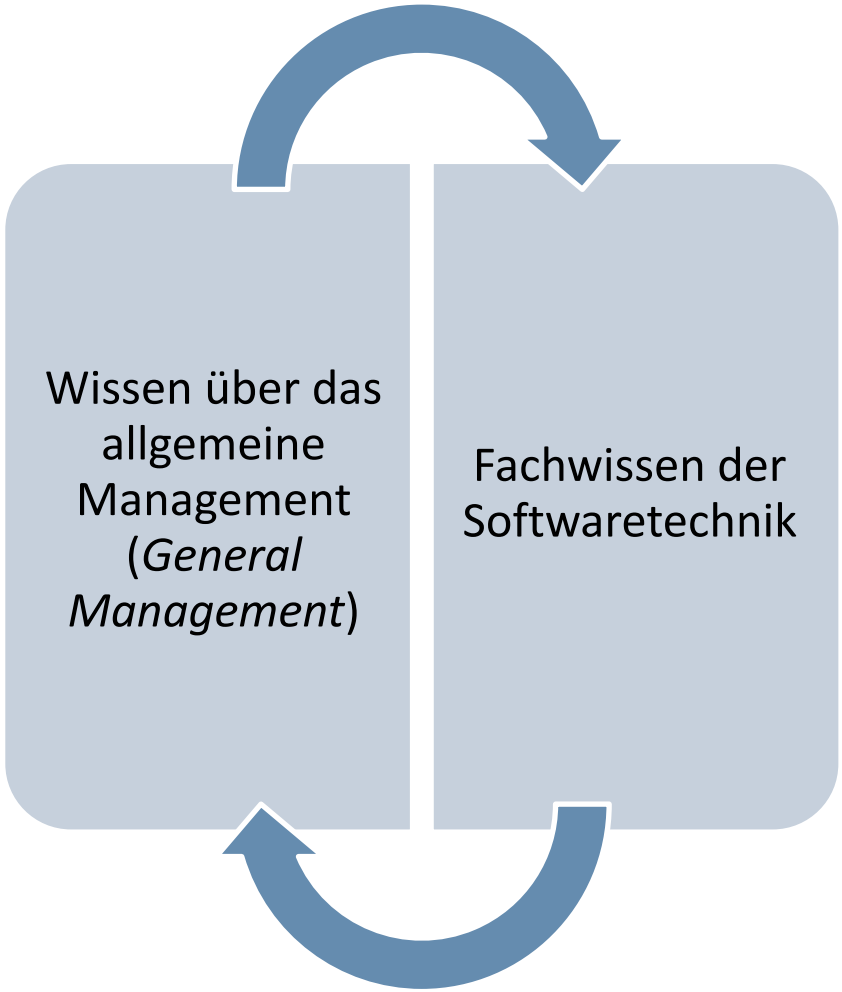
Motivation Softwaremanagement (1/3)

- Bei den Prozessen und Aktivitäten des Softwaremanagements handelt es sich oftmals um **begleitende Aktivitäten in einem Softwareentwicklungsprojekt**
- Die Ergebnisse dieser Aktivitäten sind in der Regel selbst **kein integraler Bestandteil des Endprodukts**
- Dennoch sind diese Aktivitäten eminent wichtig
- Das wird deutlich durch die **große Anzahl fehlgeschlagener Softwareprojekte**, bei denen diese Aktivitäten vernachlässigt wurden

Motivation Softwaremanagement (2/3)

- Software besitzt **spezielle Eigenschaften**, die das Management von Softwareprojekten erschweren
- Viele **Manager**, die aus **anderen Bereichen** in die Software-Branche gewechselt sind, sind an der Nichtberücksichtigung dieser Eigenschaften gescheitert
- Viele **Berufsanfänger**, die als Software-Ingenieure ihre Laufbahn beginnen, kommen unter Umständen schnell und oft ohne spezifische Vorbereitung in die Situation, ein Projektteam zu führen
- **Ziel** ist es somit, Sie als Studierende eines Informatik-Studiengangs möglichst praxisorientiert auf die vielfältigen Aufgaben des **Softwaremanagements** vorzubereiten

Motivation Softwaremanagement (3/3)



Wissensdomänen des Softwaremanagements

für

- Softwareprojekte
- softwareintensive IT-Bereiche und IT-Projekte

Fallbeispiele Softwareprojekte (1/2)

Studie über große Softwareprojekte (USA, 1979):

- 75% der Projektergebnisse **nicht eingesetzt**
- 19% der Projektergebnisse **umfassend überarbeitet**
- **Nur 6%** der Projektergebnisse **eingesetzt wie ausgeliefert** (!)

IBM Consulting-Studie über große Softwareprojekte (1994):

- 55% der Softwareprojekte **überschreiten das Budget**
- 68% der Softwareprojekte **überschreiten den Zeitrahmen**
- 88% der Softwareprojekte müssen (komplett) **überarbeitet** werden

Fallbeispiele Softwareprojekte (2/2)

MODERN RESOLUTION FOR ALL PROJECTS

	2011	2012	2013	2014	2015
SUCCESSFUL	29%	27%	31%	28%	29%
CHALLENGED	49%	56%	50%	55%	52%
FAILED	22%	17%	19%	17%	19%

The Modern Resolution (OnTime, OnBudget, with a satisfactory result) of all software projects from FY2011–2015 within the new CHAOS database. Please note that for the rest of this report CHAOS Resolution will refer to the Modern Resolution definition not the Traditional Resolution definition.

Quelle: The Standish Group International, Inc [1]. CHAOS REPORT 2015. Document on the internet: https://standishgroup.com/sample_research_files/CHAOSReport2015-Final.pdf. Last viewed: 24/10/2020.

[1] The Standish Group International, Inc. or Standish Group is an independent international IT research advisory firm founded in 1985

Report wurde wegen fehlender Transparenz, nicht offen gelegter Methodik und anderer Mängel kritisiert

Fallbeispiele Softwareprojekte
Suboptimale Softwareprojekte (1/7)

Fachhochschule Dortmund
University of Applied Sciences

2000: Entwicklung der „**Cheops**“-Abrechnungssoftware bei der RWE wird eingestellt

- Objektorientiertes Softwareentwicklungsprojekt in Java
- 4 Jahre Planung und Implementierung
- Verursachte Kosten bei der Einstellung größer als € 50.000.000
- Gründe:
  - "Herkules-Aufgabe" des **Projekt-Managements**
  - unzureichendes **Anforderungsmanagement**
  - Schlechte bzw. fehlende **Kommunikation**

Fallbeispiele Softwareprojekte
Suboptimale Softwareprojekte (2/7)

Fachhochschule Dortmund
University of Applied Sciences

Quelle: toll-collect.de; eurotransport.de

Fallbeispiele Softwareprojekte
Suboptimale Softwareprojekte (2/7)

Fachhochschule Dortmund
University of Applied Sciences

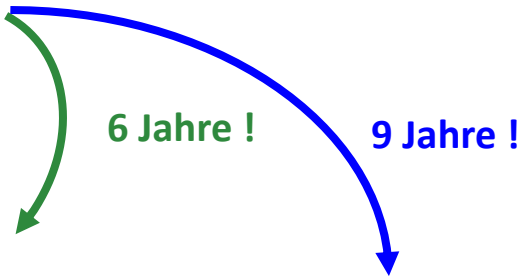
2003: ursprünglicher Starttermin von „**Toll Collect**“

- Ziel: Elektronisches und satellitengestütztes Mautsystem für das Autobahn-Netz in Deutschland
- Tatsächlicher Start mit Großteil der Funktionalität: 1. Januar 2006
- 1,6 Mrd. € (€ 1.600.000.000) Vertragsstrafen
- 3,5 Mrd. € (€ 3.500.000.000) Einnahmeausfälle
- Gründe:
  - Massive **Planungsfehler** und
  - Schlechtes **Management** verursacht den bis dato weltweit größten Fehlschlag im Bereich industrieller Softwareentwicklung

Fallbeispiele Softwareprojekte
Suboptimale Softwareprojekte (3/7)

2003: „**INPOL-neu**“ bundesweites Polizei-Informationssystem beim BKA

- 1992: Beschluss der Neuentwicklung
- 1995: Technisches Grobkonzept fertig
- 1996: 130 Projektbeteiligte
- 1998: Programmierung **wird gestartet**



- 2001: Erster Probelauf: INPOL-neu stürzt **nach wenigen Minuten** ab.
- KPMG-Gutachten: Softwaresystem zu **unausgereift** und **komplex**

Fallbeispiele Softwareprojekte
Suboptimale Softwareprojekte (4/7)

2003: „**INPOL-neu**“ bundesweites Polizei-Informationssystem beim BKA

- 2001: Projekt soll erst eingestellt werden; dann wird neue Version auf Basis von POLAS (wird in Hamburg und Hessen eingesetzt) entworfen
- 2003: nach elf Jahren wird INPOL-neu in Betrieb genommen
- Seit 2006 ist die Anwendung INPOL-neu 5.0 in Betrieb, entwickelt und betrieben wird die Software vom BKA
  - Mangelhafte **Abstimmung** zwischen Bund/Ländern kostete rund € 50 Mio.
  - Laut Focus sollen (unter Berufung auf BKA-Insider) mindestens € 70 Mio. zu den ursprünglich geplanten Kosten von € 50 Mio. hinzugekommen sein
  - Das entspricht einer **Kostensteigerung** um 140% !

Fallbeispiele Softwareprojekte
Suboptimale Softwareprojekte (5/7)

2005: Softwareentwicklungsprojekt „**Fiscus**“ wird eingestellt

- Ziel: einheitliche Software für die rund 650 Finanzämter Deutschlands
- Entwicklungskosten bis zur Einstellung > 900.000.000 Euro
- Entwicklungszeit bis zur Einstellung > 12 Jahre
- Fazit:
  - **Kein Ergebnis**
  - Extrem teurer Fehlschlag

Fallbeispiele Softwareprojekte
Suboptimale Softwareprojekte (6/7)

2009: „**Herkules**“-Projekt zur Standardisierung und Modernisierung der nichtmilitärischen IT-Infrastruktur der Bundeswehr

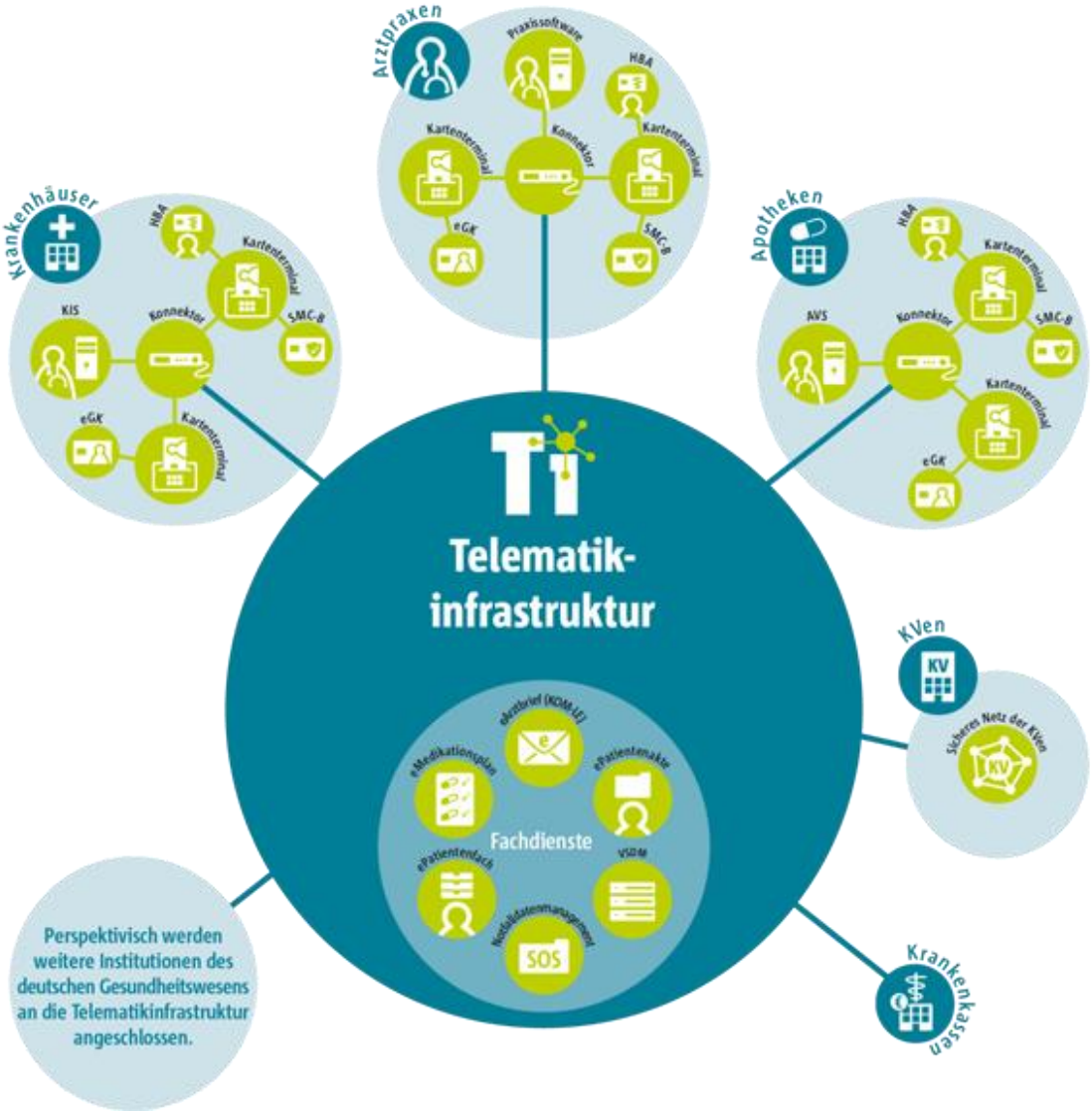
- Kostensprung von initial € 700 Mio. auf insgesamt € 7,8 Mrd.
- Das entspricht einer 1114% Kostensteigerung
- Gründe, u.a.:
  - die fehlende bzw. mangelhafte LAN-Verkabelung in zahlreichen Standorten war nicht berücksichtigt worden!

Fallbeispiele Softwareprojekte
Suboptimale Softwareprojekte (7/7)

Fachhochschule Dortmund
University of Applied Sciences

Quelle: heise.de

Fallbeispiele Softwareprojekte
Suboptimale Softwareprojekte (7/7)



Quelle: heise.de

Fallbeispiele Softwareprojekte
Suboptimale Softwareprojekte (7/7)

10/2011: Start Rollout der **elektronischen Gesundheitskarte (eGK)**

- Sollte eigentlich ab dem 01.01.2006 die Krankenversicherungskarte in der GKV ersetzen
- Ziele:
  - eGK als Baustein in verschiedenen Anwendungsfällen mit Bezug zu Datenschutz und –sicherheit (**Identifikation, Autorisierung**)
  - Funktionsbaustein in verschiedenen Anwendungsfällen über sicheres Netz der Telematikinfrastruktur, z.B. elektronische Patientenakte (ePA), eRezept
  - Potenziell mögliche Speicherung kleiner Datensätze, z.B. Notfalldaten
- Massive Kritik von Datenschützern
- 113. Deutsche Ärztetag befürwortete 2010 das eGK-Projekt aufzugeben
- Mittlerweile prognostizierte Kosten > € 14 Mrd.
- anhaltender Diskurs

Quelle: jooinn.com

Hauptgründe für Projektabbrüche (1/3)

- Vielzahl von Untersuchungen zum Erfolg bzw. Misserfolg von Softwareentwicklungsprojekten
- Vielzitiert: Chaos-Report der Standish Group
  - Report wurde wegen fehlender Transparenz, nicht offen gelegter Methodik und anderer Mängel kritisiert
- Neuere Untersuchungen kommen zu folgenden Ergebnissen:
  - In 2005 und 2007 wurden zwei internationale Befragungen durchgeführt
  - 50% aller Projekte dauerte bis zu 9 Monaten
  - Zahl der involvierten Softwareentwickler schwankte zwischen 3 und 10
  - 2005 wurden 16% und 2007 12% der Projekte abgebrochen, bevor irgendetwas ausgeliefert wurde
  - Keine signifikanten Auswirkungen von **Projektdauer** oder **Anzahl der Projektbeteiligten**

Hauptgründe für Projektabbrüche (2/3)

- **Vier Hauptgründe** für erfolglose Softwareentwicklungsprojekte
  - **Änderungen der Anforderungen** und des Umfangs (33 Prozent)
  - Mangelnde Einbindung des **höheren Managements** (33 Prozent)
  - Fehlende **Projektmanagement-Fähigkeiten** (28 Prozent)
  - Engpass im **Budget** (28 Prozent)
- Zwischen 48% (2005) und 55% (2007) der ausgelieferten Projektergebnisse waren erfolgreich
- Zwischen 17% und 22% der ausgelieferten Projektergebnisse waren nicht erfolgreich

Exkurs: Stakeholder

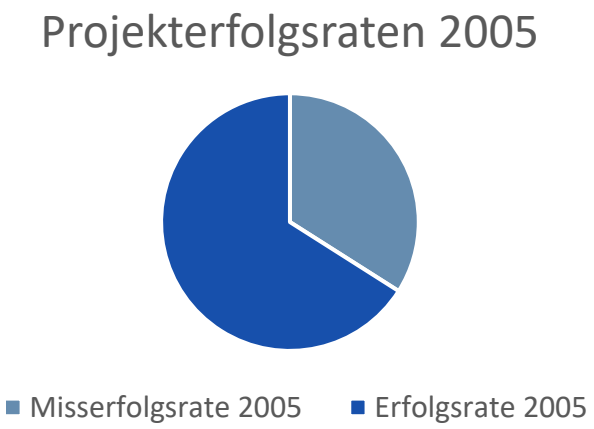
- Definition aus Gabler Wirtschaftslexikon
  (Anspruchsgruppen • Definition | Gabler Wirtschaftslexikon)

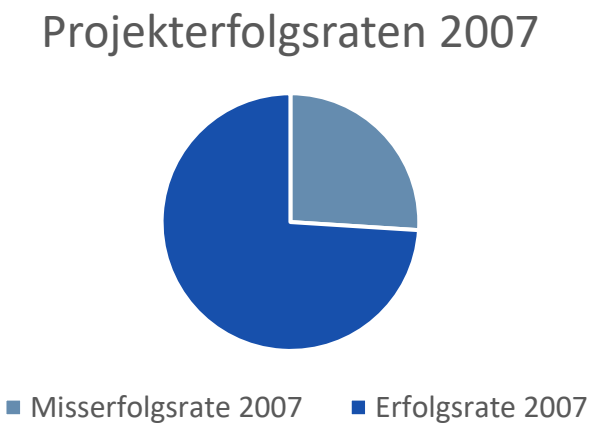
  - *Stakeholders;* Anspruchsgruppen sind alle **internen** und **externen Personengruppen**, die von den unternehmerischen Tätigkeiten gegenwärtig oder in Zukunft **direkt** oder **indirekt betroffen** sind. Gemäß Stakeholder-Ansatz wird ihnen - zusätzlich zu den Eigentümern (Shareholders) - das Recht zugesprochen, ihre Interessen gegenüber der Unternehmung geltend zu machen. Eine erfolgreiche Unternehmungsführung muss die Interessen aller Anspruchsgruppen bei ihren Entscheidungen berücksichtigen (Social Responsiveness).

Hauptgründe für Projektabbrüche (3/3)

- Kombiniert man die komplett abgebrochenen mit den nicht erfolgreichen Projekten, dann ergeben sich
  - für 2005: **34% Misserfolgsrate**
  - für 2007: **26% Misserfolgsrate**

- **Sehr hohe Misserfolgsrate** für eine angewandte Ingenieursdisziplin





Ursachenforschung (1/4)

Zahlreiche mittlere bis große IT- und Softwareentwicklungsprojekte:

- **Überschreiten** deutlich die ursprünglich veranschlagten **Kosten**
- **Überschreiten** deutlich die ursprünglich veranschlagten **Auslieferungstermine**
- Weisen oftmals erhebliche **qualitative Mängel** in den Ergebnissen auf
  - Das setzt aber immerhin voraus, dass zumindest substanzielle Ergebnisse vorliegen
- Werden nach Jahren intensiver Planungs- und Entwicklungszeit **komplett eingestellt**

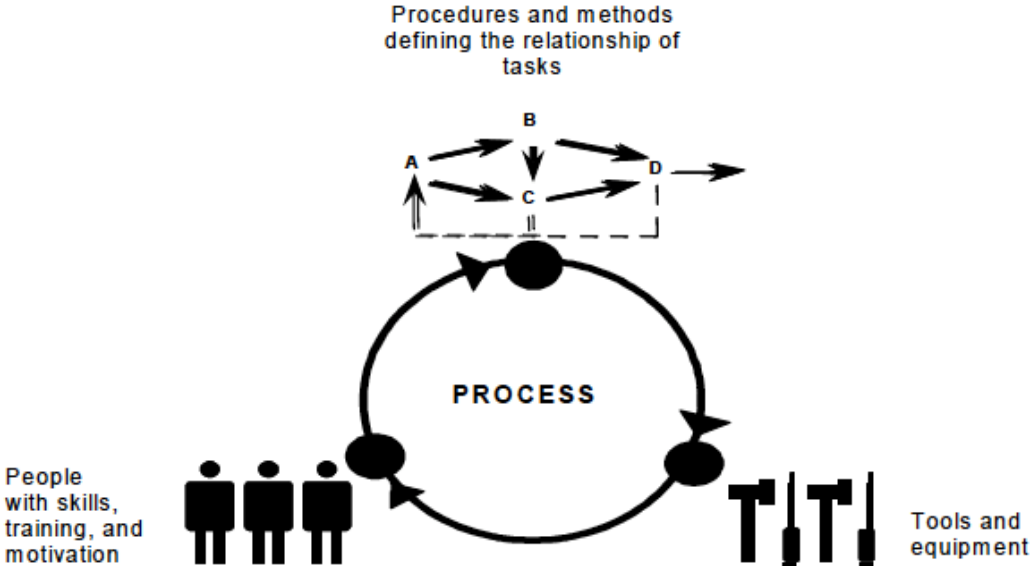
Ursachenforschung (2/4)

- Die Entwicklung eines IT-/Softwaresystems ist unter allen großen Ingenieurdisziplinen vermutlich der komplexeste Prozess
- **Hohe Komplexität** => große Wahrscheinlichkeit, dass SW-Projekt schwer überschaubar wird => hohe Kosten, lange Laufzeiten
- Mangelhaftes **Projektmanagement**
- Unzureichendes **Anforderungsmanagement**
- Fehlendes bzw. rudimentäres **Risikomanagement**
- Verbesserungswürdiges **Konfigurations-, Versions- und Release-Management**
- Keine Aktivitäten zur kontinuierlichen **Prozessverbesserung**

Ursachenforschung (3/4)

- **Unzureichende Erfahrungen** bzw. **mangelhaftes Wissen über die eingesetzten Technologien**
- Fehlende bzw. **gestörte Kommunikation** zwischen den Projektbeteiligten und/oder zwischen Auftraggeber und Auftragnehmer
- Software ist ein immaterielles Produkt
  - => schwierige **Projekt- und Ressourcenplanung**
  - => sehr schwierige **Projektfortschrittskontrolle**
  - => sehr schwieriges **Aufwands- und Kosten-Controlling**

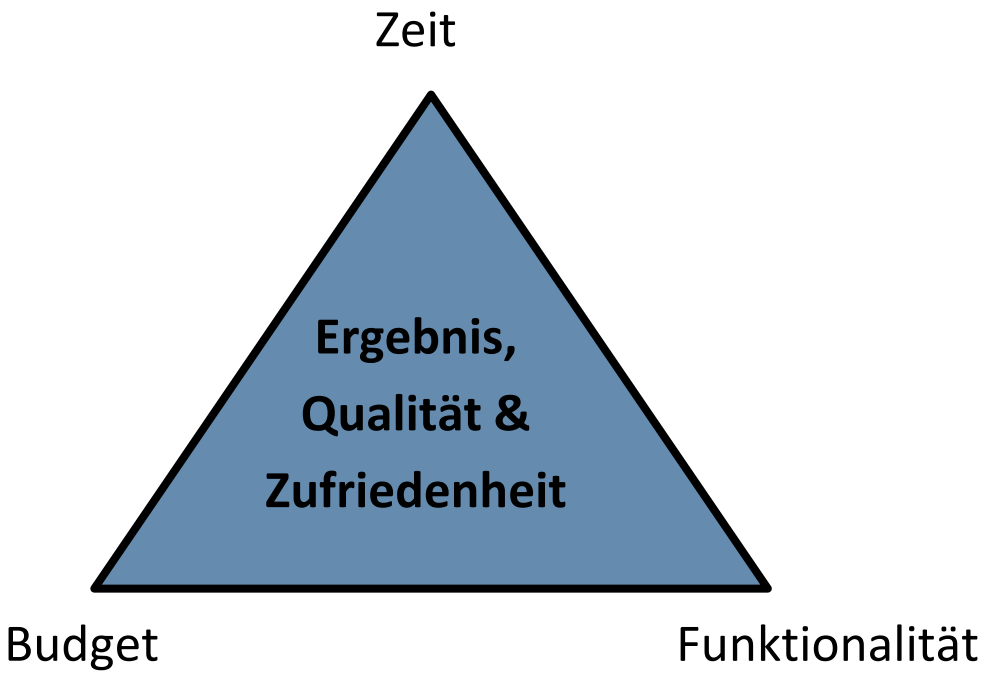
Ursachenforschung (4/4)



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

Ziele und möglicher Lösungsansatz (1/3)

- Erhöhung der Erfolgswahrscheinlichkeit von Softwareentwicklungsprojekten
- Einhaltung der Ergebnisqualität, des Aufwands, der Kosten und der Termine



Ziele und möglicher Lösungsansatz (2/3)

Lösungsansätze:

- **Reduktion der Komplexität** des Projekts => Bessere Planbarkeit eines Projekts
- Effektives **Anforderungsmanagement**
- Anwendungsspezifische Softwareentwicklung
- Effizientes **Konfigurationsmanagement**
- **Versions- und Release-Management**
- Praktikables und effektives **Risikomanagement**
- Effizientes Aufwands- und Kosten-Controlling
- Marktgerechtes **Produktmanagement**

Ziele und möglicher Lösungsansatz (3/3)

Lösungsbaustein:

- **Vorgehens- und Prozessmodelle** entwickeln und einsetzen, um Software:
  - in einem festgelegten organisatorischen Rahmen zu entwickeln und nicht ad hoc vorzugehen
  - Um die grundsätzlichen Ziele in Softwareprojekten (siehe vorhergehende Folie) weitgehend zu erreichen

- Organisatorisches
- Motivation
- **Einführung in Vorgehens- und Prozessmodelle**

Definitionen/Ziele von Vorgehens- und Prozessmodellen

- Übertragen die methodische Vorgehensweise für Fertigungsprozesse aus den Ingenieurwissenschaften auf die Softwareentwicklung
- Beschreiben die zeitlich-sachlogische Abfolge der durchzuführenden Aktivitäten in idealisierter Form und definieren
  - Die **Reihenfolge** der Entwicklungsaktivitäten
  - Die verantwortlichen **Rollen** und notwendigen Mitarbeiterqualifikationen
  - Die zu erzeugenden **Teilprodukte** bzw. Ergebnisdokumente (sog. **Artefakte**) einschließlich Layout und Inhalt
  - Die **Fertigstellungskriterien**
  - Die **Verantwortlichkeiten** und Kompetenzen
  - Die anzuwendende **Standards**, **Richtlinien**, Methoden und Werkzeuge

Beispielmodell SWT: Code & Fix

- Historie
  - Unstrukturiertes Vorgehen zur Entwicklung von Software
  - Aus der Anfangszeit der Rechentechnik
  - Oftmals nur 1 Entwickler*in

Beispielmodell SWT: Code & Fix

1. Schreibe ein Programm
2. Finde und behebe die Fehler in dem Programm

- Nachteile dieses Vorgehens:
  - Nach Behebung von Fehlern wurde das Programm so umstrukturiert, dass weitere Fehlerbehebungen immer teurer und aufwändiger wurden
    - Dies führt zur Erkenntnis, dass eine Entwurfsphase vor der Programmierung benötigt wird
  - Selbst gut entworfene Software wurde vom Endbenutzer oft nicht akzeptiert
    - Dies führte zur Erkenntnis, dass eine Definitionsphase vor dem Entwurf benötigt wird.
  - Fehler waren schwierig zu finden, da Tests schlecht vorbereitet und Änderungen unzureichend durchgeführt wurden.
    - Dies führte zu einer separaten Testphase

Einführung in die Vorgehens- und Prozessmodelle der Softwareentwicklung

- Klassische Vorgehensmodelle
  - Wasserfallmodell
  - Nebenläufiges Modell
  - Spiralmodell
  - Prototyping
- Monumentale Vorgehensmodelle
  - V-Modell XT
  - Rational Unified Process (RUP)
- Agile Prozesse
  - Scrum
  - Kanban (IT-Kanban)
  - Extreme Programming (XP)
  - Feature Driven Development
  - Adaptive Software Development
  - Crystal
  - Lean Software Development

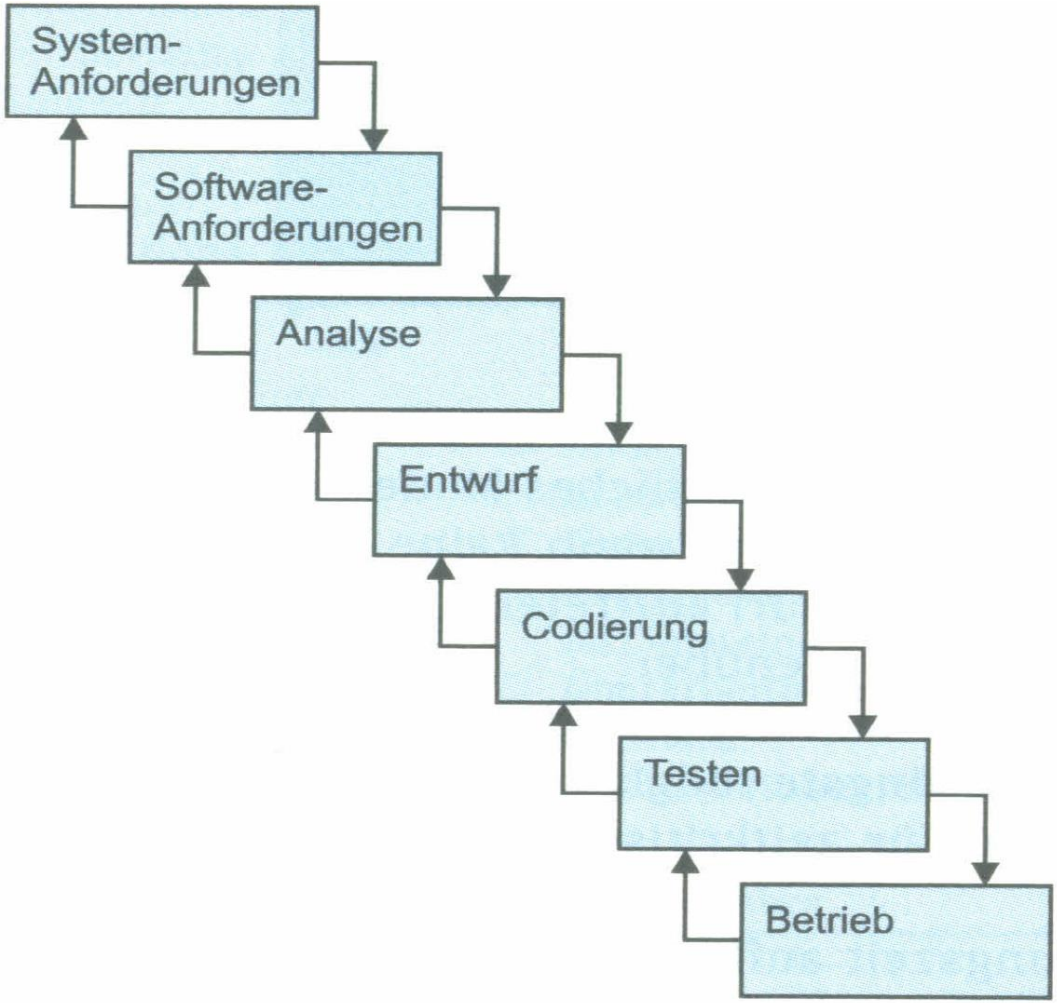
Einführung in die Vorgehens- und Prozessmodelle der Softwareentwicklung

- Klassische Vorgehensmodelle
  - Wasserfallmodell
  - Nebenläufiges Modell
  - Spiralmodell
  - Prototyping
- Monumentale Vorgehensmodelle
  - V-Modell XT
  - Rational Unified Process (RUP)
- Agile Prozesse
  - Scrum
  - Kanban (IT-Kanban)
  - Extreme Programming (XP)
  - Feature Driven Development
  - Adaptive Software Development
  - Crystal
  - Lean Software Development

Wasserfallmodell
Historie, Entstehung, Einsatzgebiete

- Seit den sechziger Jahren wachsende Probleme bei der Durchführung mittlerer bis größerer Softwareentwicklungsprojekte
- Das **Wasserfallmodell** wurde 1970 von Royce erfunden und ist die bekannteste Ausprägung des sequenziellen Modells
- Es ist eine Weiterentwicklung des **Stagewise Models**, welches festlegt, dass Software in sukzessiven Stufen entwickelt wird
- Das von Royce vorgeschlagene Wasserfallmodell erweitert das **Stagewise Model** um Rückkopplungsschleifen zwischen den Stufen

Wasserfallmodell
Prozessmodell



Wasserfallmodell
Inhalte und Ablauf

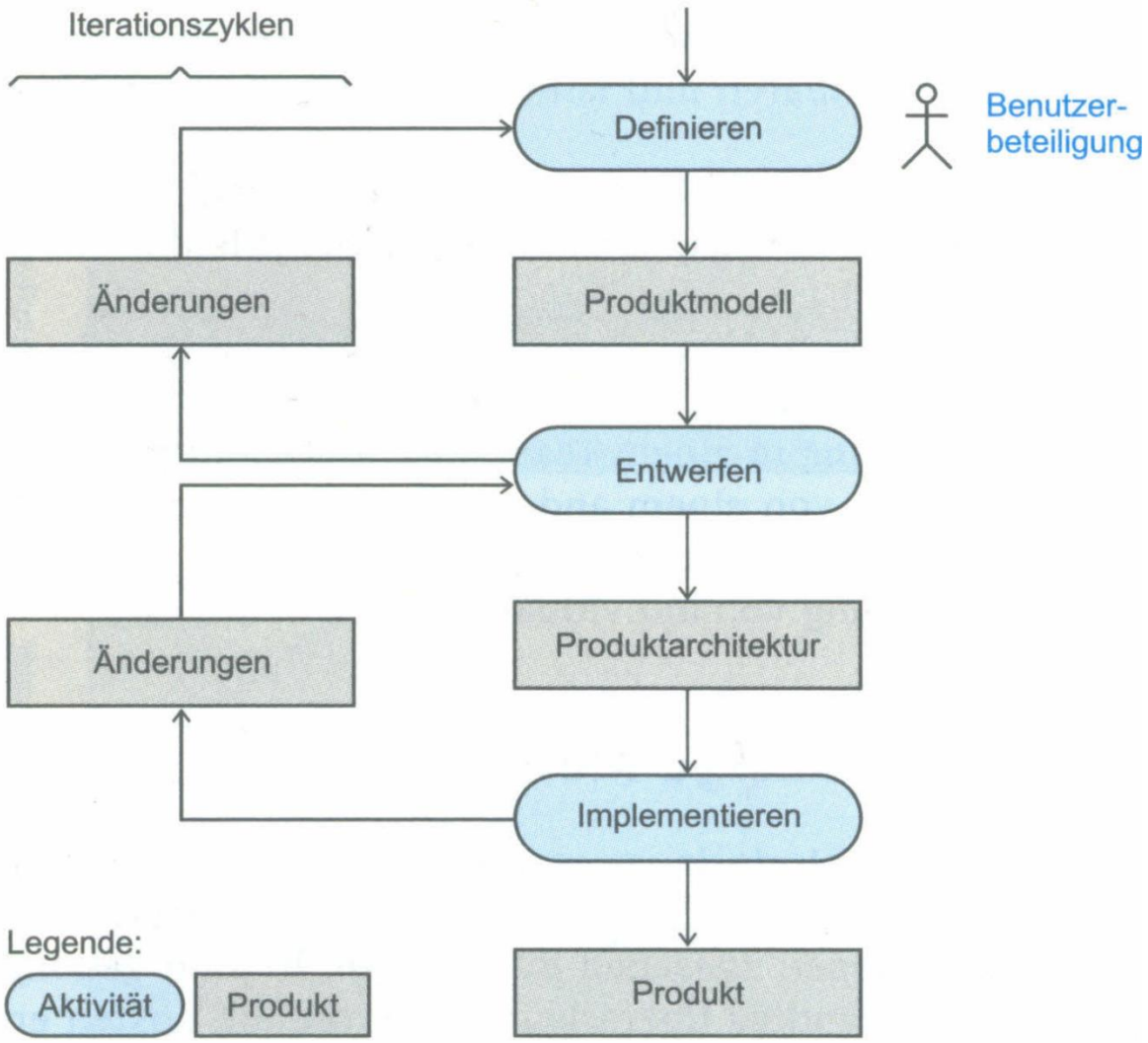
- Wasserfallmodell strukturiert den Entwicklungsprozess in sequenziell durchzuführende Phasen
- Für jede Phase sind nur die Ergebnisdokumente der vorherigen, komplett abgeschlossenen Phase relevant
- Idee: Umsetzung eines sequenziellen, vollständig vorhersehbaren Entwicklungsprozesses
- Rücksprünge sind in die direkt vorhergehende Phase möglich
- Analogie: Autoproduktion, Haus- bzw. Brückenbau

Wasserfallmodell
Einschränkungen

- Rücksprünge nur in die direkt vorhergehende Phase sind häufig unzureichend (auch wenn sie in der Regel teuer sind)
- Es wird vom (theoretischen) Idealfall ausgegangen, dass zu Beginn des Entwicklungsprozesses alle Anforderungen identifiziert werden
  - Das ist wünschenswert für Festpreisprojekte
  - Das ist in der Praxis nur selten der Fall

Wasserfallmodell
Normierte Darstellung

Fachhochschule Dortmund
University of Applied Sciences



Wasserfallmodell
Charakteristika

- *Jede* Aktivität ist in der *richtigen Reihenfolge* und in *der vollen Breite vollständig* durchzuführen
- Am Ende jeder Aktivität steht ein fertiggestelltes Dokument, d.h. das Wasserfallmodell ist ein dokumentenorientiertes Modell
- Der Entwicklungsablauf ist sequenziell, d.h. jede Aktivität muss beendet sein, *bevor* die nächste anfängt
- Eine Benutzerbeteiligung ist *nur* in der Definitionsphase (Anforderungserhebung) vorgesehen; später nicht mehr!
- Somit erfolgen der Entwurf und die Implementierung ohne Beteiligung der späteren Benutzer bzw. des Auftraggebers

Wasserfallmodell
Vorteile

- Das **Wasserfallmodell** ist nah an der planerischen und ingenieursmäßigen Denkweise aus Produktions- und Fertigungsprozessen
  - Auto, Haus, Brücke etc.
- Treten tatsächlich *keine Anforderungsänderungen* im Softwareentwicklungsprozess auf, können auch mit dem Wasserfallmodell gute Projektergebnisse erzielt werden
- „Idealisierte Wunschvorstellung“ für jedes Festpreisprojekt
  => ALLE Anforderungen sind zu Beginn des Projekts vollumfassend und präzise erhoben, bestimmt, analysiert, festgelegt und dokumentiert

Wasserfallmodell
Nachteile (1/2)

- **Problem**: Softwareentwicklung ist nicht vergleichbar mit konventionellem, gut verstandenem Produktionsprozess
  - Haus, Auto etc.
- **Ergebnis**: In der Praxis hat sich das Wasserfallmodell als ineffizient erwiesen
- Zudem: Große Gefahr, dass die Dokumentation wichtiger wird, als das eigentliche Softwaresystem
- Die Risikofaktoren werden unter Umständen zu wenig berücksichtigt, da immer der einmal festgelegte Entwicklungsablauf durchgeführt wird

Wasserfallmodell
Nachteile (2/2)

- **Problem**: Bei der Softwareentwicklung muss zu jedem Zeitpunkt mit zusätzlichen und/oder veränderten Anforderungen gerechnet werden
- Bei zusätzlichen und/oder veränderten Anforderungen sind Rücksprünge in bereits abgeschlossene, weiter zurück liegende Phasen notwendig
- Und *Rücksprung* heißt nicht *Rückblick* sondern kann ggf. erhebliche Änderungen auf die Ergebnisse einer Stufe und somit auf das gesamte Softwaresystem haben
- Aber Rücksprünge über mehrere Stufen hinweg sind im Wasserfallmodell nicht vorgesehen
- Somit kann das Wasserfallmodell nicht adäquat auf Anforderungsänderungen bzw. neue Anforderungen in späten Projektphasen reagieren

Wasserfallmodell
Klassifizierung

- Trotz der zahlreichen Nachteile hat das **Wasserfallmodell** zu einem disziplinierten, sichtbaren und kontrollierbaren Prozessablauf beigetragen
  - *„besser als ungeplantes Ad-hoc-Vorgehen“*
- Die genannten Charakteristika haben dem **Wasserfallmodell** zu einer weiten Verbreitung verholfen
- Das **Wasserfallmodell** war die Basis für viele Auftragsstandards in Behörden und in der Industrie
- Menschliche Neigung zu einer idealtypischen **wasserfallmodell-artigen** Planung zu Beginn einer Softwareentwicklung
- Es ist ein starres, grobgranulares und sehr softwarespezifisches Modell zur Entwicklung von Individualsoftware

Anekdote

- Winston W. Royce hat die heute als traditionelles Wasserfallmodell bekannte Methode in einem Konferenzpapier einer Ingenieurskonferenz beschrieben.
- Das Modell sollte als Negativbeispiel dienen und zeigen, wie Softwareentwicklung nicht durchgeführt werden sollte.
- Royce hat dann einen iterativen Prozess als kategorisch überlegenes Vorgehensmodell beschrieben.
- Das Wasserfallmodell wurde dennoch adaptiert und angewendet, bspw. durch das US Department of Defense.

Quelle: Chris Sims, Hillary Johnson. The Elements of Scrum. Dymaxicon, 2011.

Nächste LV

- Klassische Vorgehensmodelle
  - Wasserfallmodell
  - Nebenläufiges Modell
  - Spiralmodell
  - Prototyping
- **Monumentale Vorgehensmodelle**
  - V-Modell XT
  - Rational Unified Process (RUP)
- **Agile Prozesse**
  - Scrum
  - Kanban (IT-Kanban)
  - Extreme Programming (XP)
  - Feature Driven Development
  - Adaptive Software Development
  - Crystal
  - Lean Software Development
  - SAFe

Herzlichen Dank für Ihre Aufmerksamkeit !