

Überblick über IT-Projektmanagement mit Schwerpunkt auf angewandter Agilität

Prof. Dr. André Köhler

Geschäftsführer SF Group

Über mich



Prof. Dr. André Köhler

- 1997-2003 Studium der Wirtschaftsinformatik
- 2012 Promotion in Informatik
- seit 2020 Professor für IT-Management an der IU Internationale Hochschule
- seit 2008 Gründer und Geschäftsführer der SF Group
 - SF Tech GmbH
 - summit GmbH

IT-Projekt und Softwareprojekt

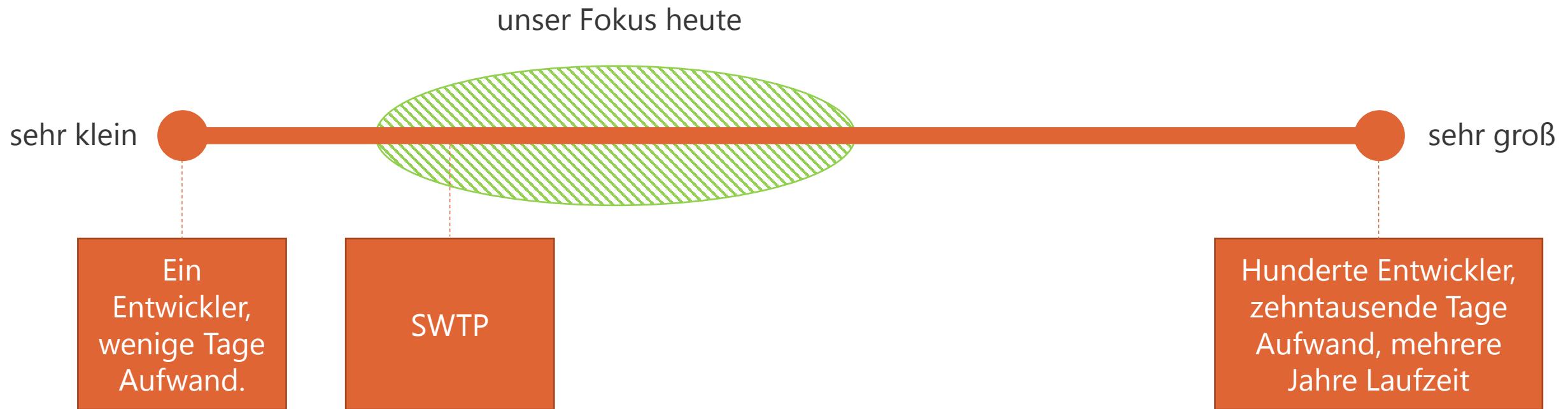
IT-Projekt

- Projekt zu Realisierung eines Vorhabens, dass nur durch den massiven Einsatz von IT zu realisieren ist.

Softwareentwicklungsprojekt (kurz: SE-Projekt, Softwareprojekt)

- IT-Projekt, das die Entwicklung von Software zum Gegenstand hat.

Spektrum von Softwareprojekten



Softwareprojekte managen

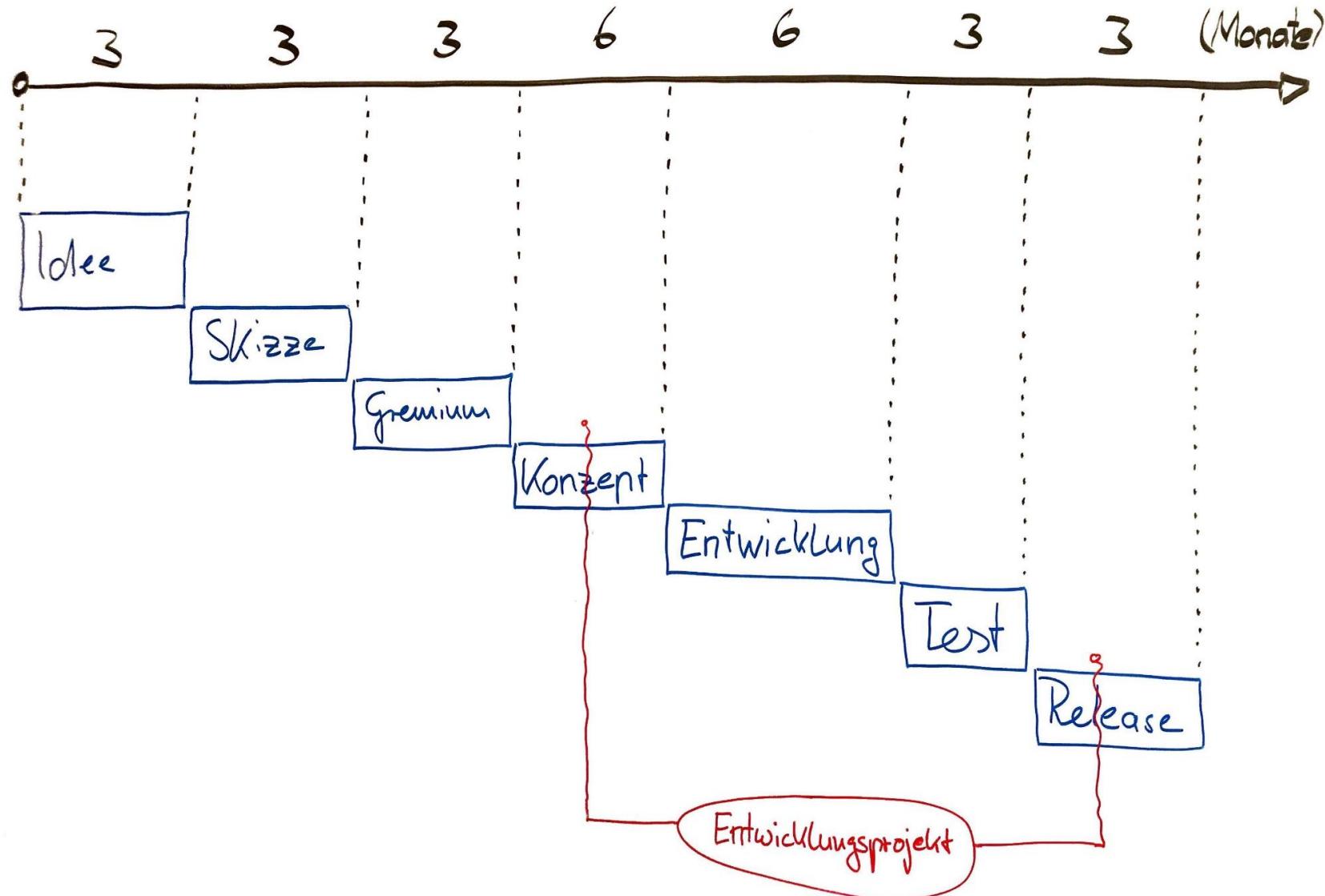
Ziele

- Projekt planen, steuern und kontrollieren, um Projektziele zu erreichen
- Typische Ziele für SE-Projekte:
 1. Alle Anforderungen umgesetzt
 2. Budget eingehalten
 3. Lieferzeitpunkt eingehalten

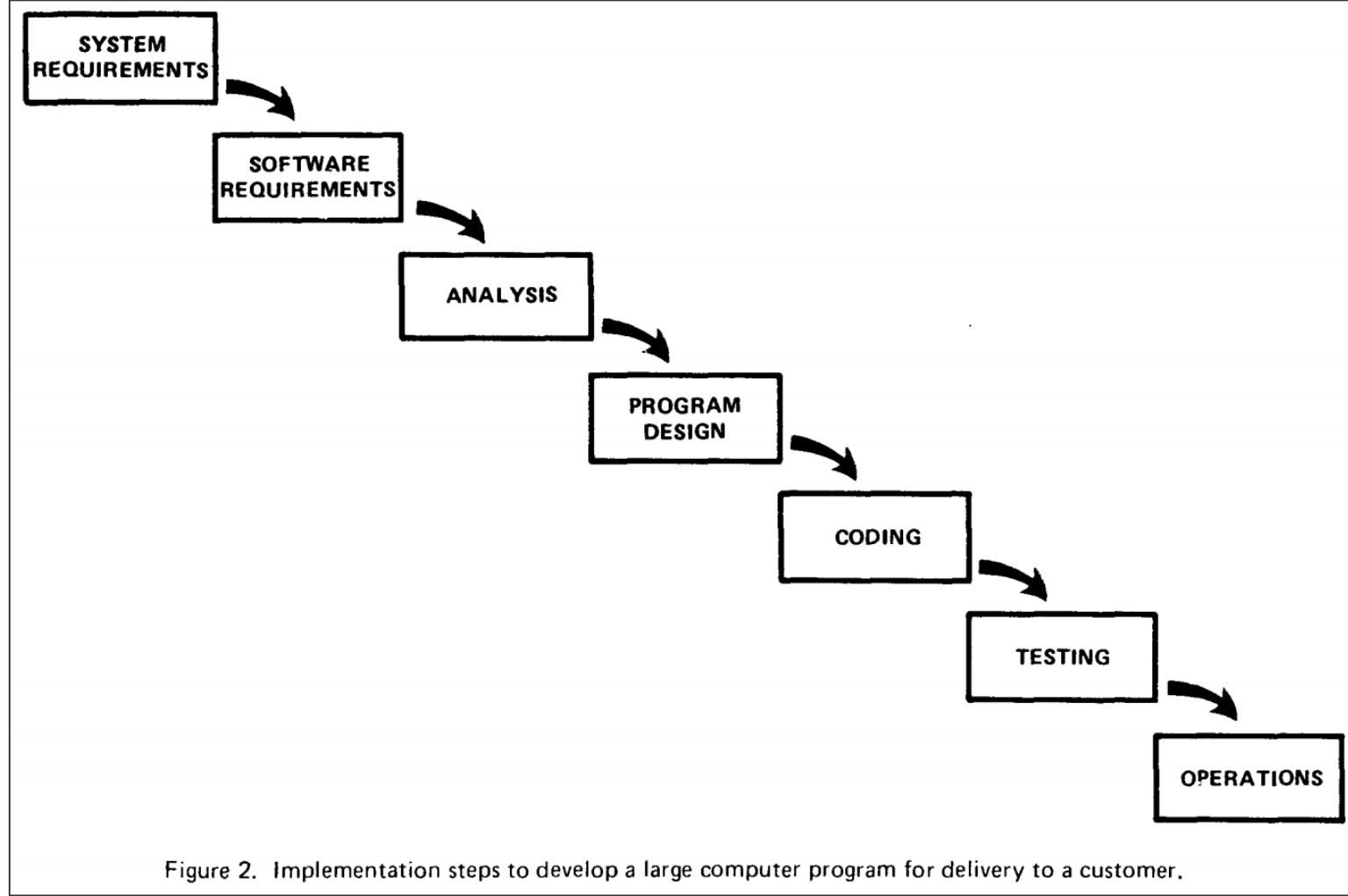
Bekannte Standards

- IPMA, PMI/PMBOK, PRINCE2, DIN 69901, ISO 21500, ...

Typischer Ablauf eines klassischen SE-Projekts

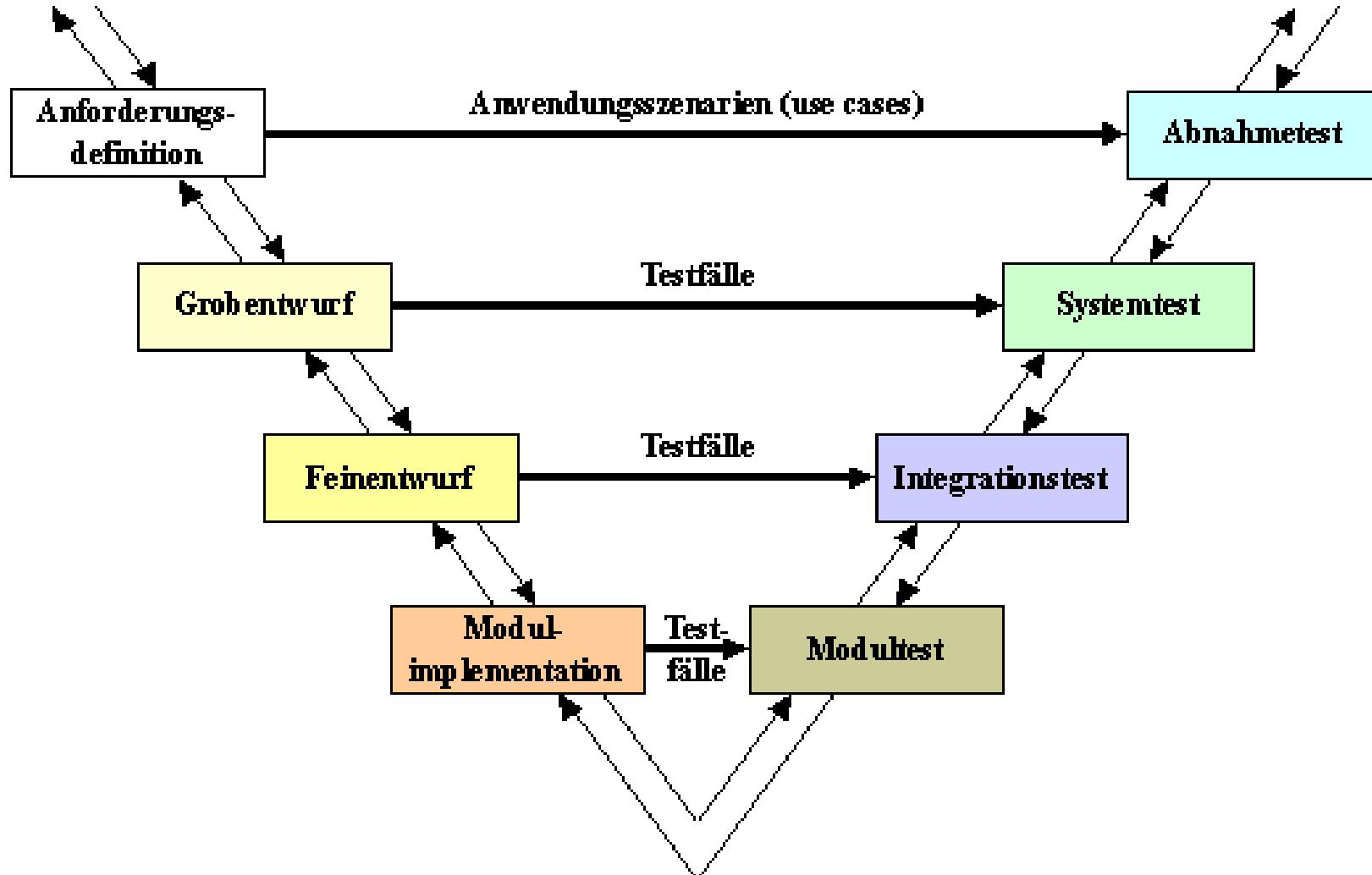


Das Wasserfallmodell



„Managing the Development of Large Software Systems“, Dr. Winston W. Royce, Proc. IEEE WESCON, 1970

Das V-Modell



Bildquelle: http://winf-wiki.fhslabs.ch/index.php?title=Datei:V-Modell_Resultate.gif

Typische Probleme mit wasserfallartigen Vorgehen

#1: Lange Gesamtdauer

#2: Illusion, dass die erzeugten Zwischenergebnisse vollständig / richtig sind

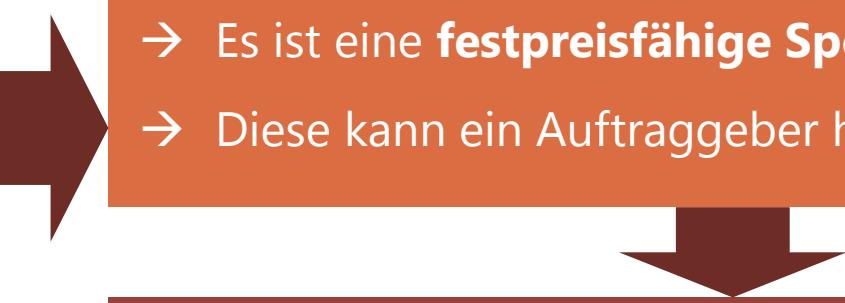
#3: Umgang mit unklaren und sich ändernden Anforderungen

#4: Risiken werden spät im Projekt erkannt

#5: „Das habe ich so aber nicht bestellt!“

#6: Wertorientierung und Investitionsschutz mangelhaft (falls mit dem Budget weniger Umfang erzeugt werden kann)

#7: ...

- 
- Es ist eine **festpreisfähige Spezifikation** nötig!
 - Diese kann ein Auftraggeber häufig nicht liefern.

Die Folge sind fachliche Unklarheiten im Projekt, daraus resultieren:

- Diskussion mit dem Auftraggeber, „Hätte man doch wissen müssen!“, unzufriedener Kunde
- Lieferzeitpunkt nicht haltbar, Budget wird überschritten, Projekt in den roten Zahlen
- Qualitätsprobleme auf Grund des massiven Lieferdrucks

Beispiele für gescheiterte IT-Projekte

- 2017 teilte die Bundesagentur für Arbeit den Stopp ein 2010 gestartetes Software-Projekt mit Titel ROBASO (Rollenbasierte Oberfläche) mit, das 14 verschiedene eigene Anwendungen auf einer Plattform bündeln sollte. Verbraten wurden rund 60 Millionen Euro.
- Die Deutsche Post DHL stoppte ihr Transformationsprojekt New Forwarding Environment (NFE) 2016, in dem die SAP zusammen mit der IBM individualisierte SAP-Module einführen hätte sollen. Summa summarum dürfte sich die „Entwicklungshilfe“ dafür auf 500 Millionen Euro belaufen haben.
- Da lief es bei Edeka besser: Ab 2007 versuchte er auf SAP umzusteigen. Das Projekt „Lunar“ sollte 200 Millionen Euro kosten, zum Schluss im Jahr 2012 waren es 350 Millionen Euro. Immerhin war es nicht gescheitert.
- 2018 stoppte Lidl nach sieben Jahren die Einführung des Warenwirtschaftssystems „Elwis“, ein auf den Discounter zugeschnittenes ERP-Gesamtsystem. Auch hier sollen 500 Millionen Euro verbrannt worden sein.

Herausforderungen im IT-Projektmanagement



Quelle: <http://www.cms-garden.org/de/file/schaukel-illustration>

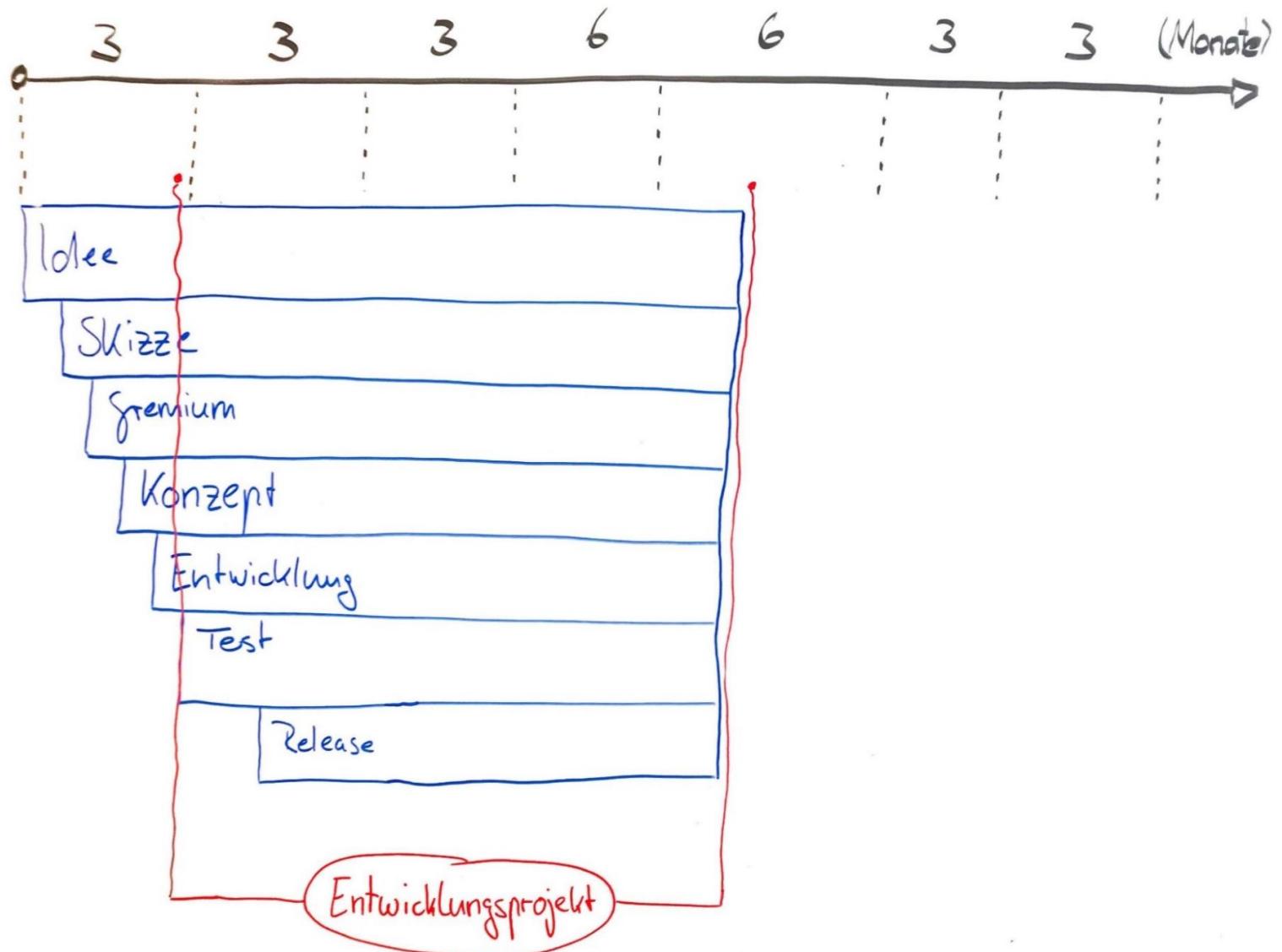
Die agile Idee

Wenn **eine festpreisfähige Spezifikation nicht herstellbar** ist, dann sollte man lieber:

- Zu Beginn nur sehr grob spezifizieren, dafür aber sehr schnell.
- Dafür schneller mit der Entwicklung beginnen.
- Durch ständiges Besprechen des Entwicklungsstands mit dem Auftraggeber die Spezifikation während des Projekts erarbeiten.

So zu arbeiten, bedeutet agil zu arbeiten.

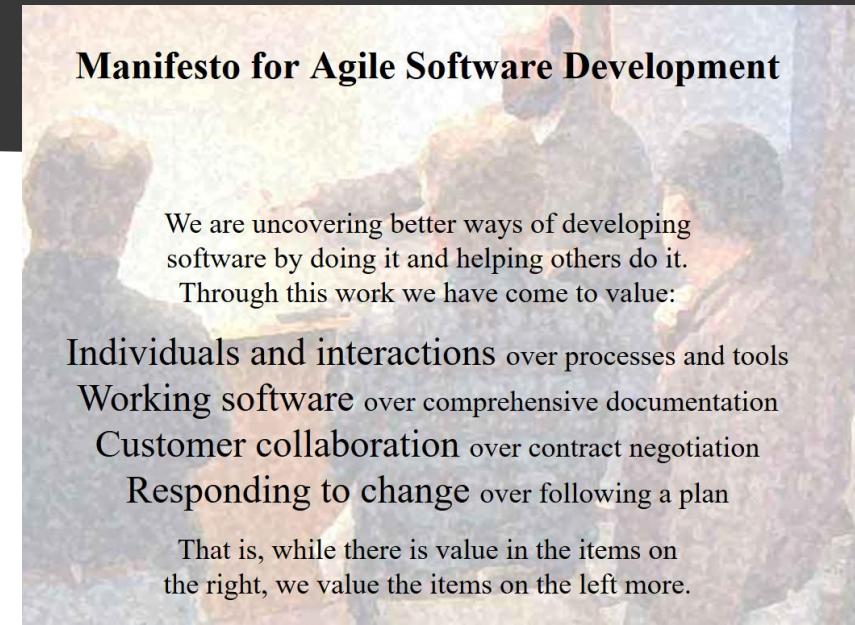
Transformation von klassisch nach agil



Agile Methoden

Agiles Manifest

- 2001
- agilemanifesto.org
- 4 Richtlinien, 12 Prinzipien



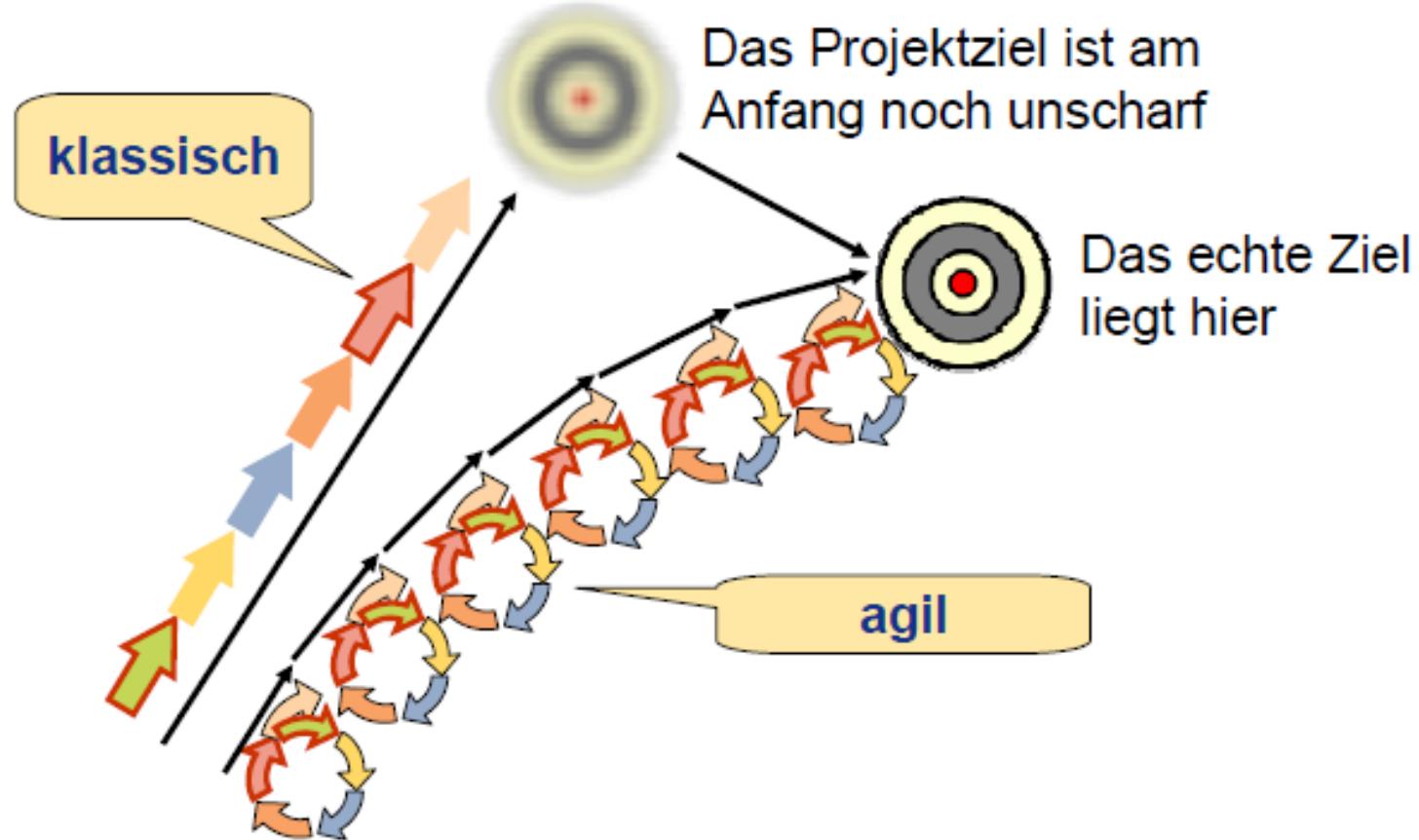
Agile Methoden für IT-Projekte

- Projektportfolio-/Multiprojektmanagement-Level: **SAFe, Nexus, LeSS, DAD, ...**
- Projektorganisations-Level: **SCRUM**
- Entwickler-Level: **XP (TDD, Pair Programming, ...)**

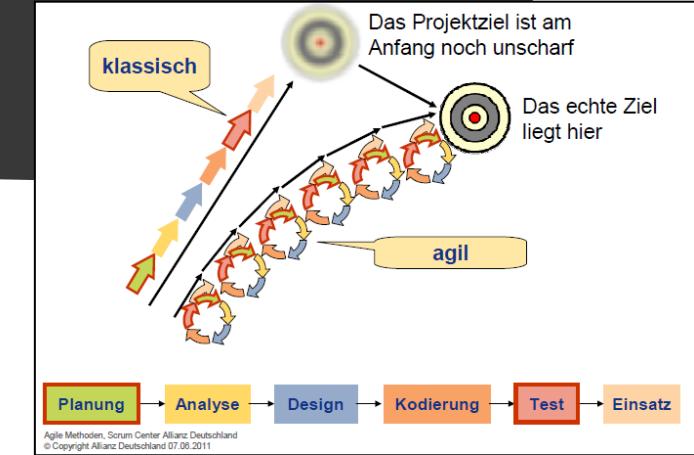
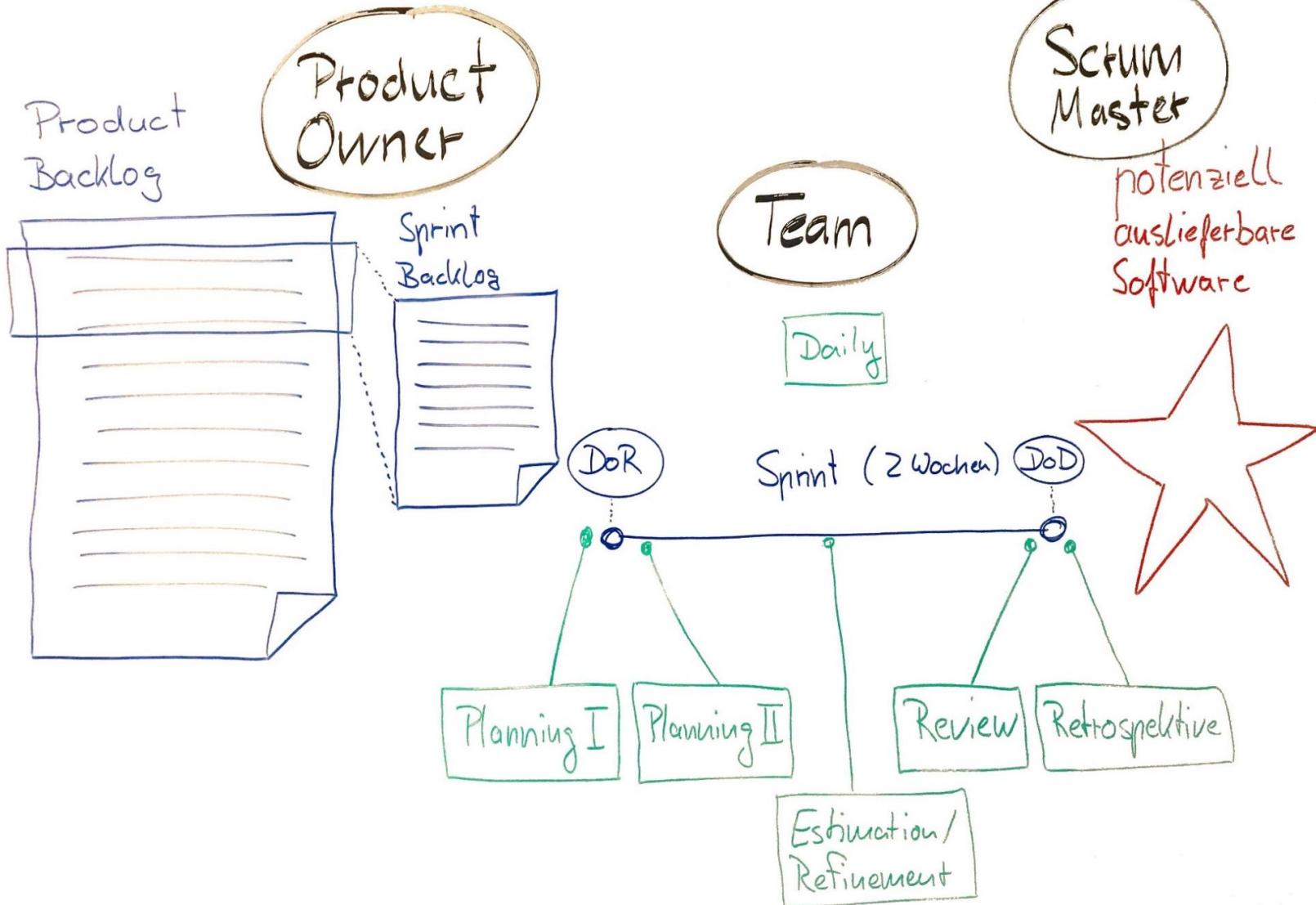
Andere Methoden der „Agil-Kultur“

- Prozessdesign: **Lean, Kanban**
- Produktdesign: **Design Thinking, Lean Start-Up**
- Unternehmensorganisation: **Holacracy und Soziokratie**

SCRUM ist ein iterativ-inkrementelles Vorgehen



Agile Softwareentwicklung am Beispiel von SCRUM



Product Backlog und Sprint Backlog in JIRA

Projects / Beyond Gravity

Board

Epics: +3

4 days remaining

Complete sprint

GROUP BY Choices

TO DO	IN PROGRESS	IN REVIEW	DONE
Implement feedback collector NUC-205	Update T&C copy with v1.9 from the writers guild in all products that have cross country compliance NUC-213	Multi-dest search UI web NUC-338	Quick booking for accomodations - web NUC-336
Bump version for new API for billing NUC-206	Tech spike on new stripe integration with paypal NUC-215		Adapt web app no new payments provider NUC-346
Add NPS feedback to wallboard NUC-208	Refactor stripe verification key validator to a single call to avoid timing out on slow connections NUC-216		Fluid booking on tablets NUC-343
	Change phone number field type to 'phone' NUC-217		Shoping cart purchasing error - quick fix required. NUC-354

Rollen

Product Owner

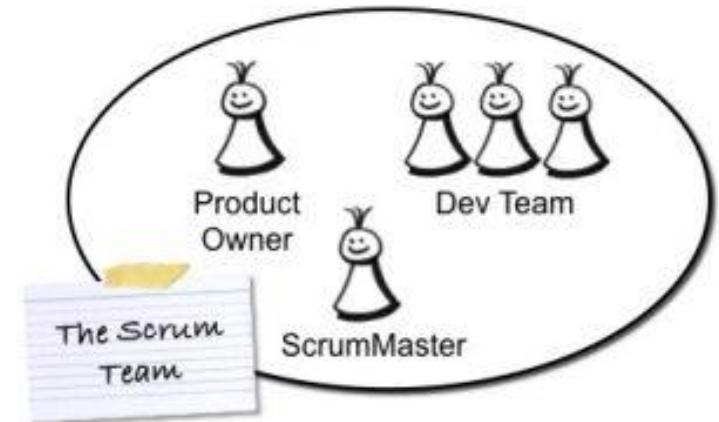
- ... sorgt dafür, dass das Team das Richtige herstellt. I.d.R. sollte das Software sein, die für den Product Owner bzw. den Auftraggeber einen möglichst hohen Nutzen erzeugt.

Team

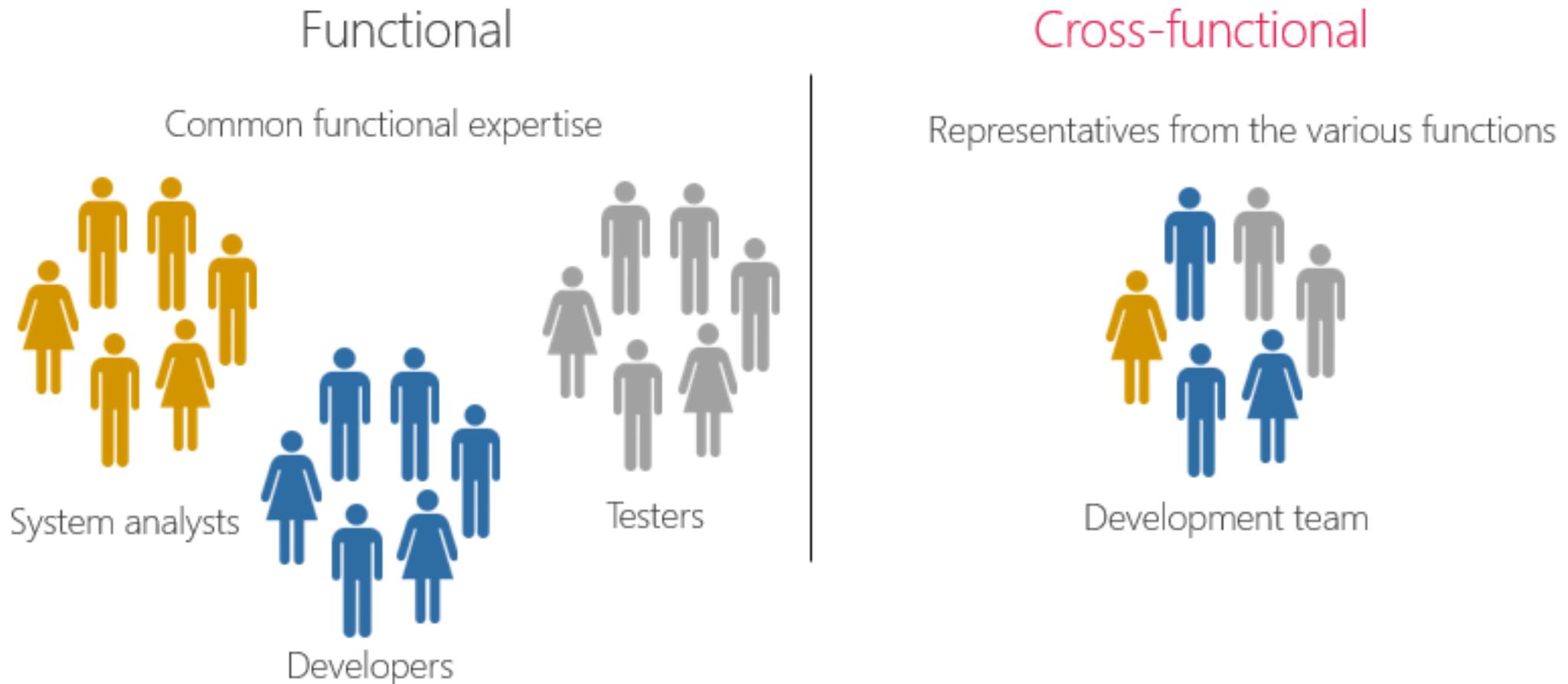
- ... stellt die beste technische Lösung zur Erfüllung der Anforderungen des Product Owners her.

Scrum Master

- ... sorgt dafür, dass das Team möglichst störungsfrei arbeiten kann. Mit seiner Arbeit steigert er kontinuierlich die Effizienz (Erhöhung des Outputs bei gleichem Mitteleinsatz) des Teams.



Cross-funktionales Team



Artefakte

Product Backlog

- Priorisierte Liste aller Anforderungen, die im Projekt umgesetzt werden könnten.

Sprint Backlog

- Liste der Anforderungen aus dem Backlog, die in einem Sprint umgesetzt werden sollen.

Potenziell auslieferbare Software

- Ergebnis eines jeden Sprints.

Definition of „done“ / Definition of „ready“

- Vereinbarung zwischen dem Product Owner und dem Team, was eine User Story (Schriftform) / User Story (Software) alles erfüllen muss, um als „fertig“ zu gelten.

Sprint

- Festgelegter, unveränderlicher Zeitraum, in dem alle Einträge im Sprint Backlog abgearbeitet werden.

Meetings

Planning I

- Product Owner und Team erstellen gemeinsam das Sprint Backlog. Start des Sprints.

Planning II

- Das Team zerlegt die User Stories des Sprint Backlogs in Entwickleraufgaben.

Estimation/Refinement

- Der Product Owner stellt neue oder veränderte Anforderungen vor und lässt den Umsetzungsaufwand vom Team schätzen.

Review

- Das Team demonstriert am Sprintende die umgesetzten User Stories anhand der potenziell auslieferbaren Software.

Retrospektive

- Das Team spricht darüber, wie der letzte Sprint gelaufen ist, was man in Zukunft besser machen könnte und plant konkrete Aufgaben, um diese Verbesserungsideen umzusetzen.

Daily

- Das Team kommt einmal täglich kurz zusammen, um über die aktuellen Aufgaben zu sprechen.

User Stories

Was ist eine User Story?

- Eine kurze (ein Satz) Beschreibung einer Anforderung an ein Softwaresystem.

Warum User Stories?

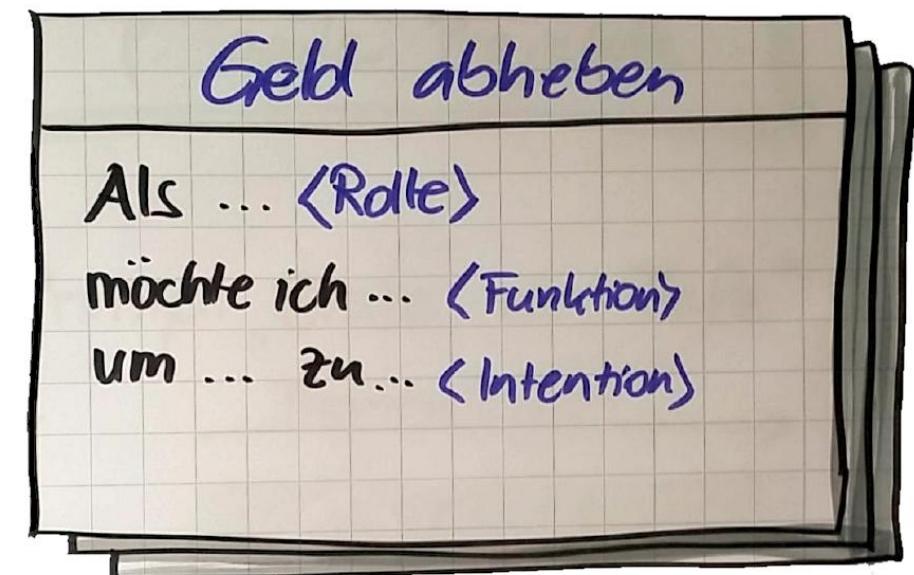
- Weltweit und seit vielen Jahren bewährtes Mittel zur Anforderungsbeschreibung in agilen Projekten.

Warum sind User Stories gut?

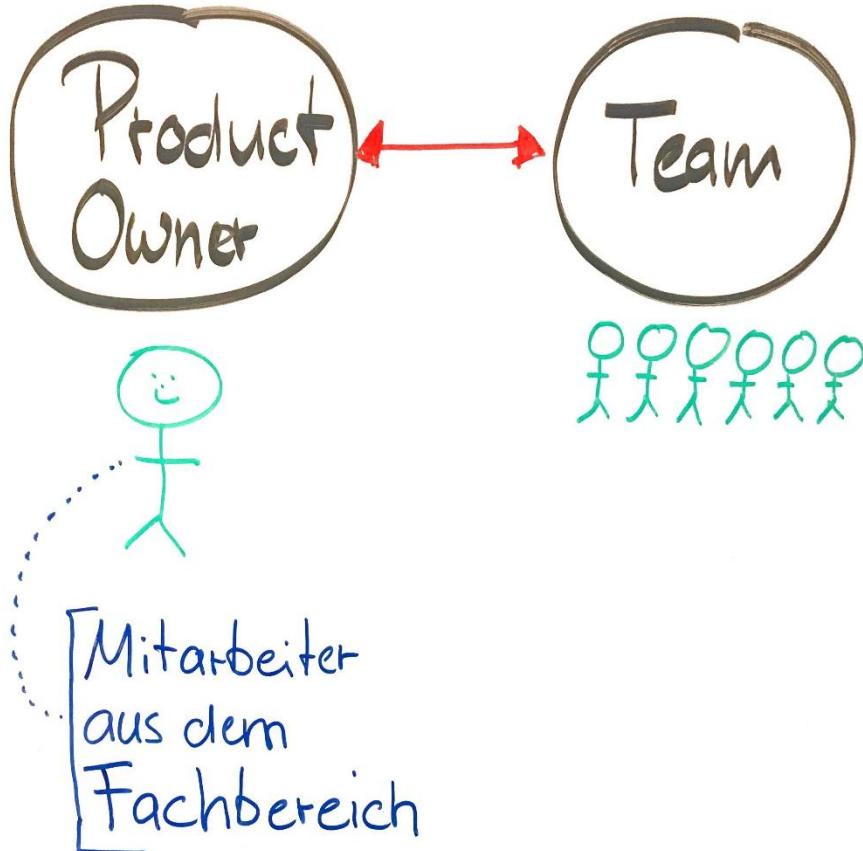
- Es wird ein konkreter Mehrwert aus Sicht des Anwenders beschrieben.
- Nach der Umsetzung hat die Software einen erhöhten Business-Nutzen.

Wie sieht eine User Story aus?

1. Die User Story selbst wird mit einer Satzschablone formuliert:
Als <Rolle> möchte ich <Ziel>, so dass <Grund für das Ziel>.
2. Kommunikation / weitere Beschreibungen.
3. Akzeptanzkriterien.



Der Product Owner in der Praxis I



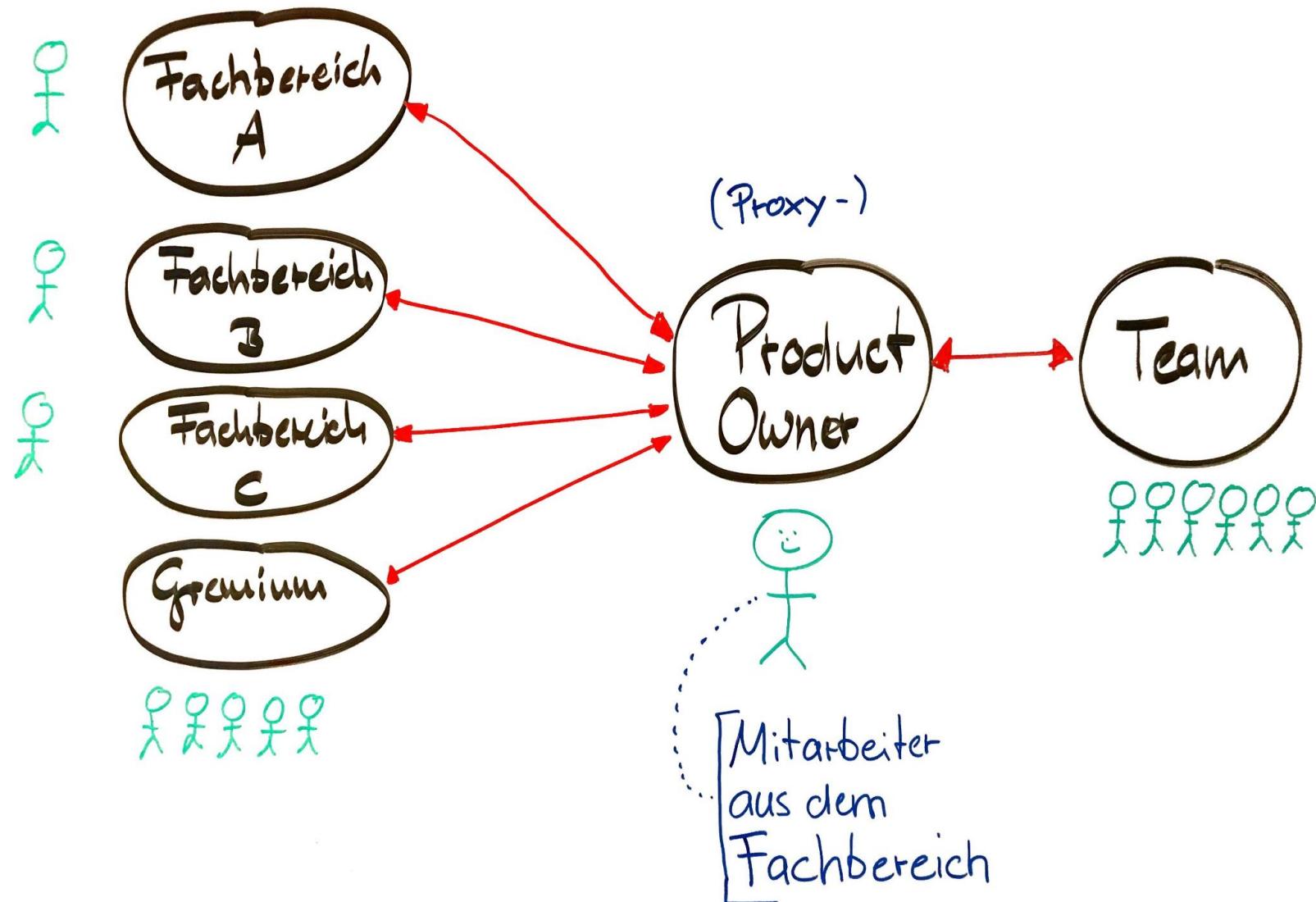
Aufgabe des Product Owners allgemein

- sicherstellen, dass ein möglichst großer Business-Nutzen mit dem Softwareprodukt/-projekt erzielt wird

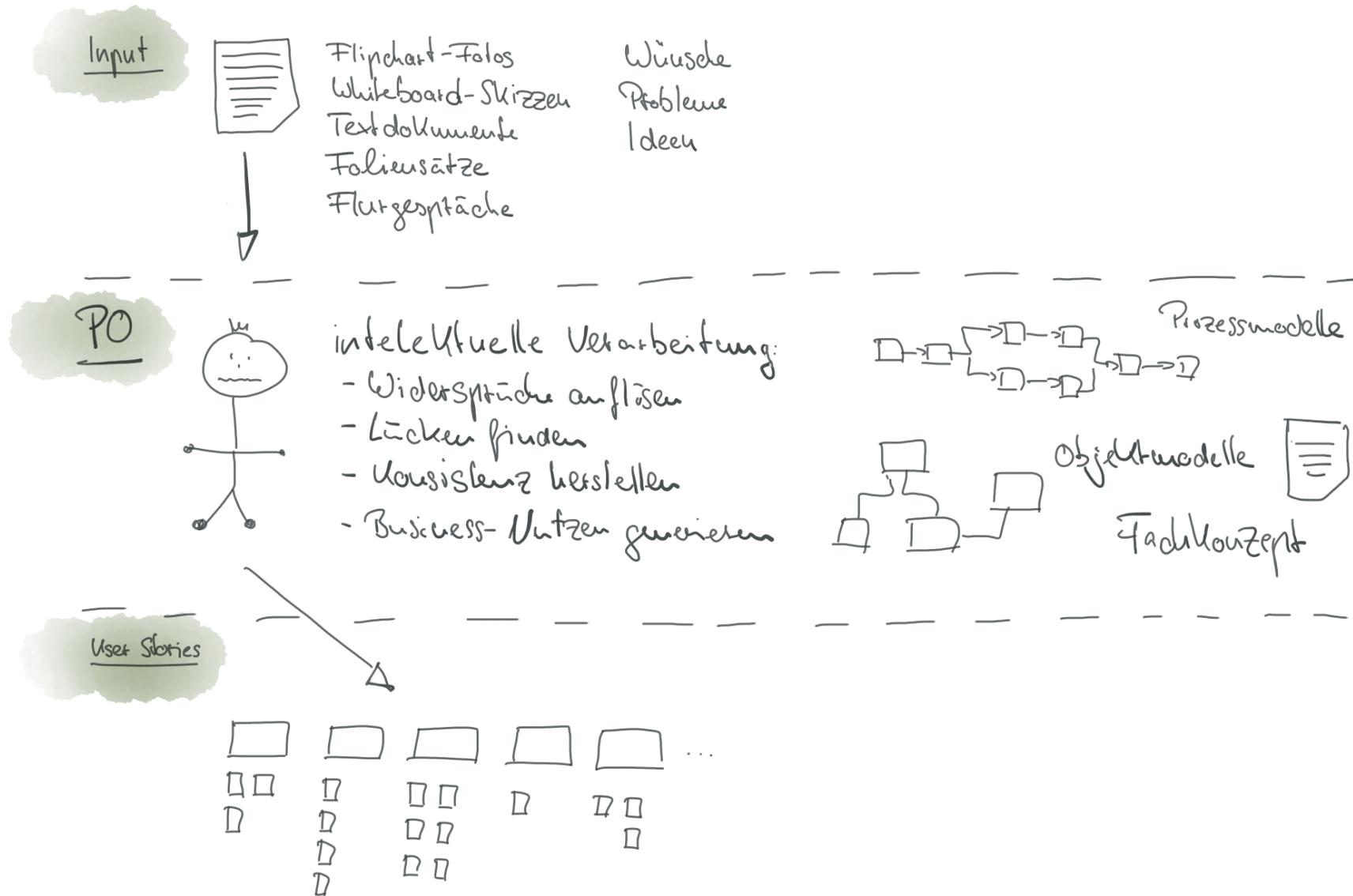
Aufgaben des Product Owners im Detail

- Product Backlog initial bereitstellen und permanent aktualisieren
- Gesamtheit der fachlichen Anforderungen überblicken, Konsistenz und Sinnhaftigkeit sicherstellen
- Vorbereitung / Führung der Meetings „Planning I“ und „Review“
- permanente Ansprechbarkeit für das Team, um z.B. Rückfragen zu beantworten oder Implementierungsalternativen zu entscheiden
- permanentes Testen des aktuellen Entwicklungsstands

Der Product Owner in der Praxis II



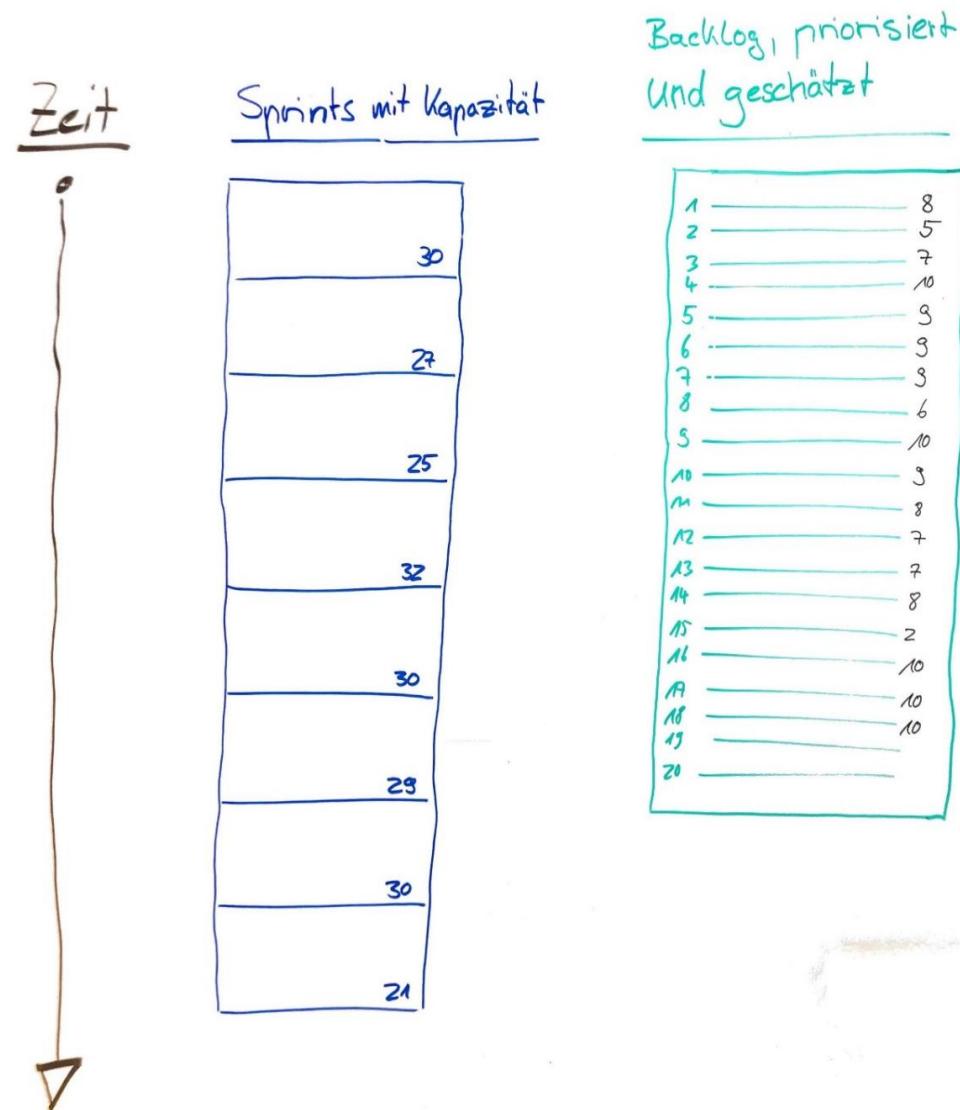
Der Product Owner in der Praxis III



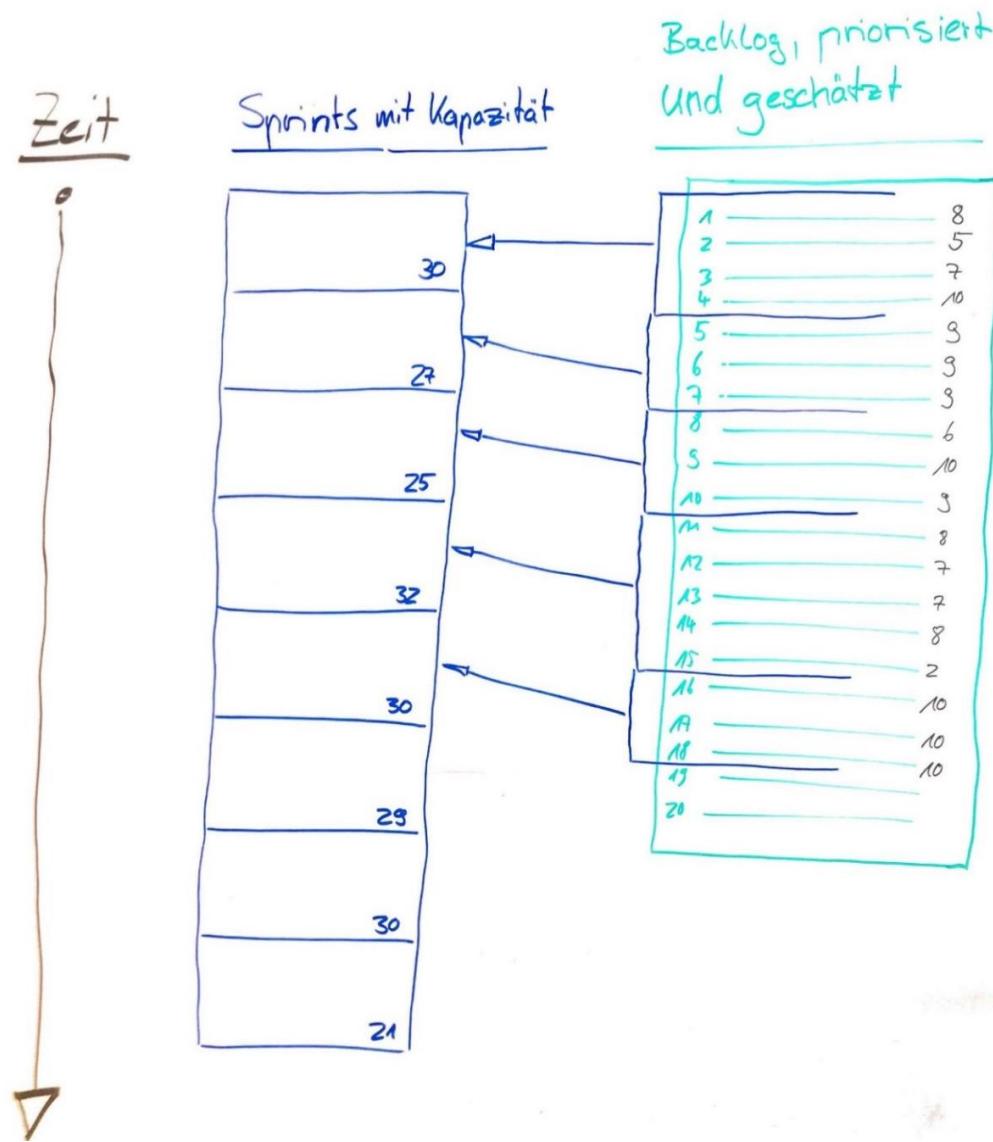
Schätzen in agilen Projekten

- Schätzen des Gesamtprojekts in einer frühen Phase
 - Ziel: grobe Aufwandsbestimmung zur Entscheidung über eine Durchführung sowie Planung eines Budgets und eines Zeitrahmens.
 - Kennzeichen: Es liegen nur wenige, grobe Informationen vor.
 - Agile Schätzmethode: **Magic Estimation**
 - Schätzen von einzelnen User Stories im Projektverlauf
 - Ziel: Aufwandsschätzung für die Planung der Sprints und die Erstellung des Releaseplans.
 - Kennzeichen: Die User Story wurde entwicklungsreif spezifiziert.
 - Agile Schätzmethode: **Planning Poker**
- Menschen können besser relativ als absolut schätzen.
 - Um relativ zu schätzen, ist die Verwendung einer passende Skala hilfreich, z.B.
 - T-Shirt-Größen: XS S M L
 - Story-Points
 - ...

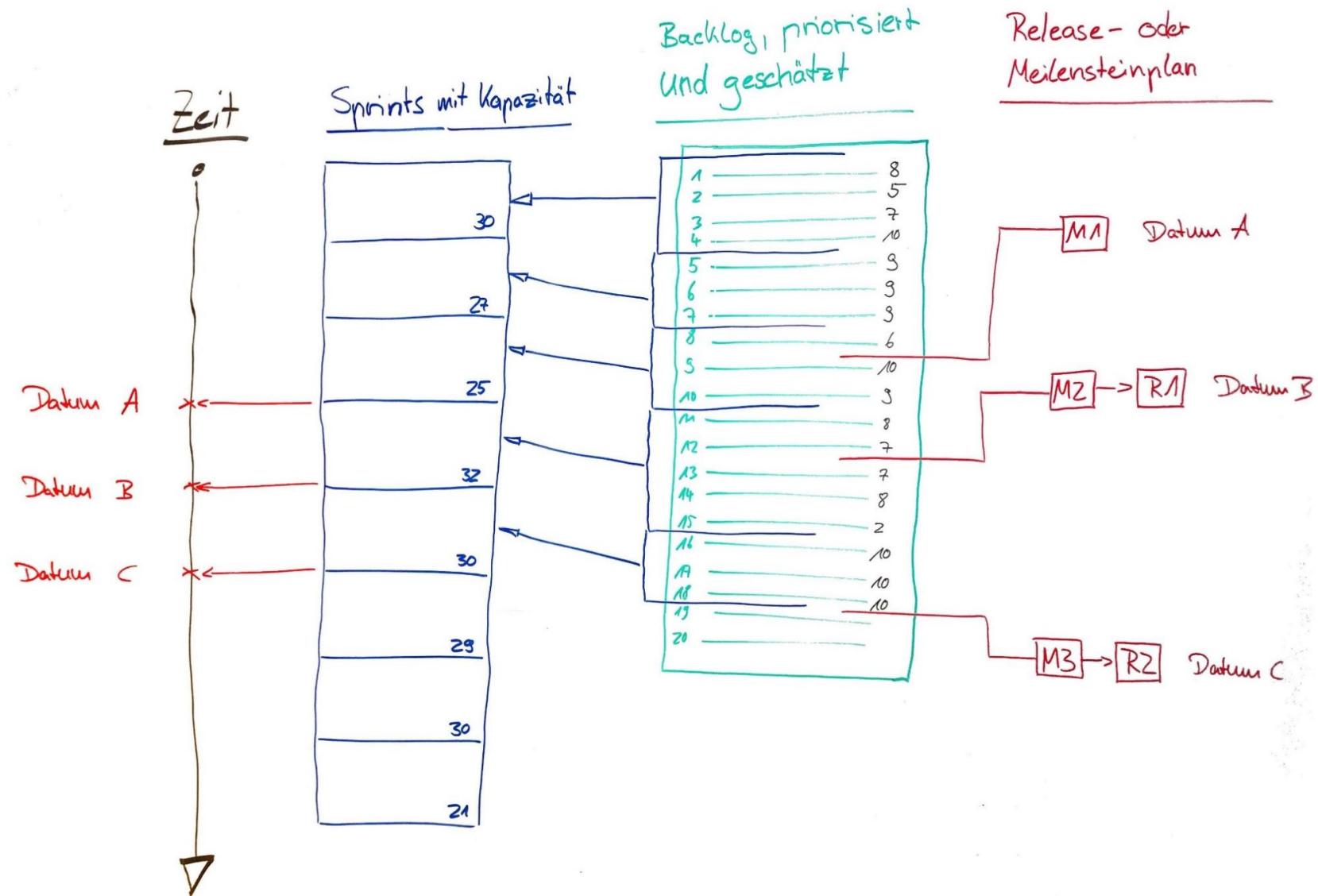
Planung von agilen Projekten I



Planung von agilen Projekten II



Planung von agilen Projekten III



Explorationssprint in SCRUM

- Ein Explorationssprint (auch Spike genannt) dient der Untersuchung von unklaren Anforderungen, Technologien oder Risiken.
- Klärung von Unsicherheiten und offenen Fragen, Erkundung neuer Technologien oder Architekturen, Reduktion technischer oder funktionaler Risiken.
- Liefert oft eine Analyse, einen Proof of Concept oder eine Entscheidungsvorlage.
- Durchführung kann sinnvoll sein zu Beginn eines Projekts, vor der Einführung neuer oder komplexer Features, bei unklaren technischen Anforderungen, bei der Wahl zwischen mehreren Lösungsansätzen.
- Beispiel: Untersuchung, ob eine bestimmte API oder Technologie für ein neues Feature geeignet ist.

Umgang mit technischen Schulden / Refactoring

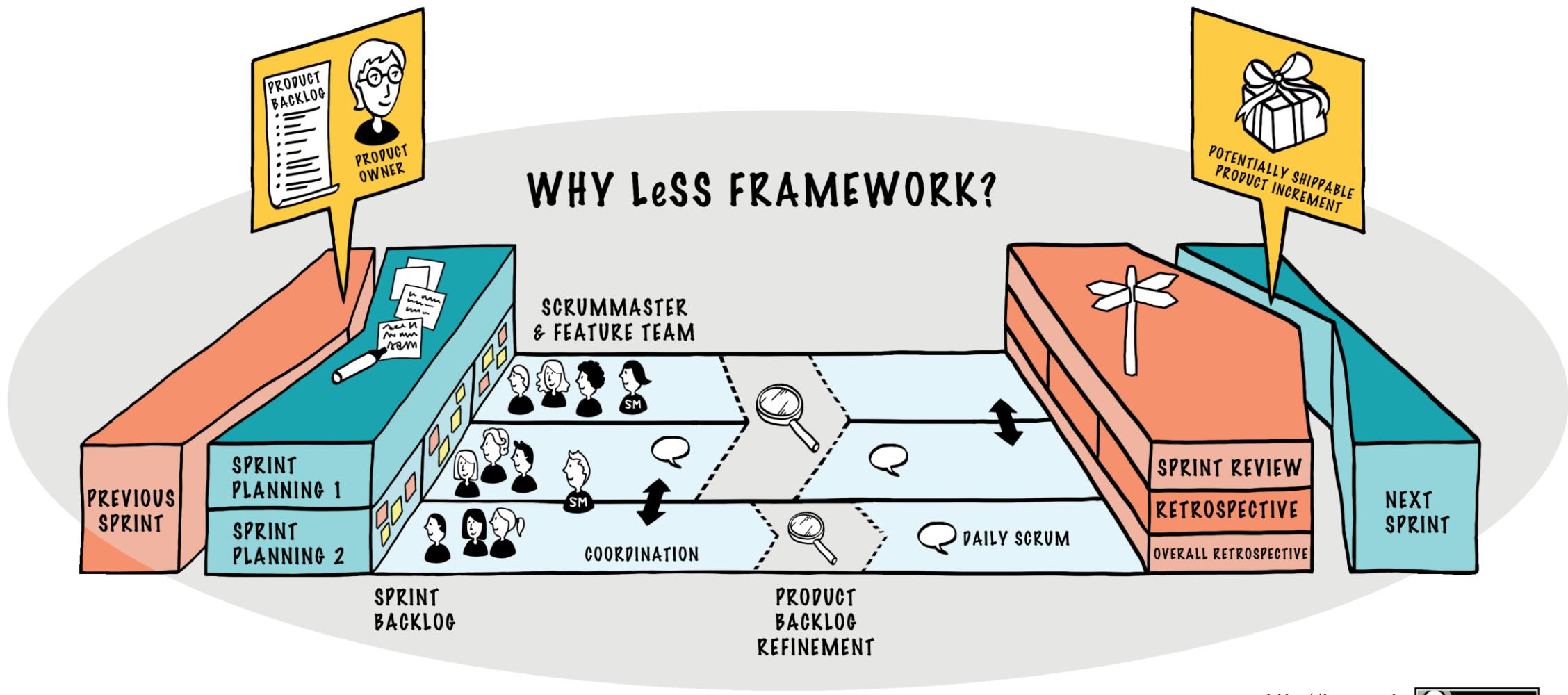
- **Technische Schulden** = Kompromisse bei der Code-Qualität oder Architektur, um kurzfristig Zeit zu sparen; unbezahlte „Schulden“, die die zukünftige Entwicklung verlangsamen
- **Ursachen:** Zeitdruck durch Fokus auf schnelle Lieferungen (Sprints), unklare/geänderte Anforderungen, mangelnde Refactoring-Zeit (technische Verbesserungen werden aufgeschoben)
- **Folgen:** Wartungskosten steigen (mehr Bugs und komplizierte Änderungen), verlangsamte Entwicklung (neue Features werden schwieriger umzusetzen), Rückgang der Code-Qualität (Komplexität und Instabilität nehmen zu)
- **Bewältigungsstrategien:** Refactoring fest einplanen (Zeit für technische Schulden in jedem Sprint reservieren), Technische Schulden priorisieren (Tickets planen und im Backlog verfolgen), Refactoring-Sprint in regelmäßigen Abständen, Team-Verantwortung für Code-Qualität und nachhaltige Entwicklung (Pfadfinderregel)

Konsequenzen agiler Entwicklung

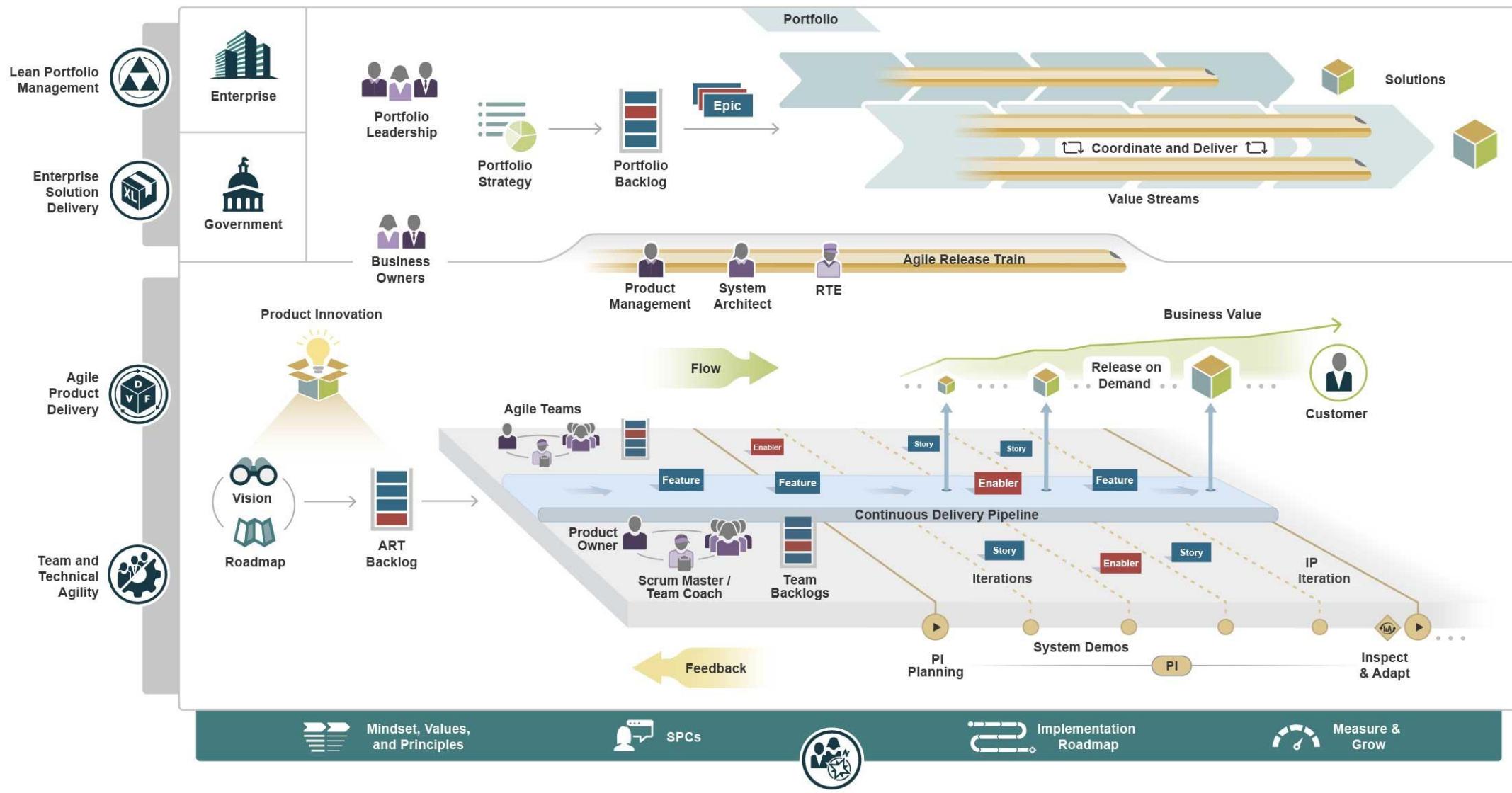
1. Es wird Entwicklungsanteile geben, die rückblickend unnütz waren.
2. Es wird – zu Gunsten der schnellen Entwicklung von Features – keine optimale Softwarearchitektur entstehen (es werden sog. technische Schulden aufgebaut). Es ist ein regelmäßiges Refactoring notwendig.
3. Die genauen Kosten können zu Beginn des Projekts nicht prognostiziert werden, ODER der finale Funktionsumfang kann bei festem Budget zu Projektbeginn nicht angegeben werden.
4. Es wird Software gebaut, die den tatsächlichen Wünschen des Kunden entspricht. Welche das sind, findet der Kunde im Detail während des Projekts heraus.

Die Annahme ist, dass die Summe dieser Konsequenzen ein Projektergebnis hervorbringt, das besser ist, als wenn man das Projekt „klassisch“ organisiert hätte.

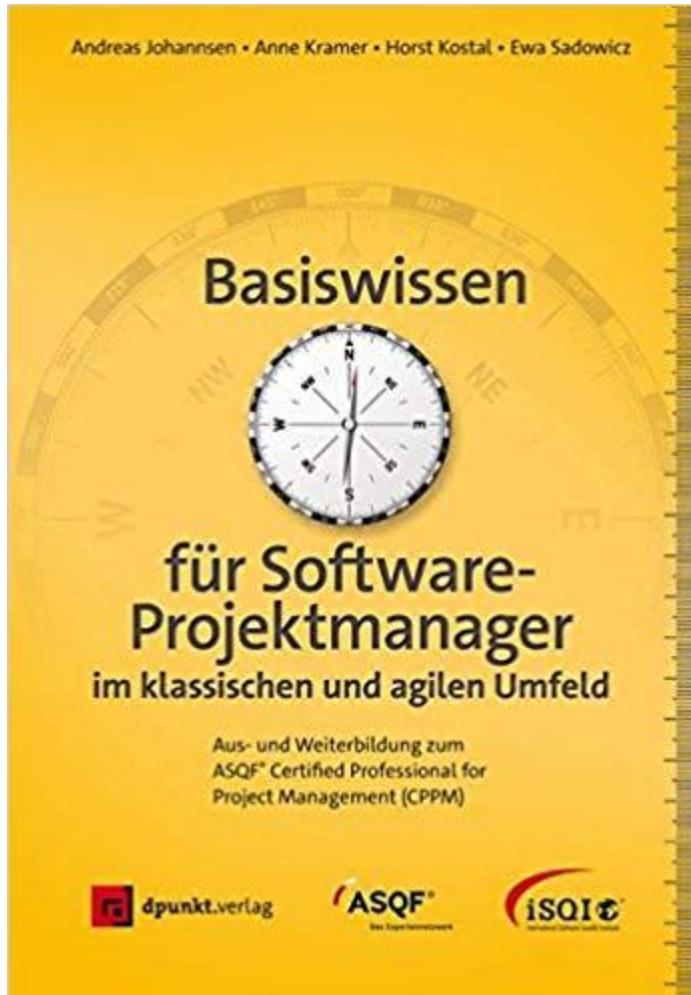
Agile Entwicklung in sehr großen Softwarprojekten: Large-Scale SCRUM (LeSS)



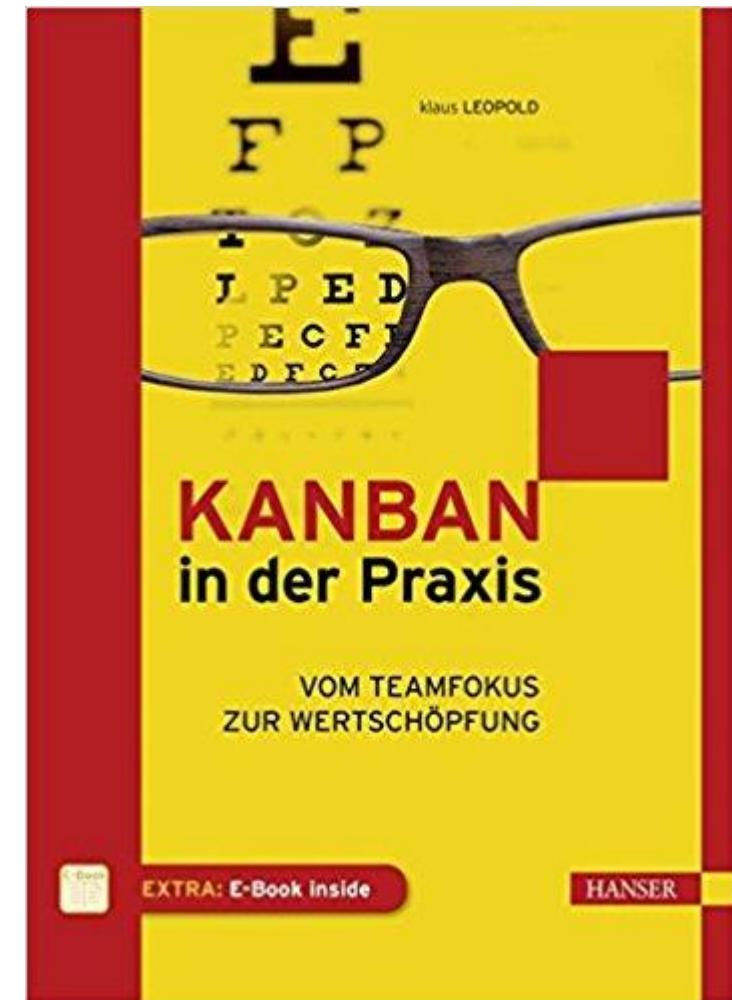
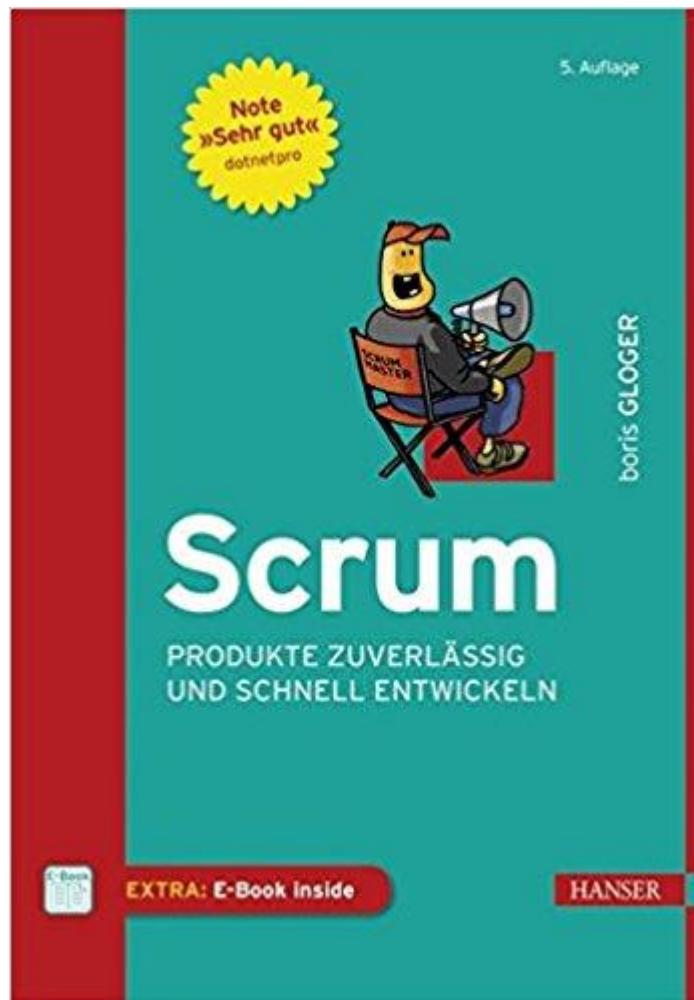
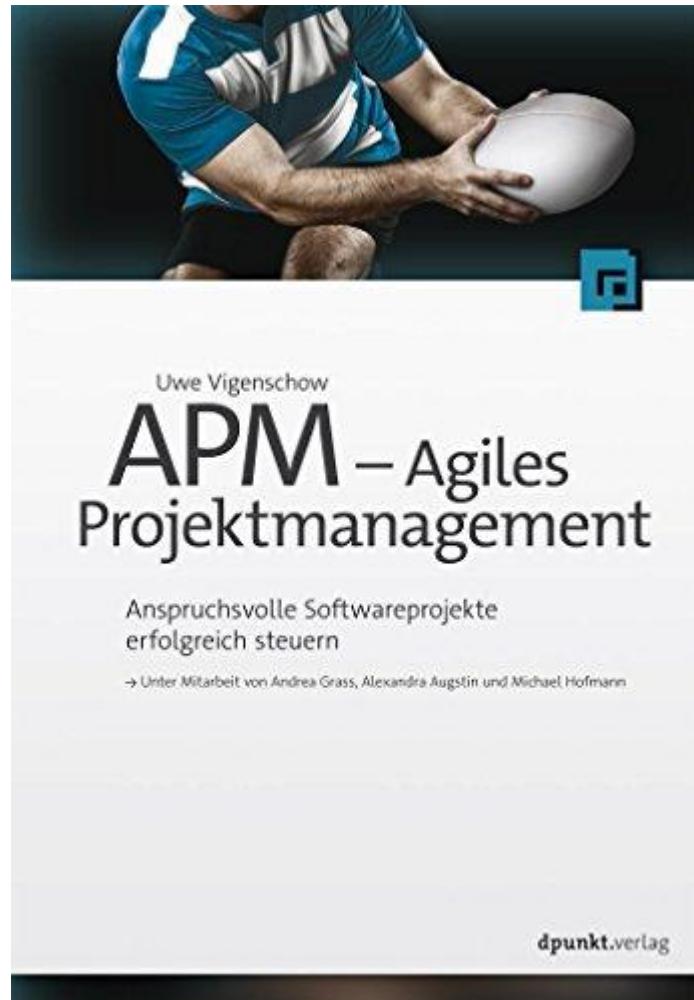
Scaled Agile Framework (SAFe)



Literaturempfehlung I



Literaturempfehlung II



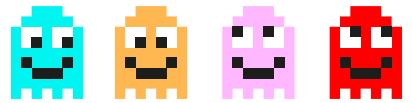
Kontakt

Prof. Dr. André Köhler

SF Group
Hainstraße 16 | 04109 Leipzig

T +49 341 23822 900

E koehler@sfgroup.de
I www.sfgroup.de



Werkstudent (m/w/d) Softwareentwicklung

- Du entwickelst in echten Kundenprojekten moderne, webbasierte Systeme für mittelständische Unternehmen (lokal bis international).
- Du arbeitest in einem agilen Team.
- Tech-Stack: Java, Spring Boot, node.js, Angular, TypeScript, git, Cloud-Architekturen
- Büro in der Leipziger Innenstadt
- Du wirst Teil eines supersympathischen Teams! ☺



Mail an Denise
recruiting@sfgroup.de



Telefon
0341 238229-02



summit

CodeBuzz

Das Tech-Festival für Developers in Leipzig

JUNI



2025

www.codebuzz.de