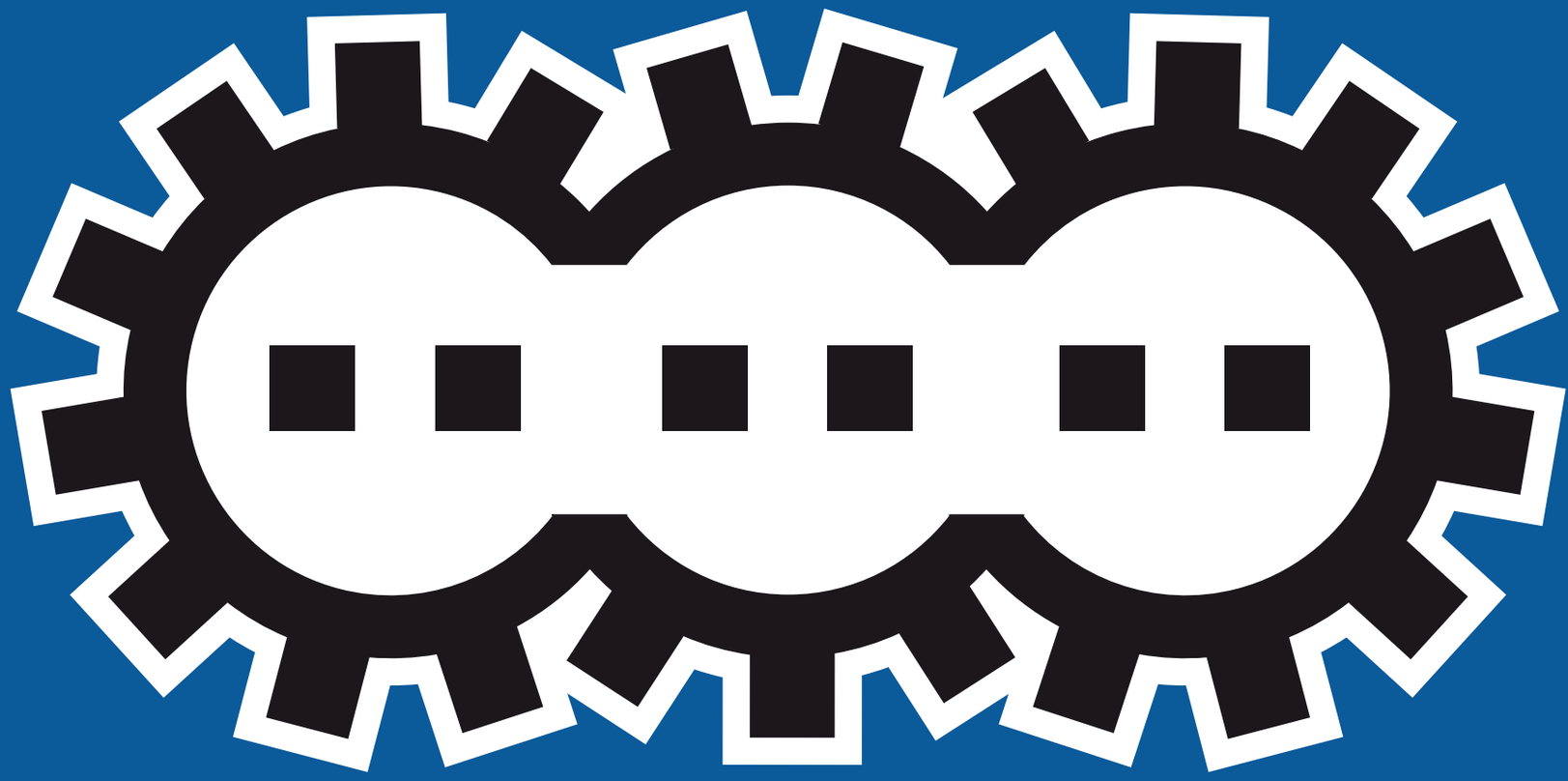
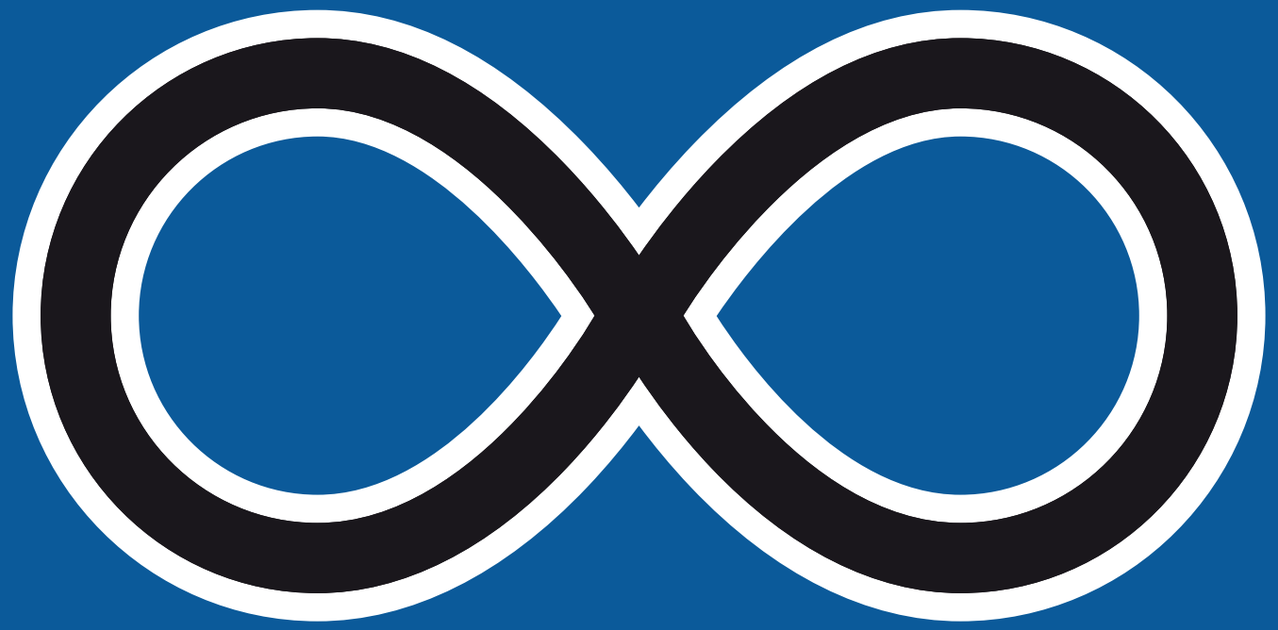


Software Engineering 2

02 - Requirements Engineering

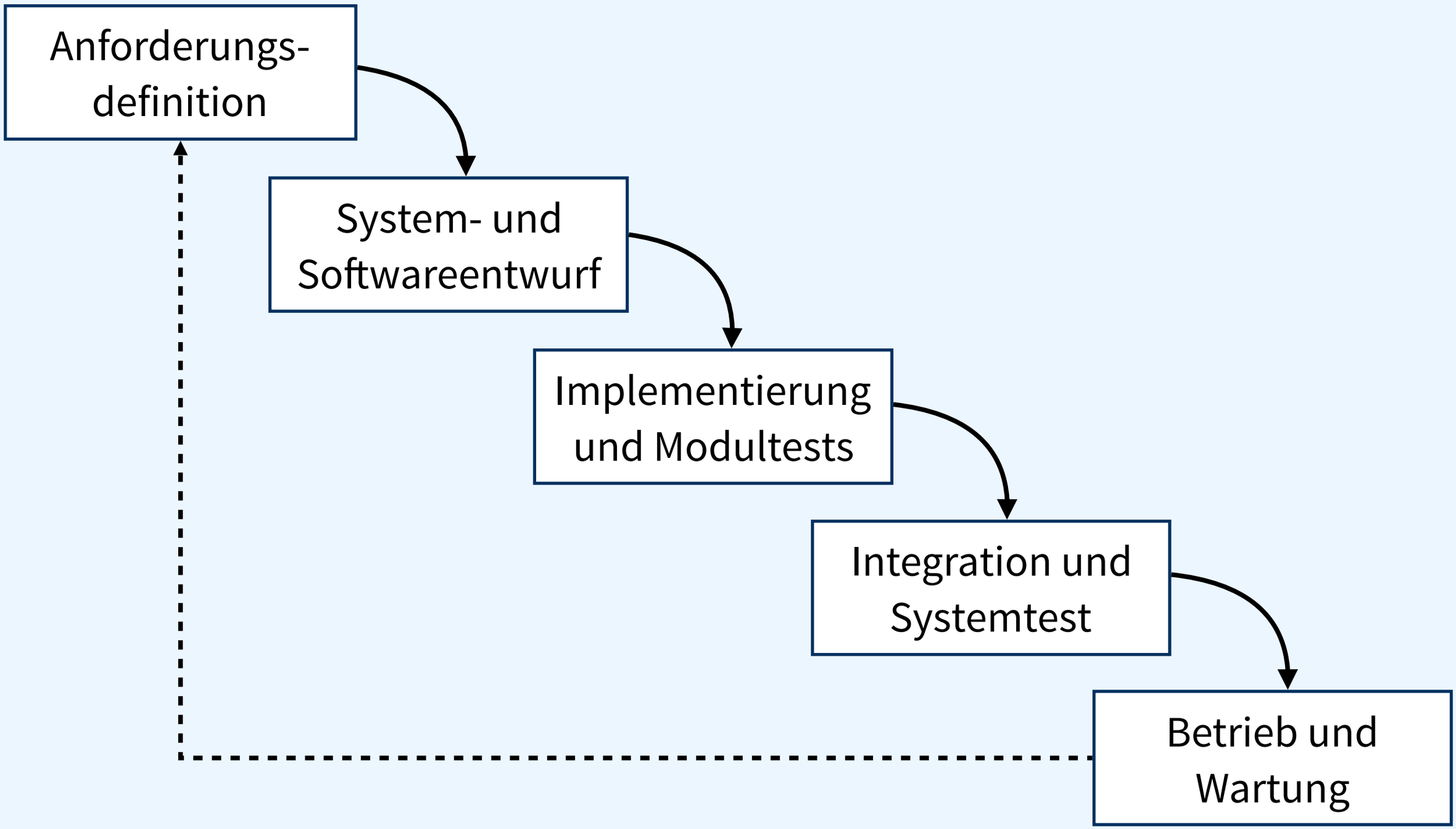
Prof. Dr. Stefan Betermieux | Fakultät Informatik | Hochschule Furtwangen



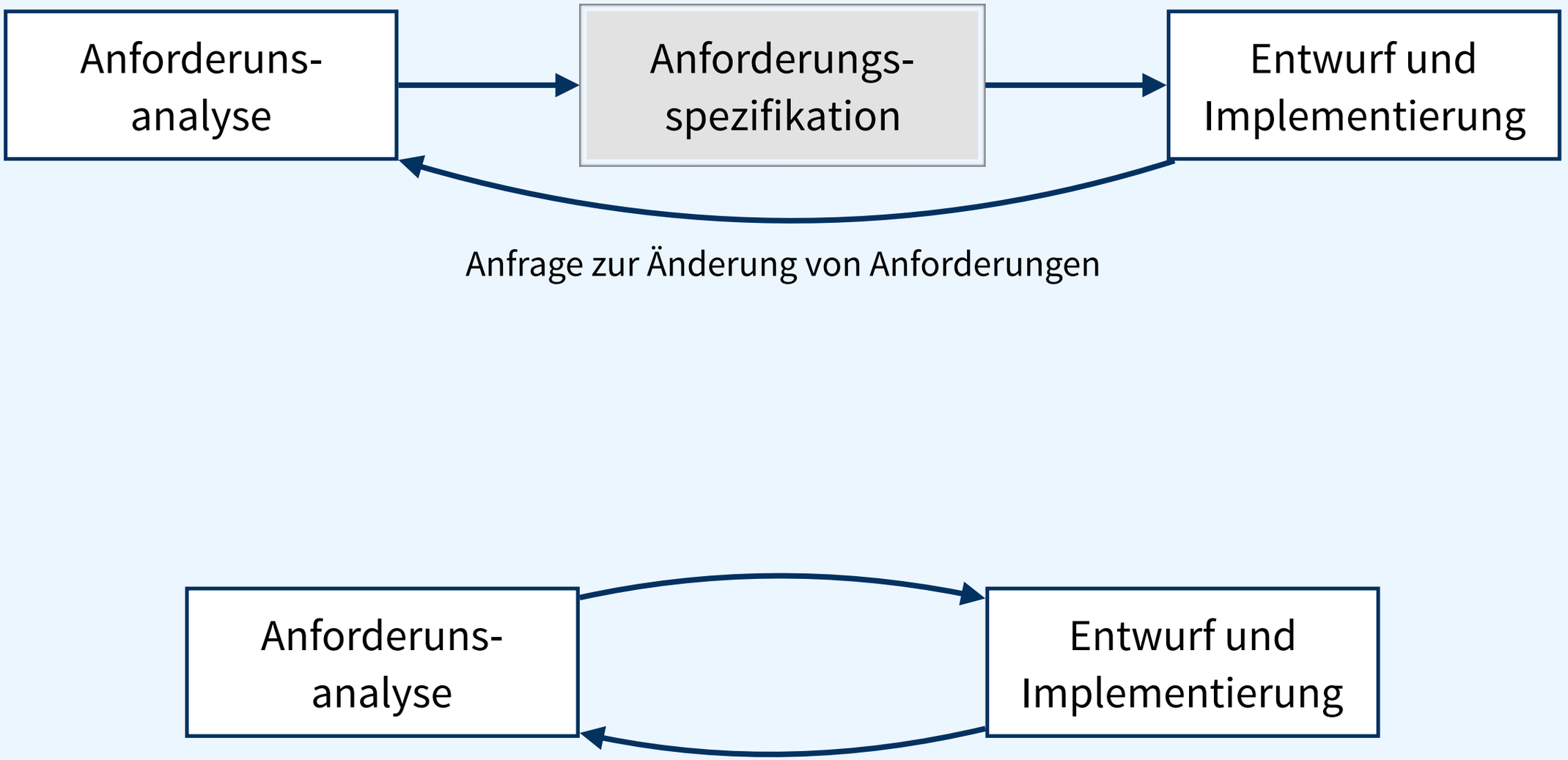
Wiederholung

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

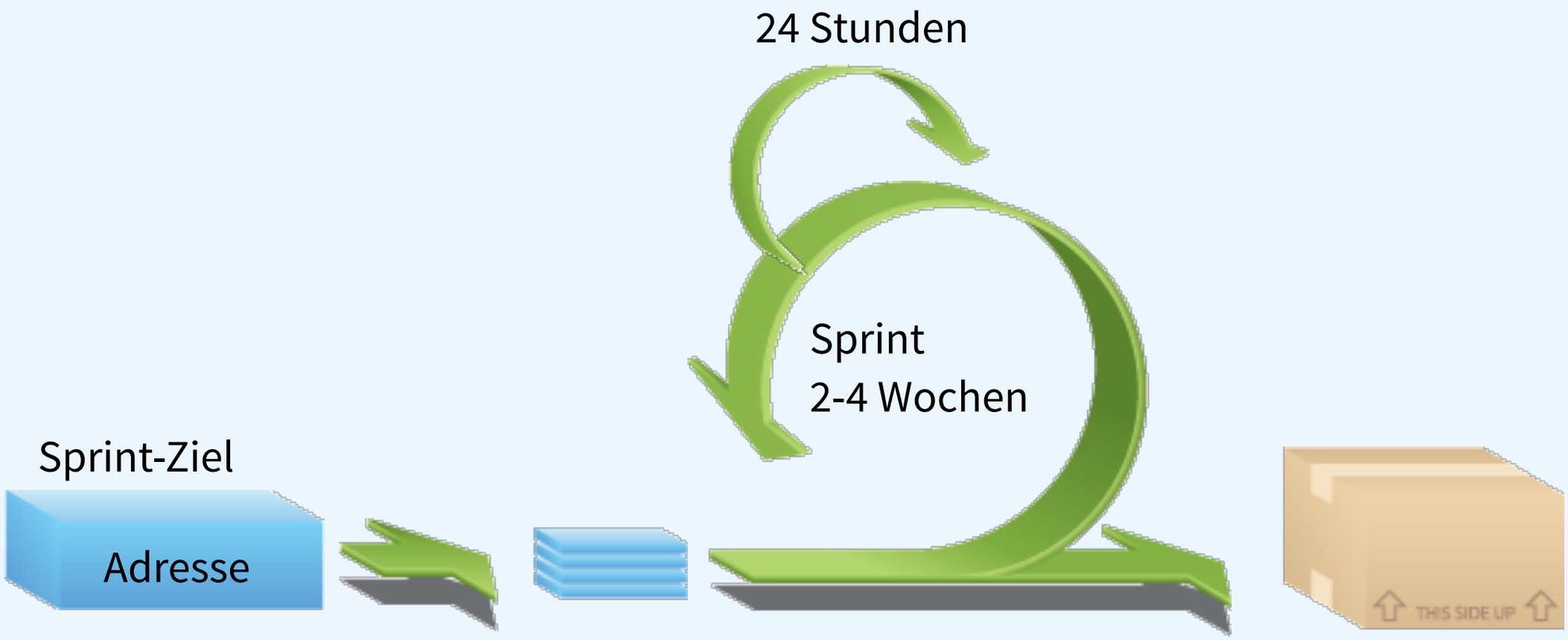
Wasserfallmodell



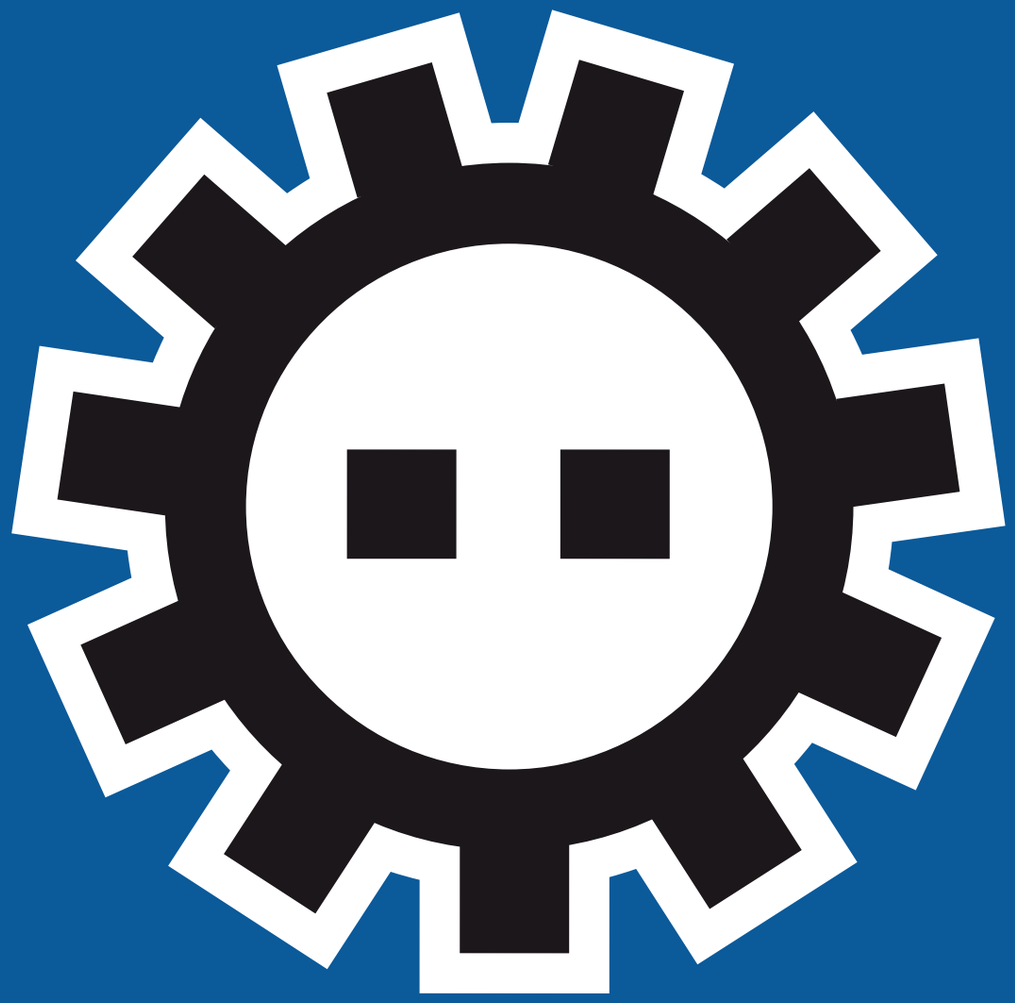
Planbasierte vs. agile Entwicklung



Scrum Prozess



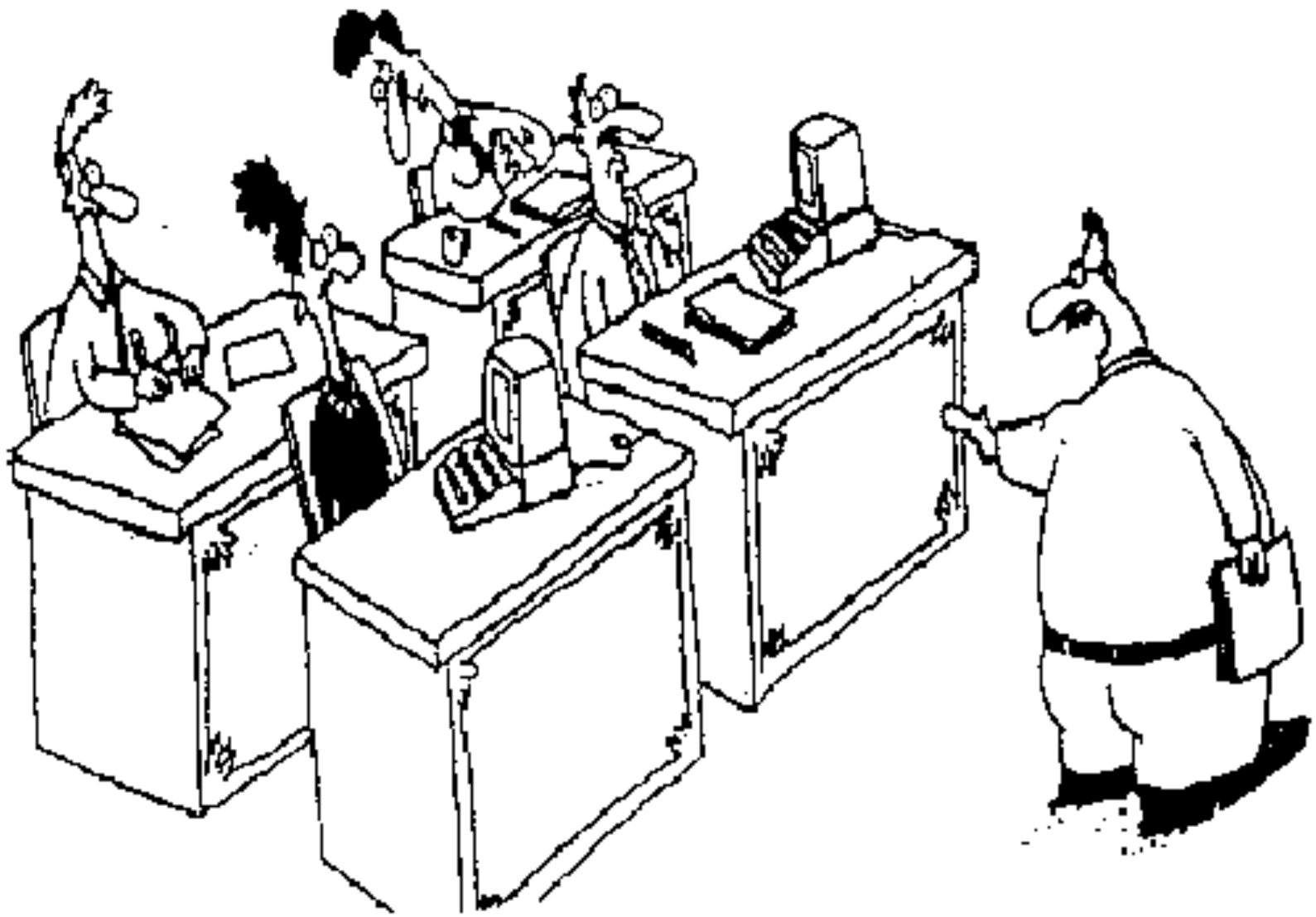
Quelle: Mountain Goat Software, LLC



Motivation

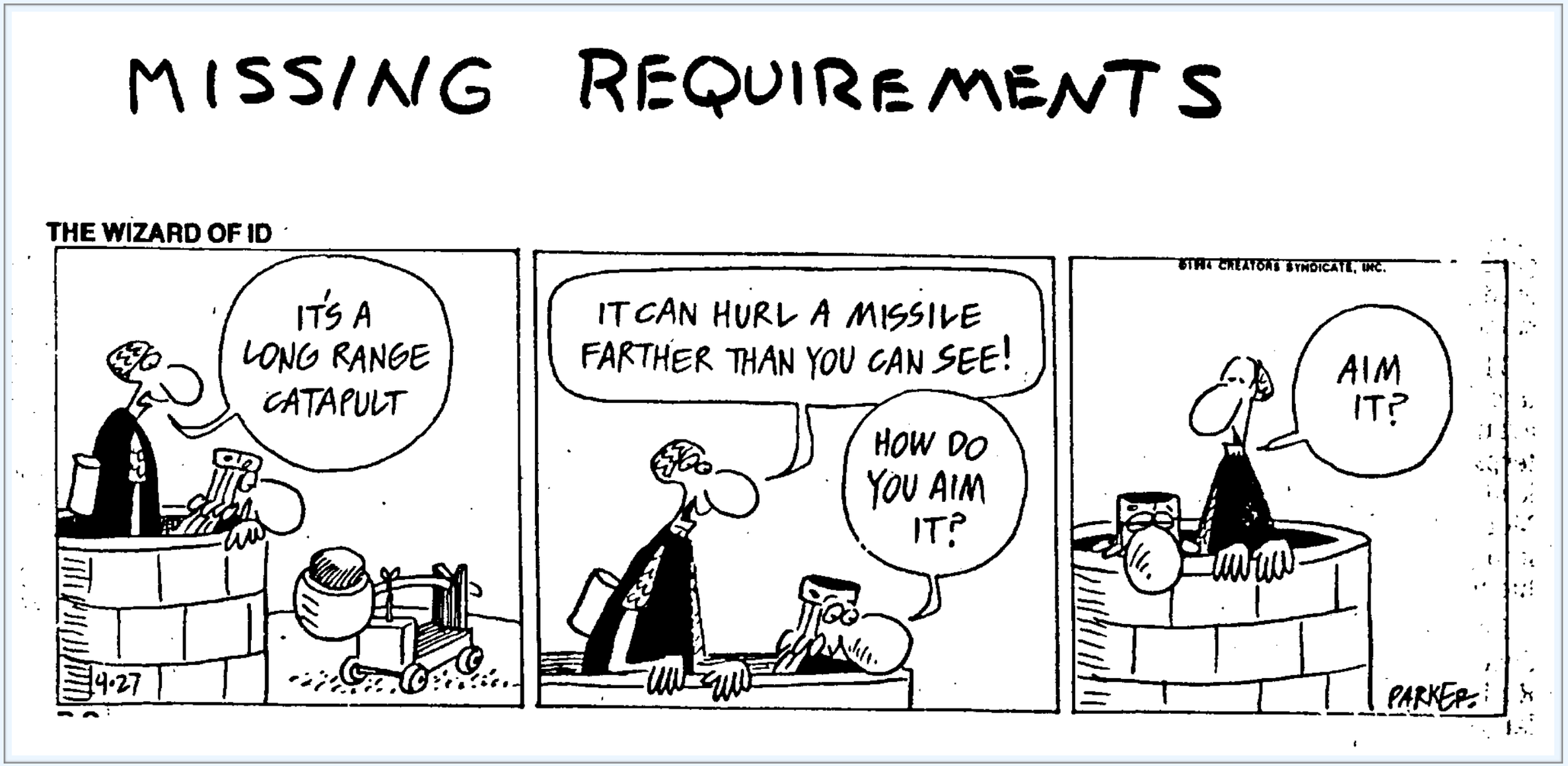
Prof. Dr. Stefan Betermieux | Fakultät Informatik | Hochschule Furtwangen

Los geht‘s …

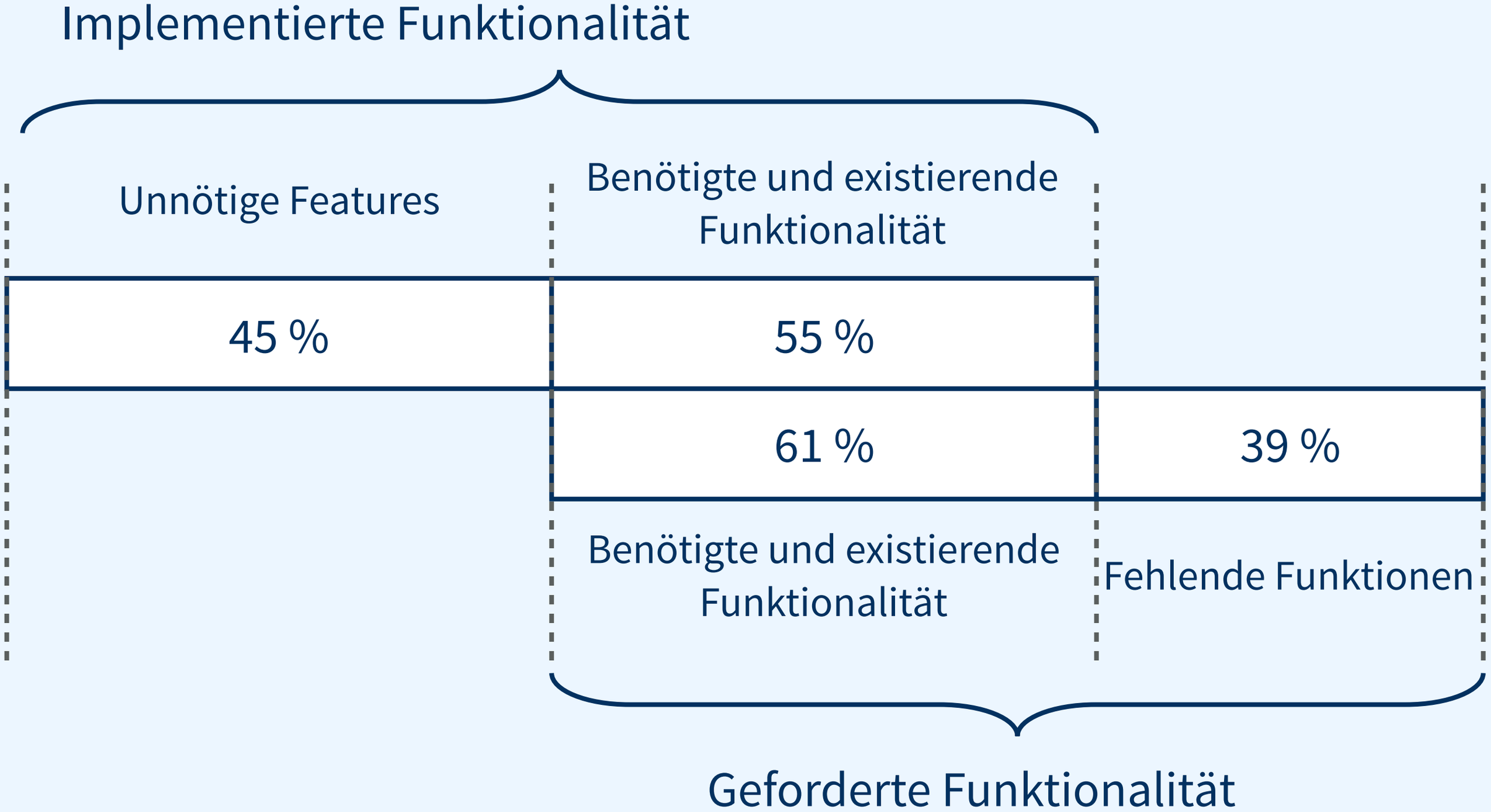


„I´ll go and find out what they need
and the rest of you start coding!“

Anforderungen



Woran Projekte scheitern?



Projektabbrüche (1)

- Vielzahl von Untersuchungen, wie erfolglos bzw. erfolgreich Softwareentwicklungsprojekte waren und sind
- Neuere Untersuchungen kommen zu folgenden Ergebnissen:
  - in 2005 und 2007 wurden zwei internationale Befragungen durchgeführt
  - 50% aller Projekte dauerten bis zu 9 Monaten
  - Zahl der Softwareentwickler schwankte zwischen 3 und 10
  - 2005 wurden 16% und 2007 12% der Projekte komplett abgebrochen, bevor *irgendetwas* ausgeliefert wurde
  - keine signifikanten Auswirkungen von *Projektdauer* oder *Anzahl der Projektbeteiligten* auf die abgebrochenen Softwareentwicklungsprojekte

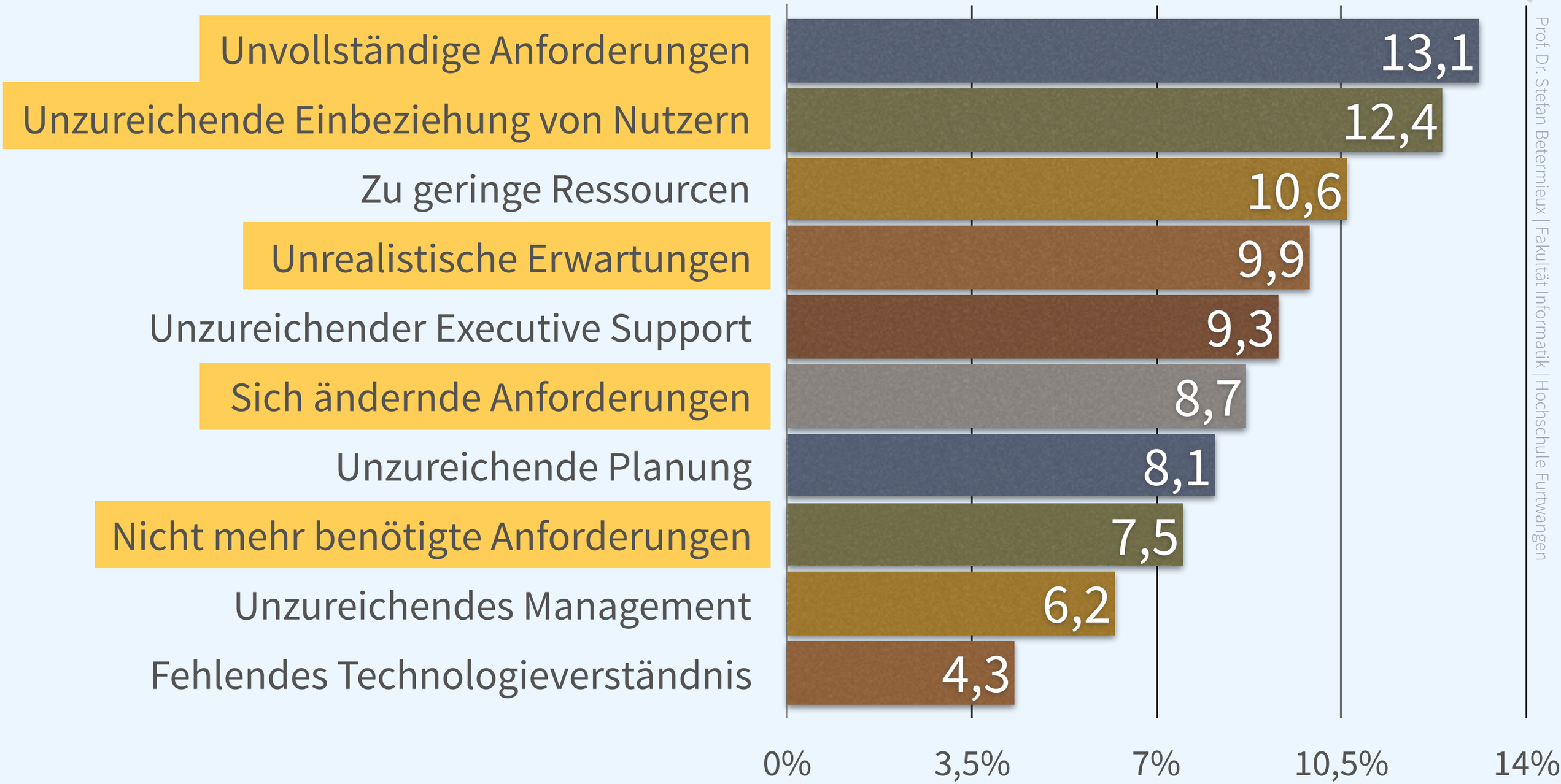
Projektabbrüche (2)

- Zwischen 48% (2005) und 55% (2007) der ausgelieferten Projektergebnisse waren erfolgreich
- Zwischen 17% und 22% der ausgelieferten Projektergebnisse waren nicht erfolgreich
- Kombiniert man die komplett abgebrochenen mit den nicht erfolgreichen Projekten, dann ergeben sich
  - für 2005: 34% Misserfolgsrate
  - für 2007: 26% Misserfolgsrate

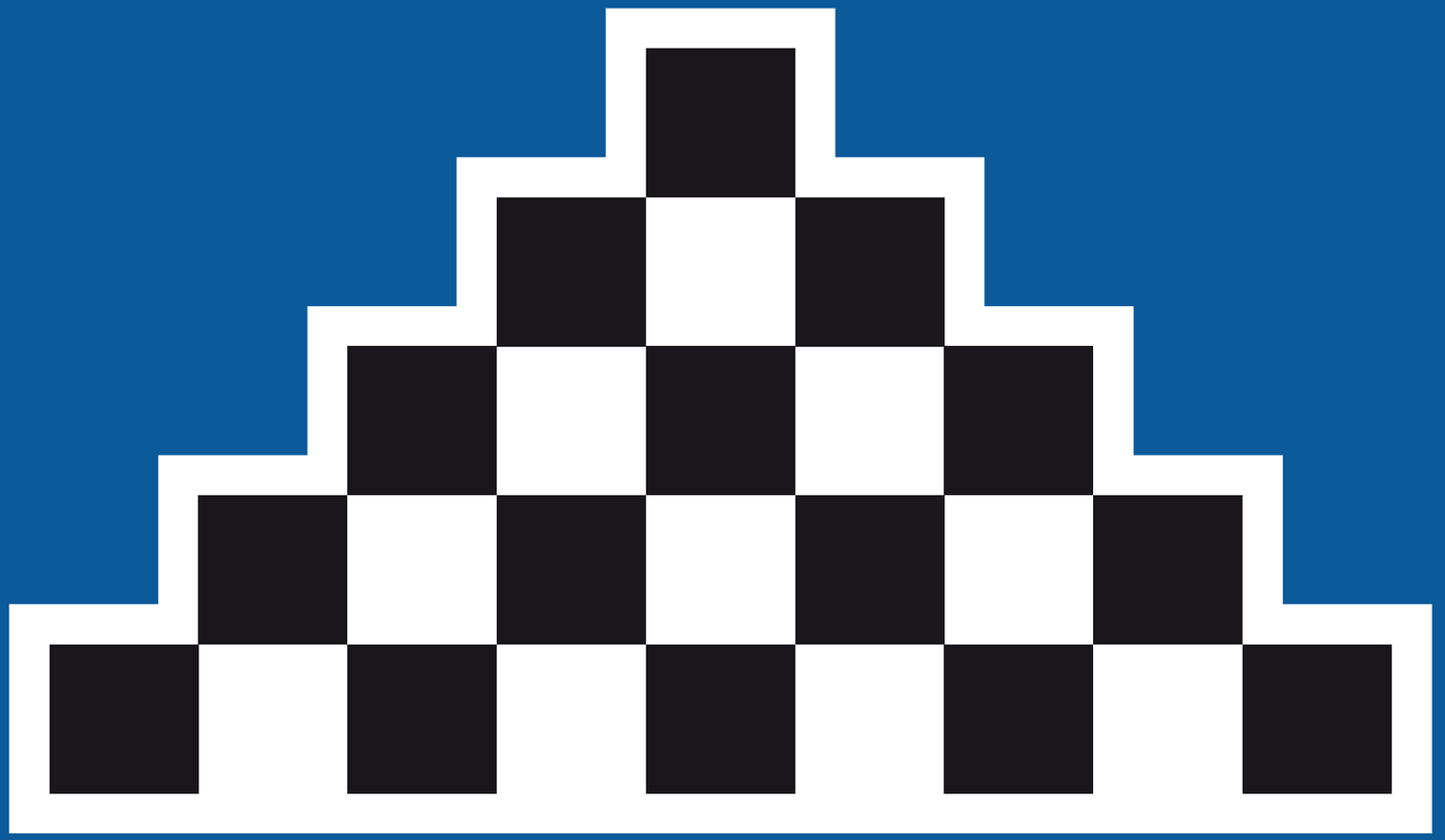
➡ Das ist eine sehr hohe Misserfolgsrate für eine angewandte Disziplin wie die Softwareentwicklung und Software-Engineering!

Gründe für das Scheitern

Faktoren, die zum Abbruch von Projekten führen:



Quelle: Chaos Report 1 The Standish Group Report



Grundlagen

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

Requirements Engineering

- Notwendige Grundlage für die Erstellung innovativer, individueller und umfangreicher Systeme
  - mit angemessenem Aufwand
  - und in der geforderten Qualität
- Fehlerfreie und vollständige Anforderungen sind die Basis für die erfolgreiche Systementwicklung
- Bereits im Requirements Engineering müssen die potenziellen Risiken aufgedeckt und soweit möglich behoben werden
- Fehler und Lücken in den Anforderungsdokumenten müssen frühzeitig erkannt werden, um langwierige Änderungsprozesse zu vermeiden

Requirements Engineering: Ziele

Das richtige Produkt entwickeln:

- Werden die richtigen Kunden und Nutzer ausgewählt?
- Werden die richtigen Anforderungen ermittelt?
- Reflektieren die Anforderungen aktuelle Kundenwünsche?
- Steht das Produkt im richtigem Preis-/ Leistungsverhältnis?
- Werden die Anforderungen verständlich niedergeschrieben?
- Können die Anforderungen (optimal) umgesetzt werden?
- …

Anforderungen und Ziele

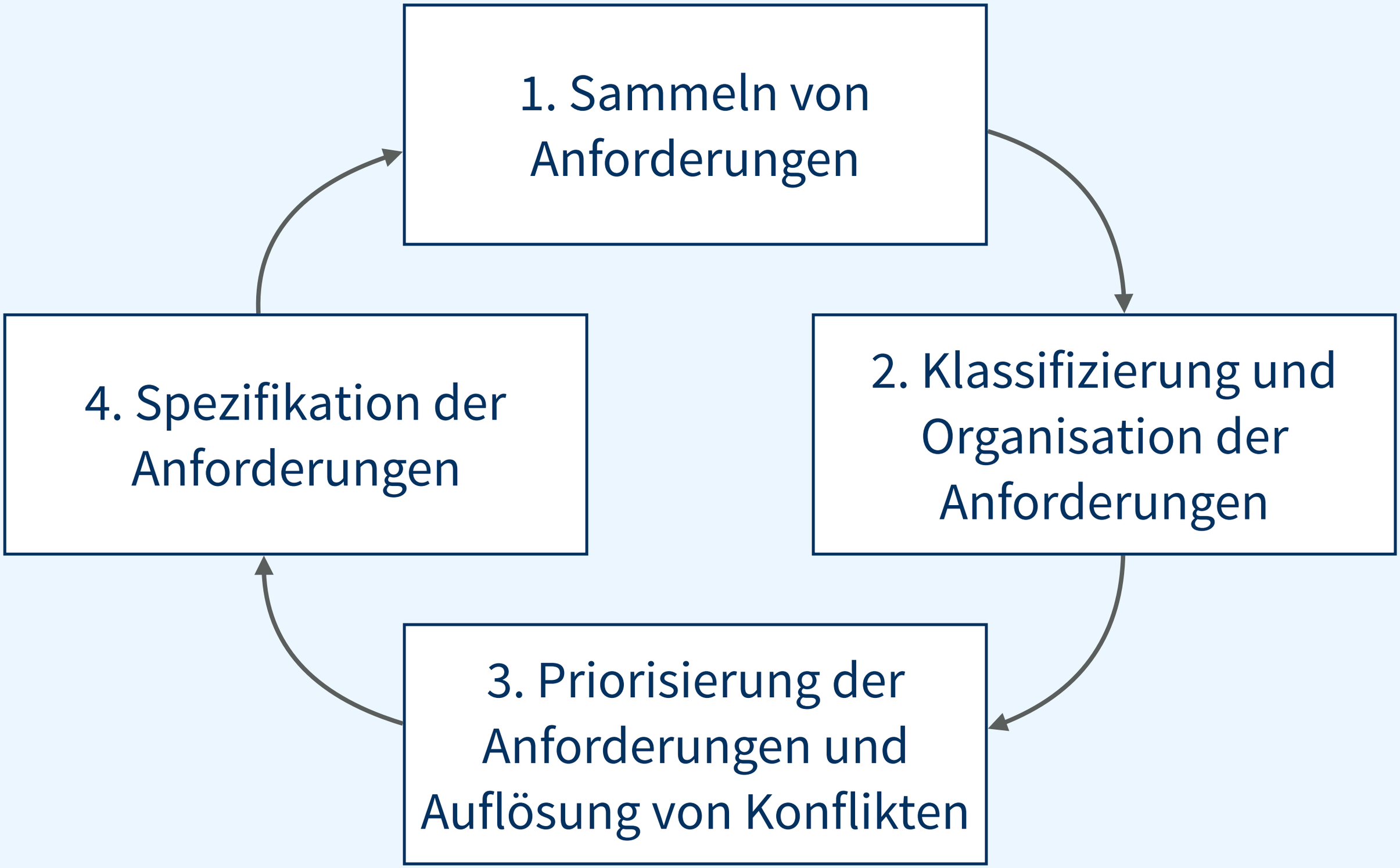
Übung

- Welche Anforderungen stellen Sie an den Einkauf von Tomaten im Supermarkt?
- Welche Ziele verfolgen Sie mit dem Kauf von Tomaten?

- Anforderungen (z. B.):
  - Produkt: frisch, Farbe, Form
  - Preis
  - Art der Präsentation
  - verfügbar
- Ziele (z. B.):
  - Salat zu essen

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

Anforderungsanalyse



Definition: Anforderung

- Eine dokumentierte Darstellung einer Bedingung oder Fähigkeit gemäß 1 oder 2:
  1. Beschaffenheit oder Fähigkeit, die von einem Benutzer zur Lösung eines Problems oder zur Erreichung eines Ziels benötigt wird.
  2. Beschaffenheit oder Fähigkeit, die ein System oder System-Teile erfüllen oder besitzen muss, um einen Vertrag, eine Norm, eine Spezifikation oder andere, formell vorgegebene Dokumente zu erfüllen.

  Gemäß IEEE Standards Board: IEEE Std 610.12-1990

- In der (Software-)Technik ist eine Anforderung (= Requirement) eine Aussage über eine zu erfüllende Eigenschaft oder zu erbringende Leistung eines Produktes, Systems oder Prozesses

  Quelle: Wikipedia

Definition: Ziel

- Was ist ein Ziel?
  - unter einem Ziel wird ein erstrebenswerter Zustand verstanden, der in der Zukunft liegt und dessen Eintritt von bestimmten Handlungen bzw. Unterlassungen abhängig ist, also nicht automatisch eintritt
- Warum sind Ziele notwendig?
  - alle Anforderungen richten sich immer auf das Ziel aus. Es darf keine Anforderungen geben, die kein bestimmtes Ziel verfolgen.
- Wie finde ich ein Ziel?
  - Analyse der Ist-Situation; die Probleme der bestehenden Systeme oder Visionen herausarbeiten und den Zielzustand definieren.

Kreativitätstechnik SMART

Ziele müssen klar, eindeutig, messbar, unmissverständlich und erreichbar sein. Hier hilft *SMART*:

- **S**pezifisch: Ziele müssen eindeutig sein
  - Was konkret? Genaue Beschreibung des erwünschten Zustandes
- **M**essbar: Ziele müssen messbar sein
  - Woran erkennbar? Kriterien mit Hilfe derer sich der Erfolg überprüfen lässt
- **A**ktiv beeinflussbar / Angemessen: Ziele müssen erreichbar sein
  - Das Ziel liegt im eigenen Einflussbereich
- **R**elevant / Realisierbar: Ziele müssen bedeutsam sein
  - Ist das Ziel wichtig im Zusammenhang mit den Unternehmenszielen, herausfordernd und erreichbar?
- **T**erminiert: Zu jedem Ziel gehört eine klare Terminvorgabe
  - Wann genau? Präziser Termin, zu dem das Ziel erreicht sein soll

Übung

- Was ist das Ziel Ihres Studiums?
- Was ist das generelle Ziel eines Projektleiters?

- Ziel des Studiums (z.B.):
  - Job, Geld, Unterhalt
- Ziel des Projektleiters (z.B.):
  - Abnahme durch Kunde / Kundenzufriedenheit
  - Projektziel des Unternehmens erreicht: Termin, Qualität, Kosten

Stakeholder

- Stakeholder sind:
  - Eine Person, Personengruppe oder eine Organisation, die aktiv am Projekt beteiligt ist,
  - oder von dem Projektverlauf oder dem Projektergebnis beeinflusst wird,
  - oder gegebenenfalls den Projektverlauf oder das Projektergebnis selber beeinflusst,
  - dazu gehören auch Standards, Normen oder sonstige Richtlinien

Stakeholder-Analyse

- Stakeholder-Analyse besteht im wesentlich aus drei Schritten:
  - Identifikation der Stakeholder / Projektbeteiligten
  - Bestimmung der Anforderungen der Stakeholder
  - Ableitung von Konsequenzen und Maßnahmen für das Projekt
- Stakeholder-Management ist ein
  - dauerhafter Prozess während der gesamten Projektlaufzeit

Übung

- Wer sind Stakeholder im Supermarkt beim Verkauf von Kopfsalat?
- Wer sind Stakeholder beim Projekt „Studium“?

- Kopfsalat: Kunde, Verkäufer, Marktleiter, Transporteur, Erzeuger / Gärtner, Staat / Steuer, Gesundheitsamt / Lebensmittelkontrolle
- Studium: Studenten, Professoren / Lehrbeauftragte / Assistenten, Verwaltung, Eltern, Prüfungskommission, Studentenwerk, Externe Arbeitgeber

Arten von Anforderungen

Arten von Anforderungen

- Funktionale Anforderung
  - eine funktionale Anforderung legt fest, *was* das Produkt tun soll
  - Beispiel: „Das Produkt soll den Saldo eines Kontos zu einem Stichtag berechnen.“
- Nicht-funktionale Anforderung
  - eine nicht-funktionale Anforderung legt fest, *welche* qualitativen *Eigenschaften* („Qualitätsanforderungen“) das Produkt haben soll
  - Beispiel: „Das Produkt soll dem Anwender innerhalb von einer Sekunde antworten.“

Funktionale Anforderungen

- Beschreiben die Funktionen und Dienste, die ein Softwaresystem bereitstellen soll
  - Reaktion des Softwaresystems auf bestimmte Eingaben
  - auch: Beschreibung, was das System NICHT leisten soll
- Man unterscheidet hierbei
  - Benutzeranforderungen und
  - Systemanforderungen

Benutzer-/Systemanforderungen

- Benutzeranforderungen:
  - sind Aussagen in natürlicher Sprache,
  - intuitiv verständliche Diagramme zur Beschreibung der Funktionen und Dienste, die das spezifizierte Softwaresystem leisten soll,
  - die Randbedingungen, unter denen es betrieben wird.
- Systemanforderungen:
  - basieren auf Benutzeranforderungen
  - legen die Funktionen, Dienste und Beschränkungen detailliert und möglichst präzise fest (→ Quantifizierbarkeit).
  - es muss genau spezifiziert werden, welche Anforderungen zu implementieren sind.

Beispiel

- Benutzeranforderung:
  - das Krankenhausverwaltungssystem soll am Ende des Monats ein Bericht über die Kosten aller verschriebener Medikamente erstellen
- Systemanforderung:
  - am letzten Arbeitstag eines Monats werden die Daten aller verschriebener Medikamente gesammelt
  - um 17:30 Uhr dieses Tages wird der Bericht als PDF erstellt
  - Zugriff auf die Liste bekommt nur das autorisierte Management

Nichtfunktionale Anforderungen (1)

- Sind Eigenschaften eines Softwaresystems
  - Anforderungen, die NICHT die durch das Softwaresystem bereitzustellenden Funktionen bzw. zu leistenden Dienste betreffen
- Sind selten an *einzelne* Systemfunktionen gebunden
- Oftmals sind einzelne nichtfunktionale Anforderungen deutlich relevanter als einzelne funktionale Anforderungen
- Es können hierbei unter anderem auch:
  - Vorgehensmodelle der Softwareentwicklung
  - Programmiersprachen und/oder
  - Entwicklungswerkzeuge festgelegt werden

Nichtfunktionale Anforderungen (2)

- Produktanforderungen
  - Effizienz
  - Zuverlässigkeit und Robustheit
  - Sicherheitsanforderungen
  - Ergonomische Anforderungen (»Look and Feel«)
- Unternehmensanforderungen
  - Entwicklungsanforderungen
  - Projektbedingte Rahmenbedingungen (Zeit, Kosten, Personal, …)
- Externe Anforderungen
  - Rechtliche Anforderungen (Bildschirmarbeitsverordnung)
  - Ethische Anforderungen

Anforderungsattribute

Identifikation von Anforderungen

- Jede Anforderung muss durch eine eindeutige Identifikation gekennzeichnet sein, zum Beispiel: <ProjektNr>-<BereichNr>-<AnforderungNr>
  - jede Anforderung muss einen sprechenden bzw. beschreibenden Kurznamen haben
- Jede Anforderung kann eine Klassifikation haben, welche die Anforderungen in einzelne Gruppen zerteilt
  - typische Klassifikationen sind zum Beispiel:
    - » Wichtig, Unwichtig, Optional
    - » Must Have, Should Have, Could Have, Nice To Have
  - diese Priorisierung erleichtert später die Planung der Reihenfolge der Implementierung der Komponenten

Abnahmekriterium

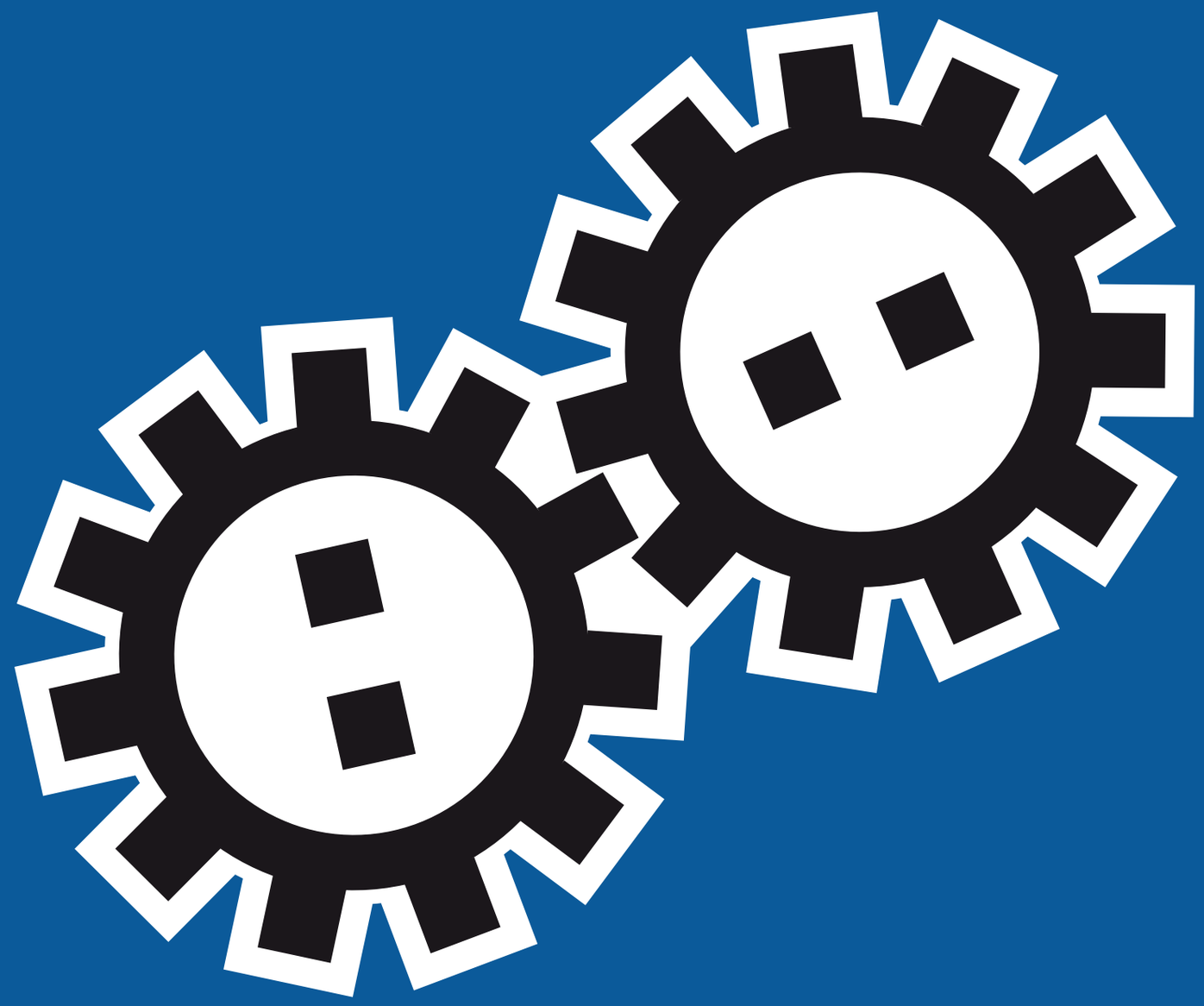
- Am Ende eines Projektes bzw. einer Iteration wird das entstandene Produkt gegen die Anforderungen getestet
  - mit den abgestimmten Anforderungen werden bereits die Test-Fälle für die Abnahme generiert
- Die Anforderung sollte damit einen Verweis auf den Test-Fall beinhalten:
  - eindeutige Identifikation von einem oder mehreren Test-Fällen
  - damit wird sichergestellt, dass die Abnahme des System gegen die Anforderungen erfolgt und nicht gegen die tatsächlich realisierte Implementierung
  - ein ganz entscheidender Unterschied!

Zusammenfassung Anforderungen

- Notwendige Attribute
  - Anforderungs-ID (eindeutige Bezeichnung)
  - Name / Kurzbeschreibung
  - Typ der Anforderung (funktional, etc.)
  - Klassifikation
  - Priorität
  - Status
- Optionale Attribute
  - Detailbeschreibung
  - Geschätzte Kosten
  - Quelle der Anforderung
  - Hinweis auf Testfälle

Qualitätskriterien

- Identifizierbar
  - jede Anforderung muss eindeutig identifizierbar sein
- Vollständig
  - alle Anforderungen müssen explizit beschrieben sein, es darf keine impliziten Annahmen über das zu entwickelnde System geben
- Nachvollziehbar
  - für jede Anforderung sollte es nachvollziehbar sein, in welcher implementierten Funktionalität die Anforderung umgesetzt wurde und umgekehrt
- Konsistent
  - alle Anforderungen sollten wechselseitig widerspruchsfrei sein



Techniken

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

Scrum

Agile Anforderungsermittlung

User Stories

- In *Scrum* werden die Anforderungen mithilfe von sogenannten *User Stories* („Anwendererzählung“) erhoben
- *User Stories* werden zusammen mit sog. *Akzeptanztests* im Rahmen von Scrum eingesetzt, um die Anforderungen zu bestimmen
- Die Anforderungen werden hierbei in der Begriffswelt des Anwenders mithilfe von natürlicher Sprache dokumentiert
- Eine *User Story* ist kurz gehalten und umfasst oftmals nur ein bis zwei Sätze
- Die User Story wird auf einer sog. *Story Card* notiert, wobei der Autor der Story Kunde des Softwareentwicklungsprojekts sein sollte

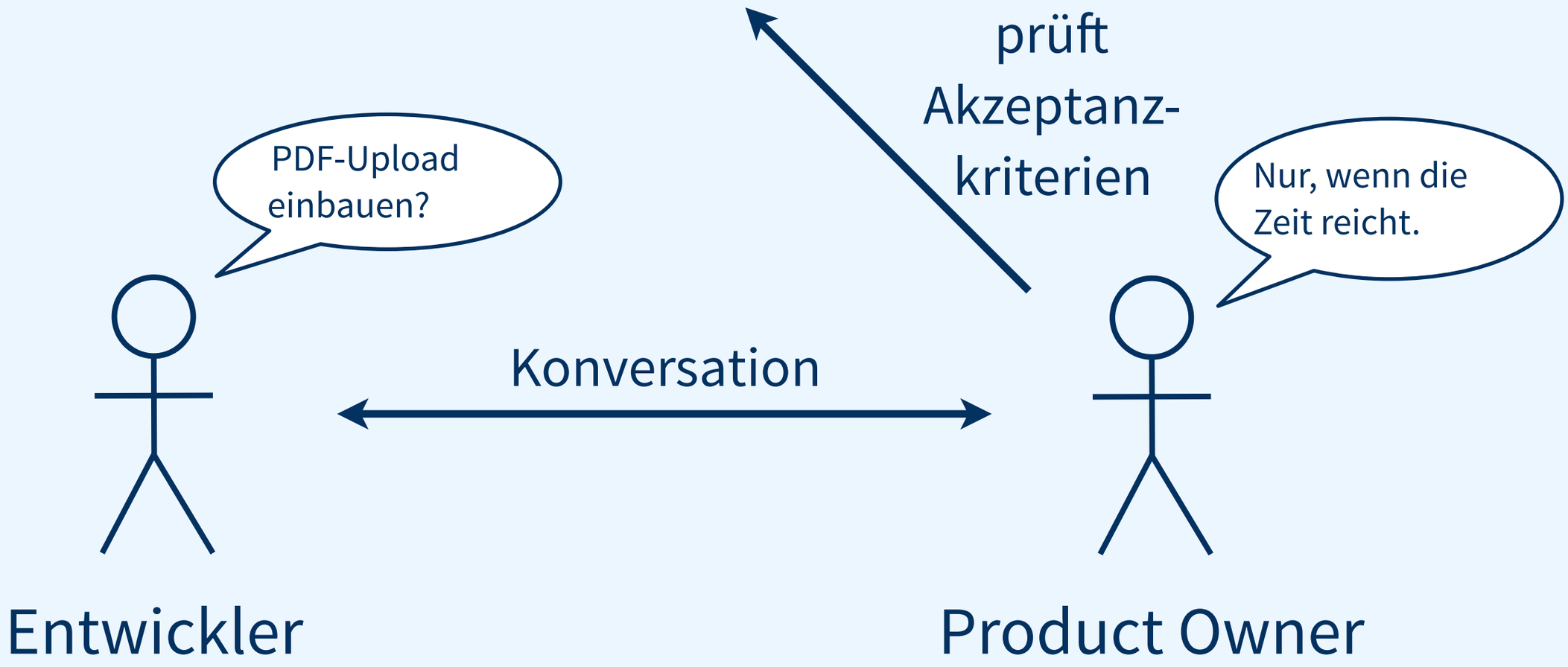
Story Card

Als Mitglied will ich mein Profil erstellen.

Bestandteile einer User Story

Als Mitglied will ich mein Profil erstellen.

Story Card



User Stories

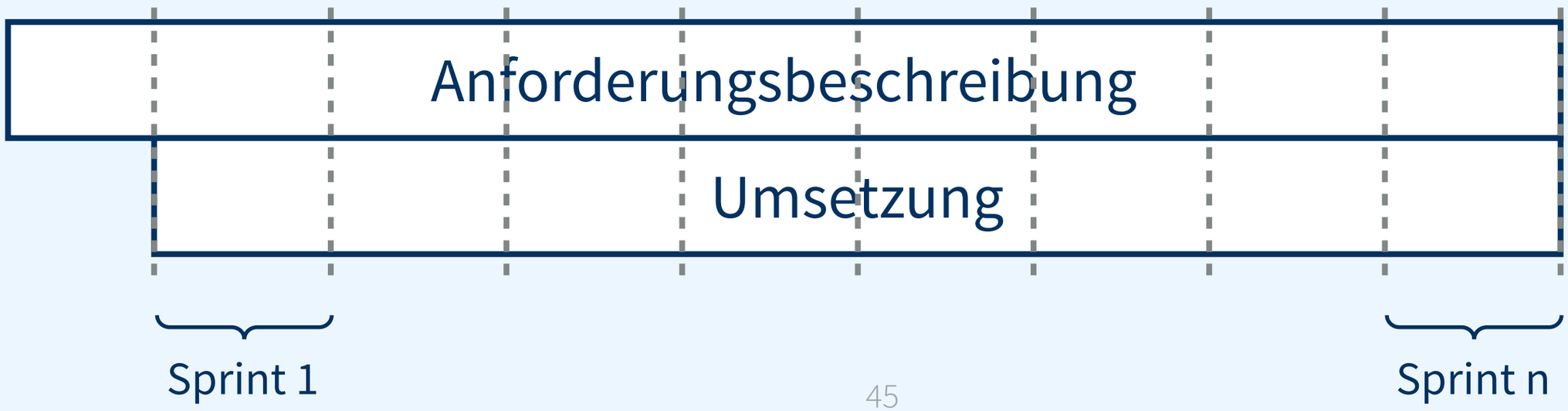
- Ein wichtiger Grundsatz bei User Stories ist, dass sie aus der Sicht eines Anwenders geschrieben werden
- Daher werden sie auch frei von technischen Begrifflichkeit (»Entwickler-Jargon«) gehalten
- Jeder, der am Projekt beteiligt ist, also auch jeder Anwender, sollte sie im Optimal-Fall verstehen können
- Hingegen sollte das Entwicklungsteam tolerant gegenüber der Fachsprache der Anwender sein

Trad. Anforderungsermittlung

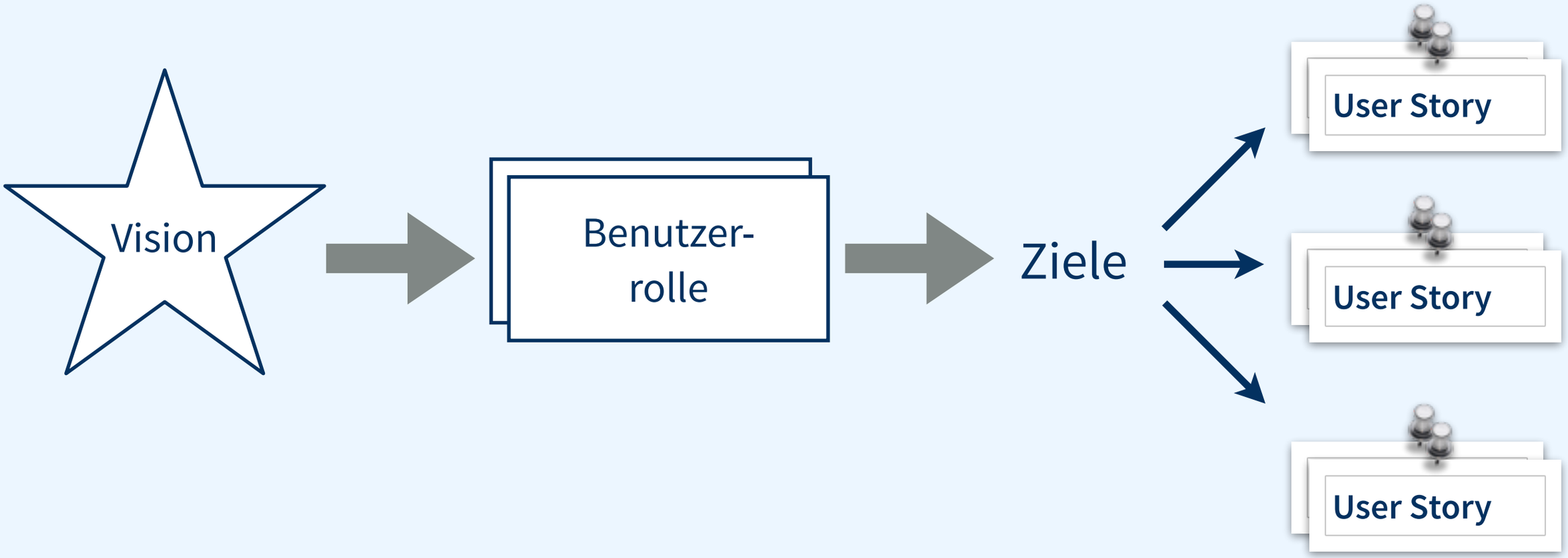
- Frühe, vollständige und genaue Beschreibung der Anforderungen
  - traditionelle Arbeitsorganisations- und Planungsverfahren aus der Fertigungsindustrie
  - genaue und umfassende Schätzung in Festpreisprojekten
  - bei komplexer und innovativer Softwareentwicklung oftmals ineffizient
- Probleme:
  - Aufbau eines umfangreichen Anforderungsinventars
  - Informationsverlust durch Übergaben
  - Überproduktion von Funktionalität
  - unausgeglichener Arbeitsanfall

Agile Anforderungsermittlung

- Daher werden Anforderungen in Agilen Modellen wie Scrum nicht einmal zu Projektbeginn erhoben und beschrieben
- Anforderungsbeschreibung und Umsetzung erfolgen zeitnah und überlappend:
  - Phasen „verschwimmen“
  - keine separierten Definitions und Implementierungs- bzw. Umsetzungsphase



Schreiben von User Stories (1)

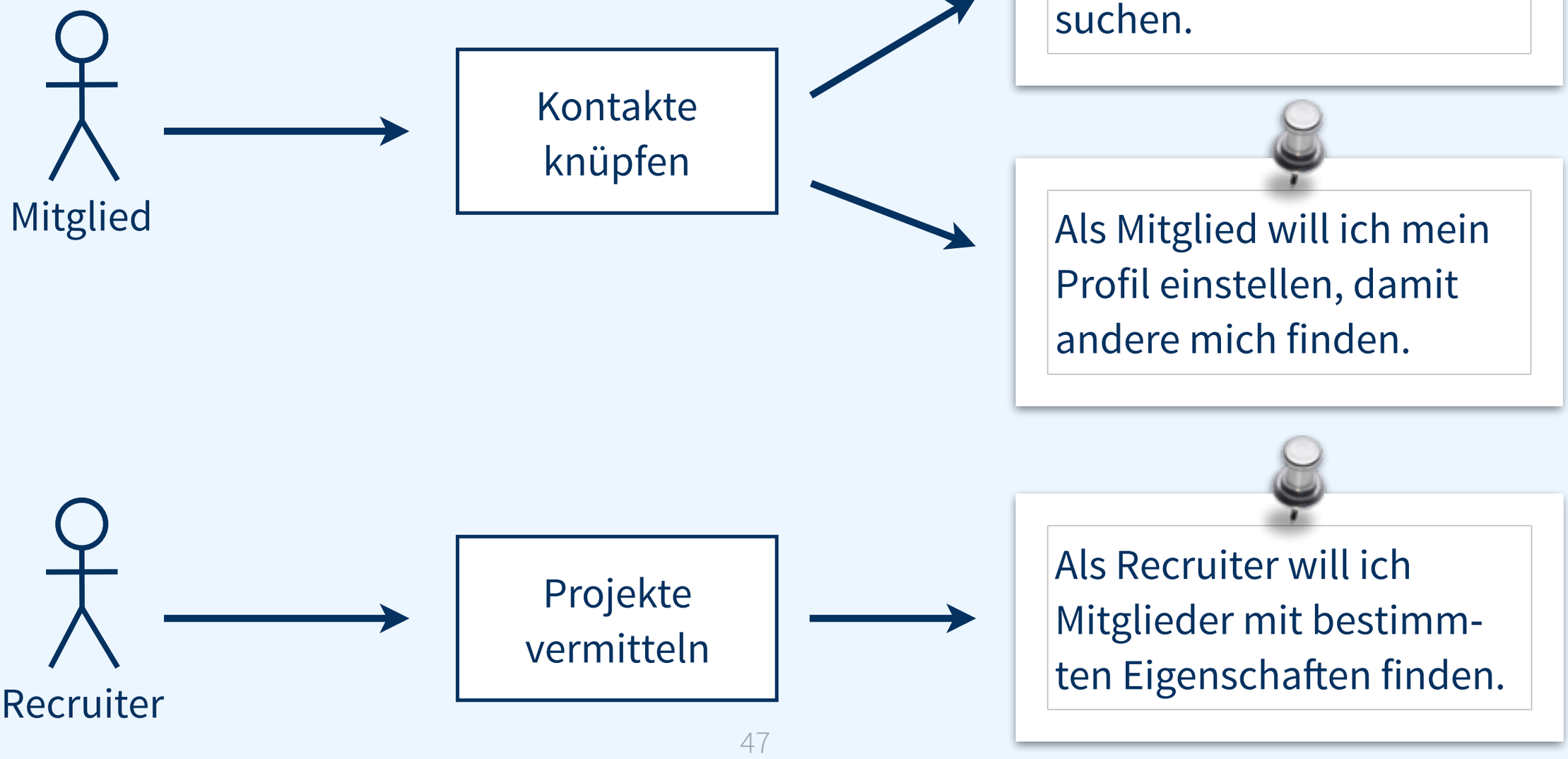


Schreiben von User Stories (2)

Rolle

Ziele

User Story



Schablonen

- *User Stories* können formlos oder ähnlich wie bei *Use Cases* mithilfe einer Vorlage (dort: *Anwendungsfallspezifikationsschablone*) angelegt werden:
  »Als <Rolle beschreiben>
  möchte ich <Ziel/Wunsch eintragen>,
  um <erwarteter Nutzen>«
- Die beiden folgenden Beispiele zeigen einen alternativen Aufbau aus jeweils einer Überschrift und einem einzigen Satz:
  - Anwendung starten: Die Anwendung startet, indem sie das zuletzt bearbeitete Dokument des Anwenders öffnet, damit der Anwender Zeit spart.
  - Anwendung schließen: Wenn der Anwender die Anwendung beendet, erscheint eine Anfrage, ob das bearbeitete Dokument gespeichert werden soll, damit Änderungen nicht verloren gehen.

Anforderungspriorisierung

Priorisierung

Alle Einträge im Anforderungsdokument sollten priorisiert sein

- Wichtige Anforderungen können als erste umgesetzt werden und sind in jedem Fall Bestandteil der Produktversion
- Wichtige Anforderungen können frühzeitig dem Kunden und den Endanwendern vorgeführt werden
- Anforderungen nach Nutzen, Risiko und Kosten priorisieren:

Kriterium	Erläuterung
Wert/Nutzen	Welchen Mehrwert schafft die Realisierung der Anforderung? (schwierig!)
Risiko	Welches (potenzielle) Risiko wird durch die Umsetzung der Anforderung beseitigt?
Kosten	Welcher Aufwand bzw. welche Kosten fallen bei der Realisierung der Anforderung an?

Kano-Modell (1)

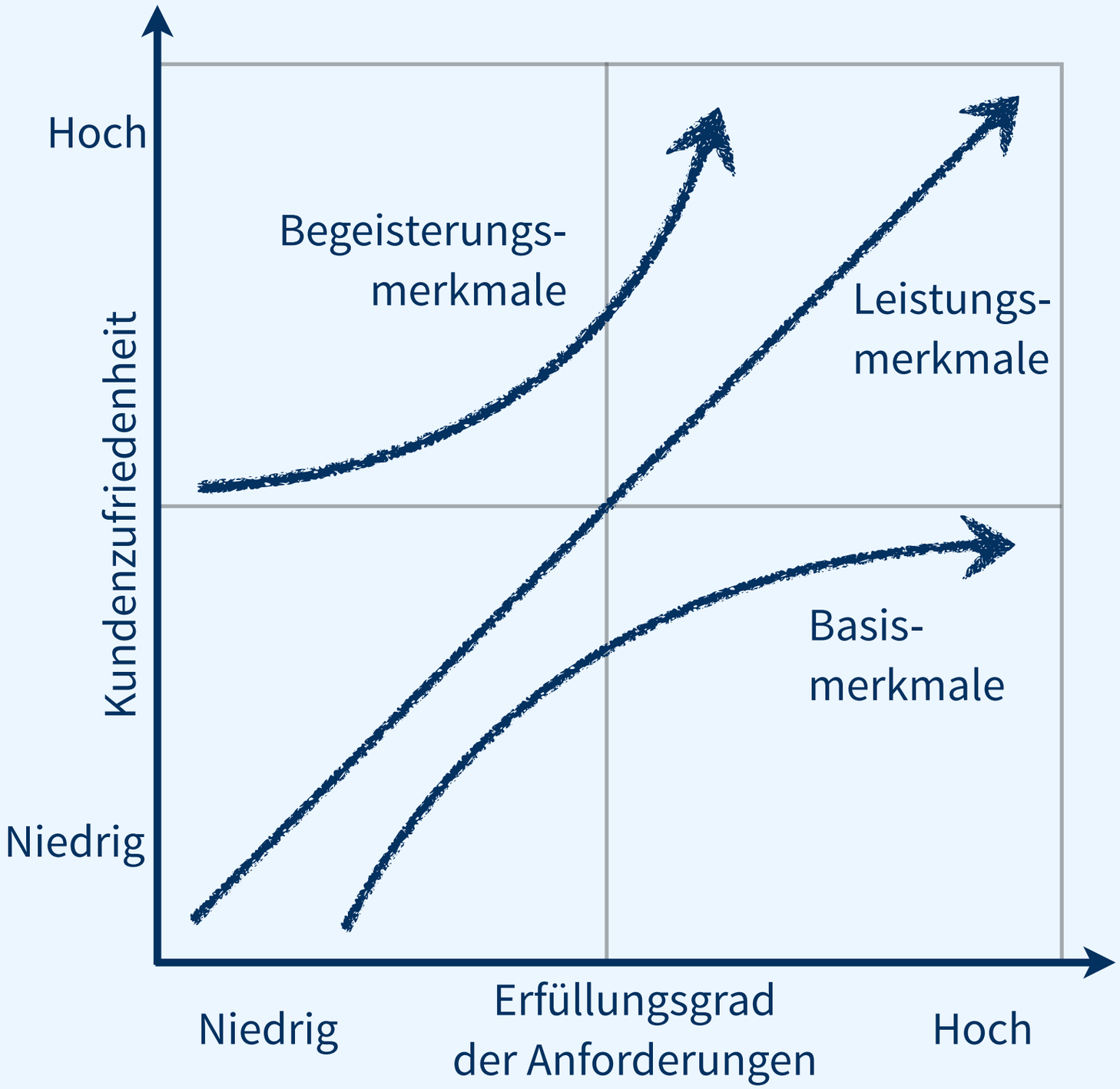
- Kano-Modell unterteilt Anforderungen in Basis-, Leistungs- und Begeisterungsmerkmale (Bestimmung des Nutzens)
- z.B.: *Basismerkmale* eines Mountain-Bikes:
  - funktionaler Rahmen
  - Laufräder
  - Bremsen
- z.B.: *Leistungsmerkmale* eines Mountain-Bikes
  - Federweg (je mehr, je besser)
  - Gewicht (je weniger, je besser)
- z.B.: *Begeisterungsmerkmale* eines Mountain-Bikes:
  - Kontrolle über die Federgabel *während des Fahrens*
  - Farbe und Qualität der Lackierung

Kano-Modell (2)

- *Basismerkmale* sind essenziell notwendig, um Software einsetzen und vertreiben zu können
  - aber: Kano-Modell sagt voraus, dass die Basisanforderungen rasch zu einer Stagnation der Kundenzufriedenheit führen!
- *Leistungsmerkmale* führen zu einem linearen Anstieg der Kundenzufriedenheit nach dem Motto »je mehr, je besser«
- *Begeisterungsmerkmale* führen zu hoher Kundenzufriedenheit
- Auch wenn *Basisanforderungen* essenziell notwendig sind
  - müssen diese nicht als erstes realisiert werden!

➡ Geschickte Kombination von Anforderungen aus den drei Kategorien hilft, die Kundenzufriedenheit und den Mehrwert zu optimieren!

Kano-Modell (3)



Quelle: Dr. Nariaki Kano (1978)

Danke

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