Wasserfall vs. SCRUM

Duale Hochschule Baden-Württemberg (DHBW) Karlsruhe
Studiengang Wirtschaftsinformatik, WWI17B2

Marc Schanne
marc@schanne.org

https://moodle.dhbw.de/course/view.php?id=3132

- Entwicklungsprozess
  - Teamarbeit 20. + 27. Juni
  - Einzelinterviews 30.06.
  - schriftl. Reflexion bis 12.07./12h00
- klassisch bzw. Wasserfall
  - Phasen und Artefakte
- agil bzw. SCRUM
  - Agiles Manifest
  - User Stories, Abschätzen
  - Sprints mit Grooming
  - Review + Retrospective
  - Definition of Done
- Teams

Portfolio Workshops

- Refactoring-Workshops 20. und 27. Juni 2020
  - Entwicklungsprozesse praktisch einsetzen
    - klassisch (Wasserfall) und agil (SCRUM)
    - gegebenes Zeitfenster: 9h30 – 15h30
  - 1h Einführung (Lastenheft vs. Epic/User Stories)
  - 4h Refactoring
    - Meilensteine bzw. Sprints, Pausen selbst planen
  - 1h gegenseitige Team-Updates von je zwei Teams

Portfolio Bewertung SE II

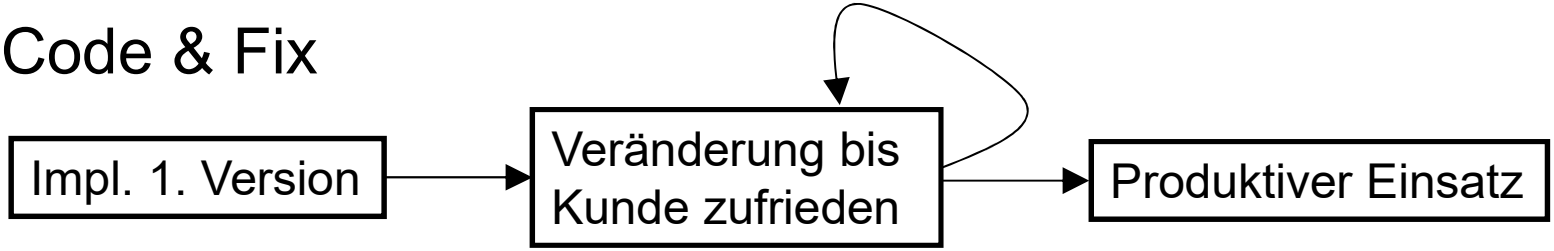
- Einführung in Sprachen und Sprachkonzepte
  - 15 %
- Praxis und Coding im Team
  (Artefakte mit Historie in Github)
  - 30 %
- Einzelinterview
  - 40 %
- Schriftl. Reflexion
  (mind. 2 Seiten in vollst. Sätzen,
  max. Schriftgr. 12, Zeilenabst. max. 1,5)
  - 15 %

Refactoring vs. Entwicklung!

- Wichtige Anmerkung!
  - Eigentlich sind Refactorings nicht Teil der „normalen“ Entwicklungsarbeit!
  - Refactorings entsprechen in der Regel dem Abbau von technischer Schuld („technical debts“).
  - Diese Arbeit ist natürlich zu erledigen und bedingt normalerweise die Reduzierung der Produktivität des Entwicklerteams.
  - In unserem Szenario wird die Produktivität durch die geleistete Entwicklungsarbeit = Refactoring selbst bestimmt.

vor Wasserfall-Modell

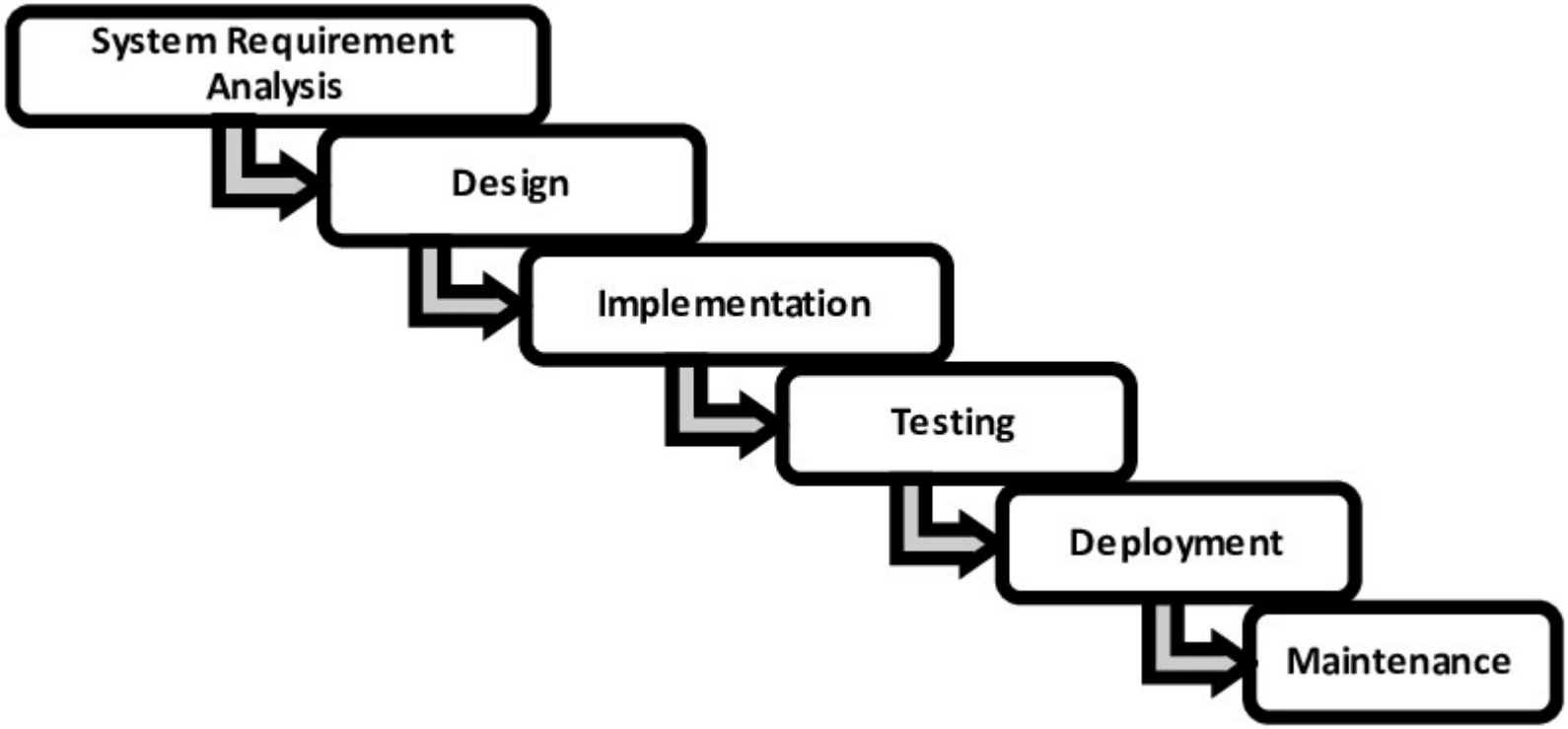
- Statt schlecht planbarem
  - Code & Fix



- hat Turing schon in den 50er Jahren die Notwendigkeit der Planung durch Zerlegung der Entwicklung in Teilschritte eingeführt
- Und dies hat Royce 1970 als Wasserfall-Modell in der Softwareentwicklung etabliert …

Sequentielles Wasserfall-Modell

- Phasen aus Winston W. Royce: Managing the Development of Large Software Systems, 1970



Lasten-/Pflichtenheft

- Basis der Entwicklung nach Wasserfall-Modell ist meist die Definition eines **Lastenhefts** durch den Auftraggeber (AG), evtl. auch mit Unterstützung des Auftragnehmers (AN) bei Erarbeitung.
- Mit dem **Pflichtenheft** spezifiziert der AN dann konkrete Aufgaben zur Umsetzung zu festgelegten **Meilenstein**-Terminen. Auf dieser Basis beauftragt der AG den AN.
- Die Abnahme erfolgt am Ende der Entwicklung, evtl. auch schrittweise zu jedem Meilenstein.

Wasserfall-Modell (mit *Artefakten*)

Planung
AG gibt Wünsche an, Erstellung eines ***Lastenhefts*** mit allem, was AG vom Produkt erwartet

Artefakte (oft als Dokumente) bei Übergabe zur nächsten Phase

Definition
Analyse und Definition der Anforderungen, Entwicklungsteam erstellt ***Pflichtenheft*** mit allem, was an Aufgaben zu definierten Meilensteinen zu erledigen ist, Umsetzung der Wünsche in konkreten Teilaufgaben

Sequentielle Abarbeitung

Entwurf
Modellierung des Problems und Erstellen eines ***Produktentwurfs*** (z.B. UML-Diagramme)

Rollen speziell zu Phasen, z.B. RE, Designer, Progr., Tester, etc.

Implementierung
Umsetzung des Entwurfs, ***Programmcode***

Test
Zusammenführen der Teile und ***Tests***

UND Projektleiter

Meilensteine sind entlang dieser Zeitlinie

Wartung und Pflege
Benutzerhandbuch, Benutzerschulung, Verbesserungen und Anpassungen

Meilensteinplan für Workshop

- 9h30 Projekteinführung (Gennaro, Marc)

- 10h30 Einrichtung/Beginn
- dazwischen mind. zwei Meilensteine
- 14h30 letzter Meilenstein, finales Release

- danach: Vorstellung der Arbeit (d.h. eigener Prozess) in gegenseitigen Team-Updates
- Wechselt regelm. die Hüte der einzelnen Rollen (d.h. auch PL) …

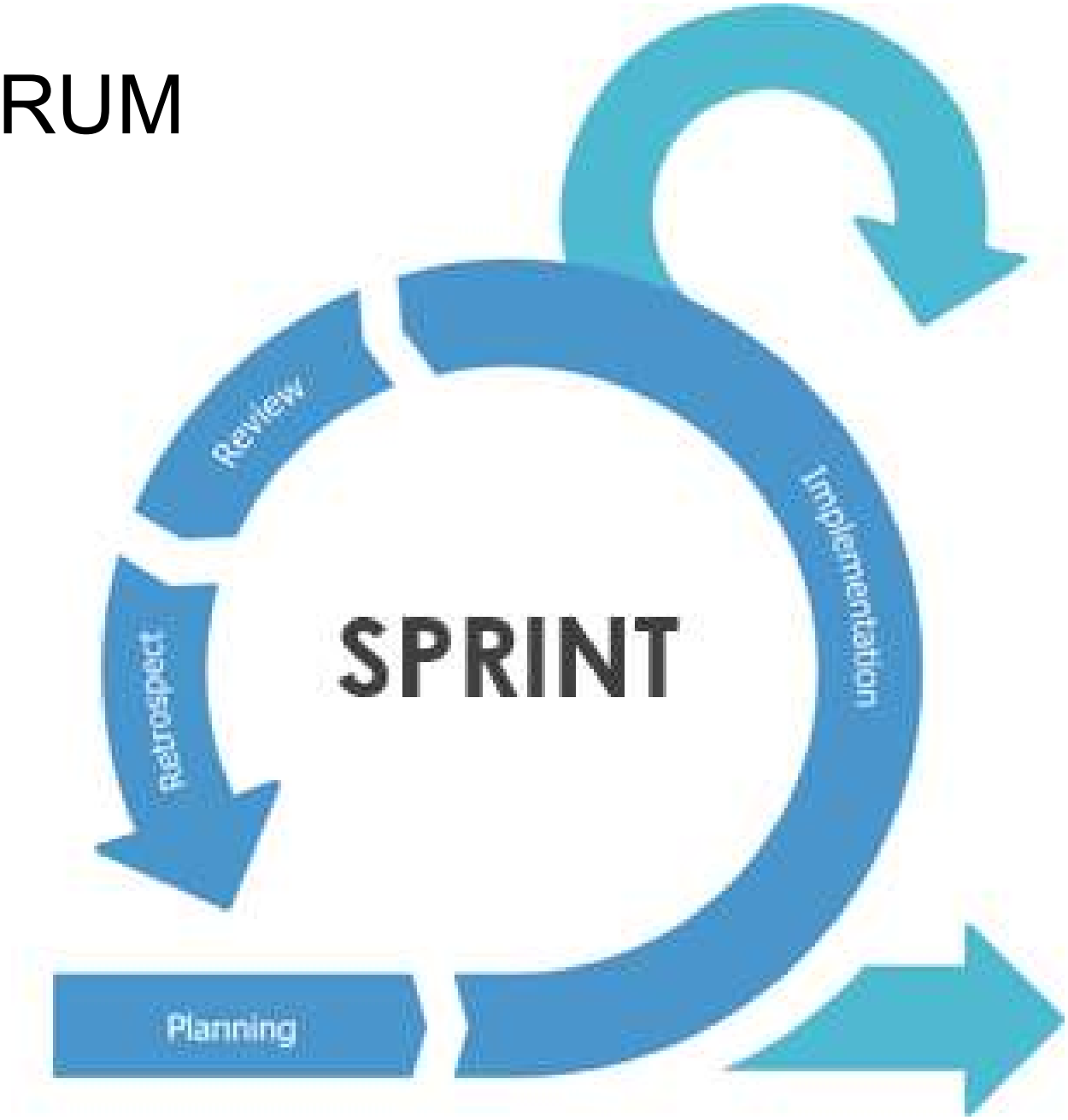
Abgaben im Workshop

- Git-Repo mit
  - Lasten-/Pflichtenheft (incl. Design)
  - Quellcode (incl. Tests)
  - Benutzerhandbuch
  - Zwischenstand bei Meilensteinen muss durch explizite Commits erkennbar sein

Agiles Manifest, 2001

- **Individuen und Interaktionen** sind wichtiger als Prozesse und Werkzeuge
- **Funktionierende Software** ist wichtiger als umfassende Dokumentation
- **Zusammenarbeit mit dem Kunden** ist wichtiger als Vertragsverhandlung
- **Reagieren auf Veränderung** ist wichtiger als das Befolgen eines Plans

SCRUM



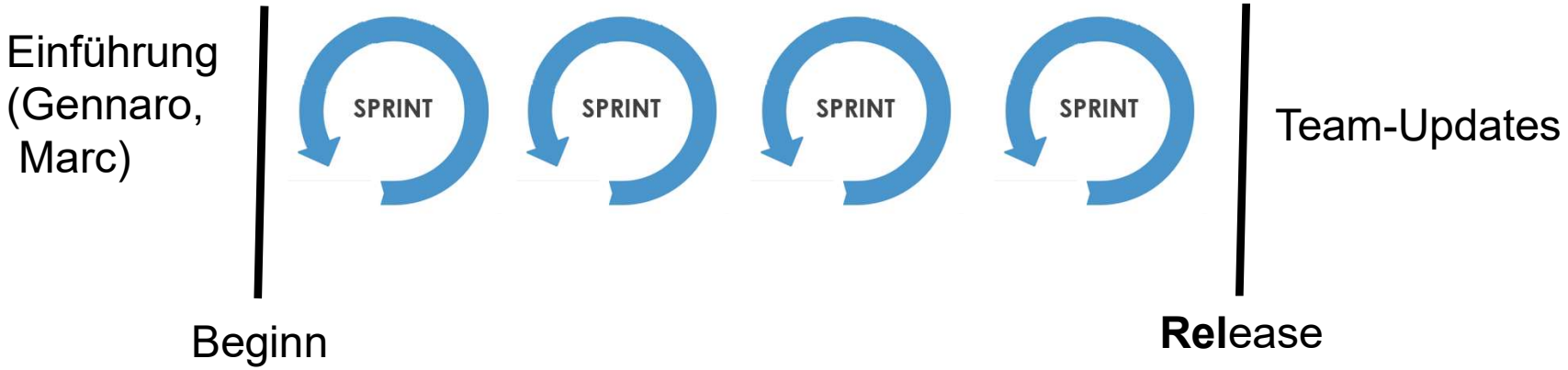
Grundlagen von SCRUM (1/2)

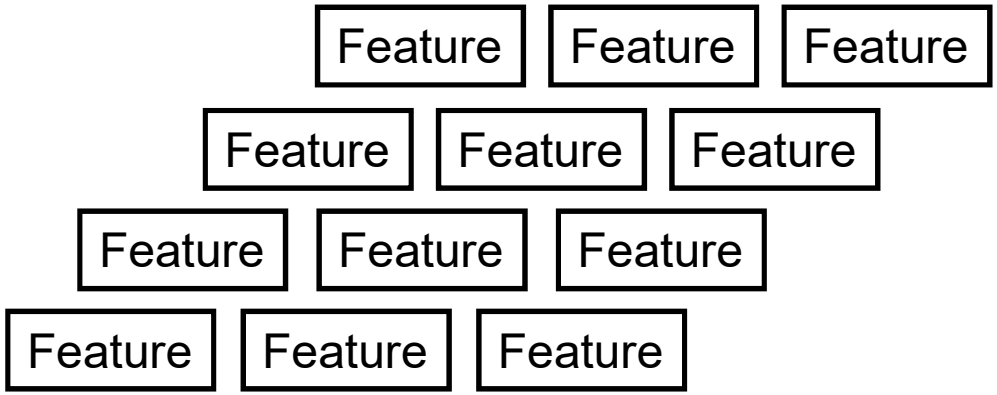
- disziplinübergreifendes Entwicklerteam, das stabil und zu 100% engagiert ist
- Entwicklerteam entscheidet, wie die Arbeit erledigt wird. Es ist selbstorganisierend
- Entwicklerteam plant seine Sprints zu Beginn jedes Sprints nacheinander
- Product Owner entscheidet, was produziert werden soll
- Entwicklerteam entscheidet, wie viel es in einem Sprint produzieren kann
- Ziel des Entwicklerteams in einem Sprint wird geteilt, ist klar und ändert sich nicht

Grundlagen von SCRUM (2/2)

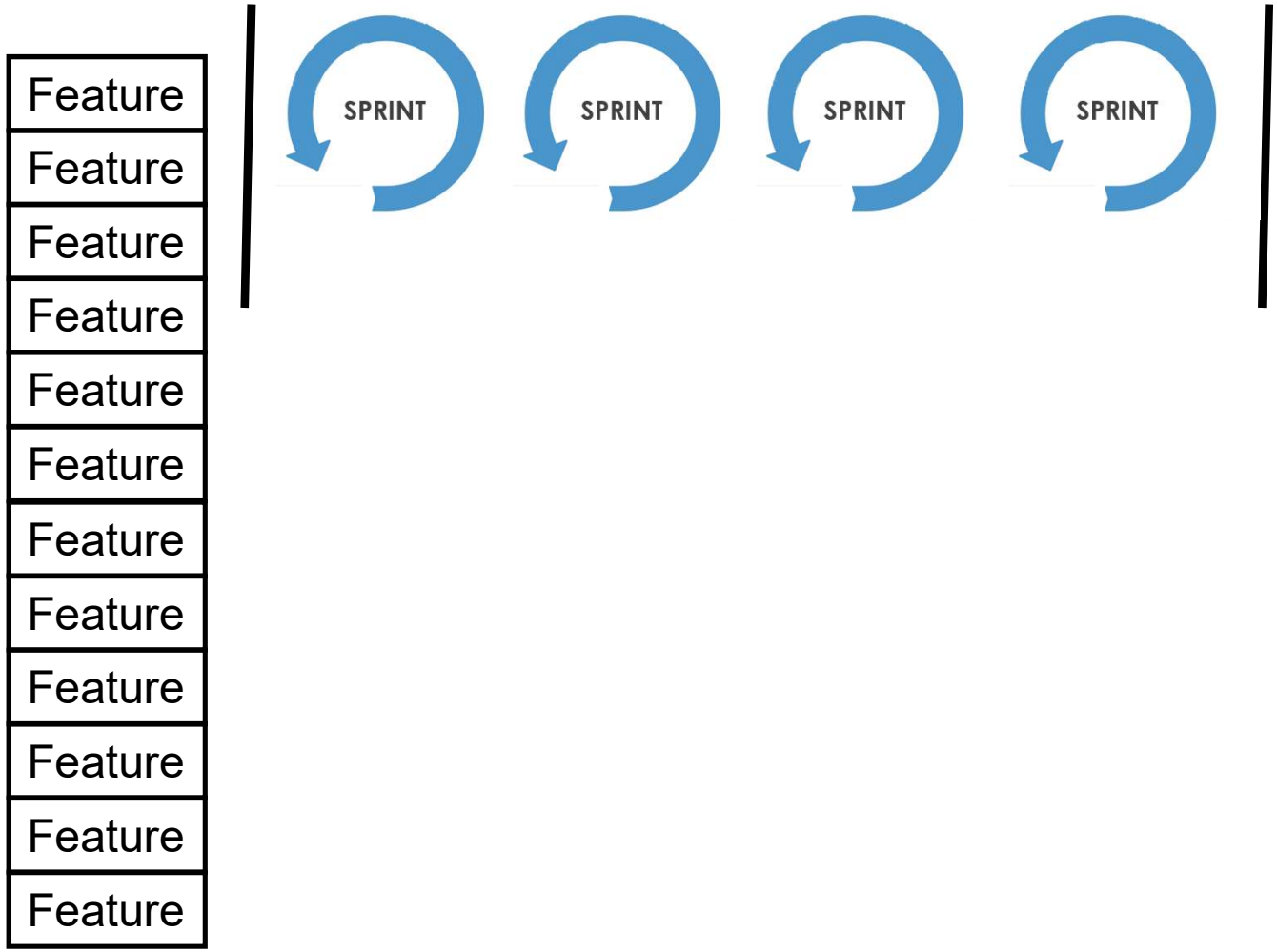
- Entwicklerteam versucht, sein Ziel zu erreichen, kann jedoch zu viel liefern oder
- Unterlieferung, basierend darauf, wie sich der Sprint entwickelt
- Jeder Sprint ist eine Zeitbox - seine Länge wird niemals verlängert
- Jeder Sprint produziert ein „potenziell versandfähiges Produkt“ in anderen Worten:
  gründlich getestet, fehlerfrei und „erledigt“
- Es gibt eine "Definition of Done", die angibt, was „erledigt" bedeutet
- Am Ende jedes Sprints prüfen und passen wir Produkt und Prozess an

Workshop-Plan mit SCRUM

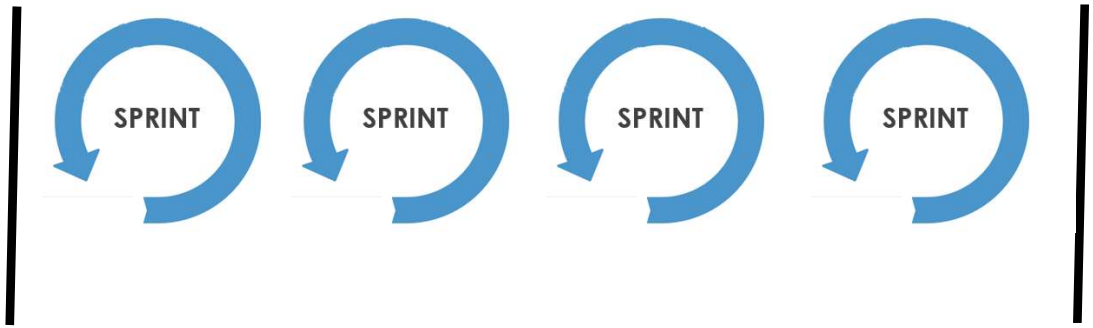


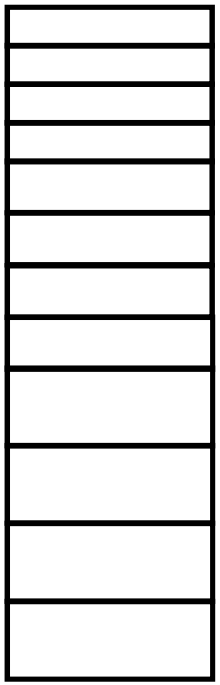


SCRUM

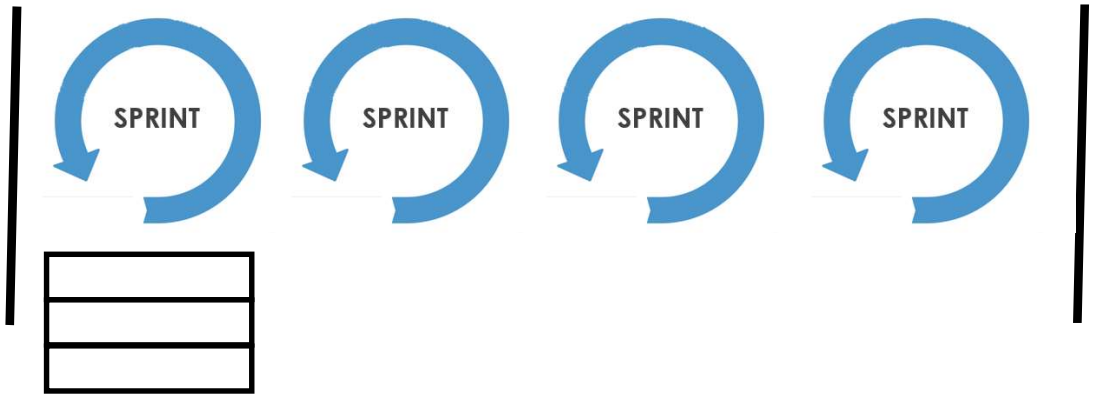

Feature
Feature
Feature
Feature
Feature
Feature
Feature
Feature
Feature
Feature
Feature
Feature
SPRINT
SPRINT
SPRINT
SPRINT

SCRUM

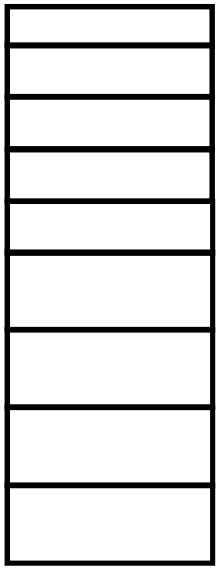




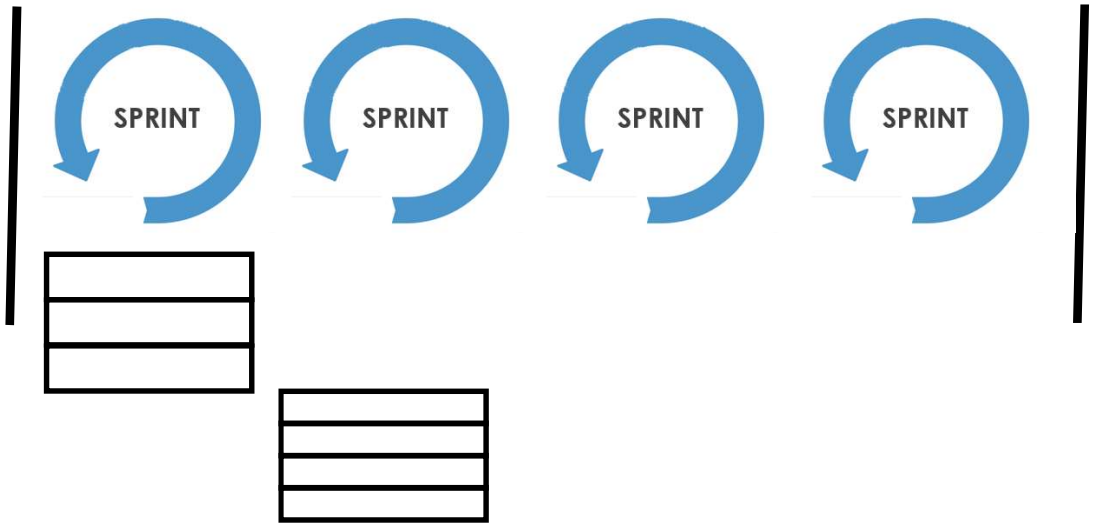
SCRUM



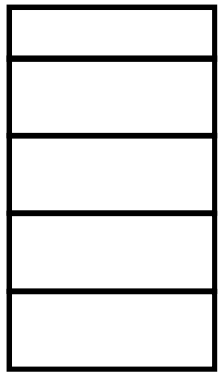
„erledigt“ – umgesetzt, getestet,
keine Fehler übrig geblieben



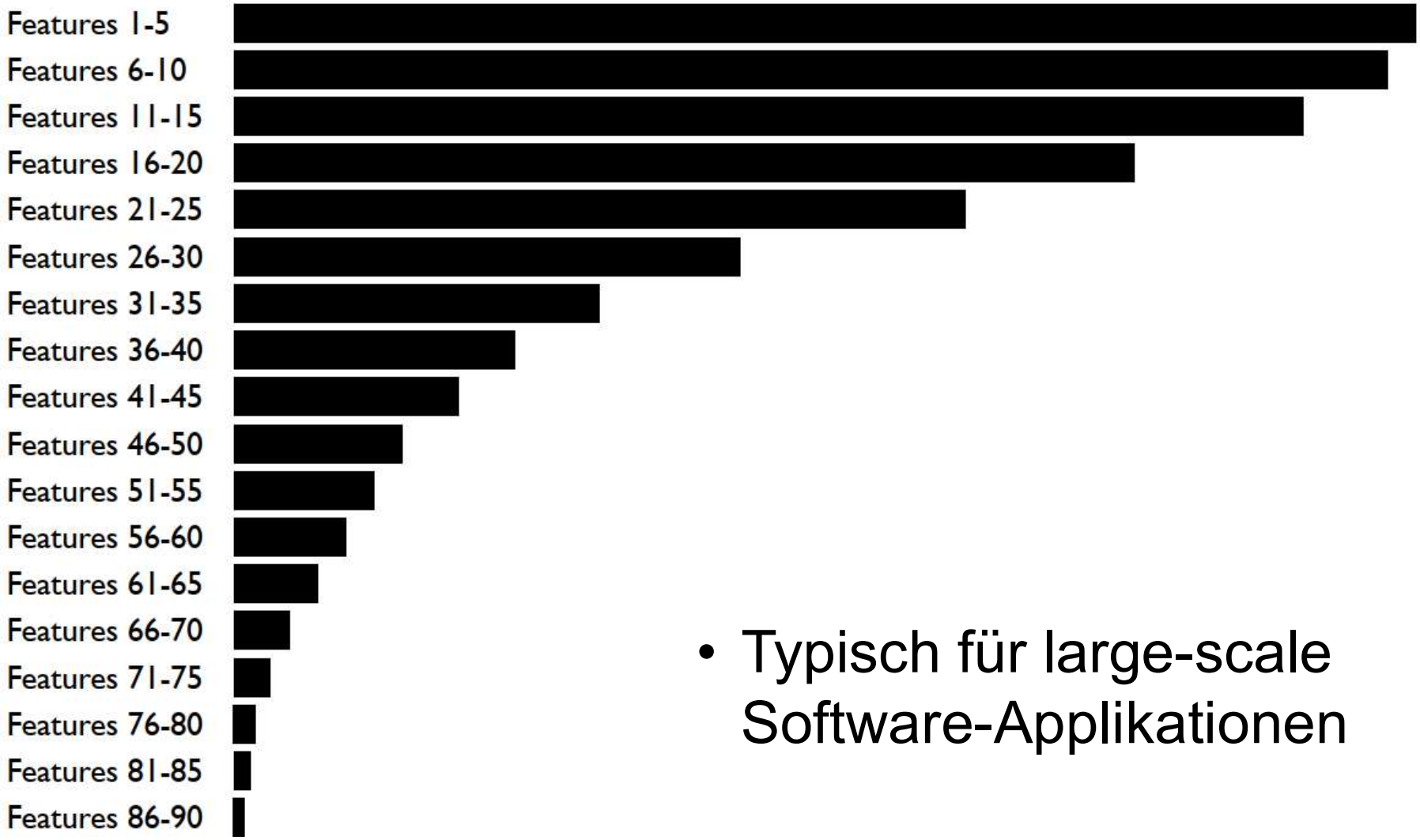
SCRUM



„erledigt“ – umgesetzt, getestet,
keine Fehler übrig geblieben



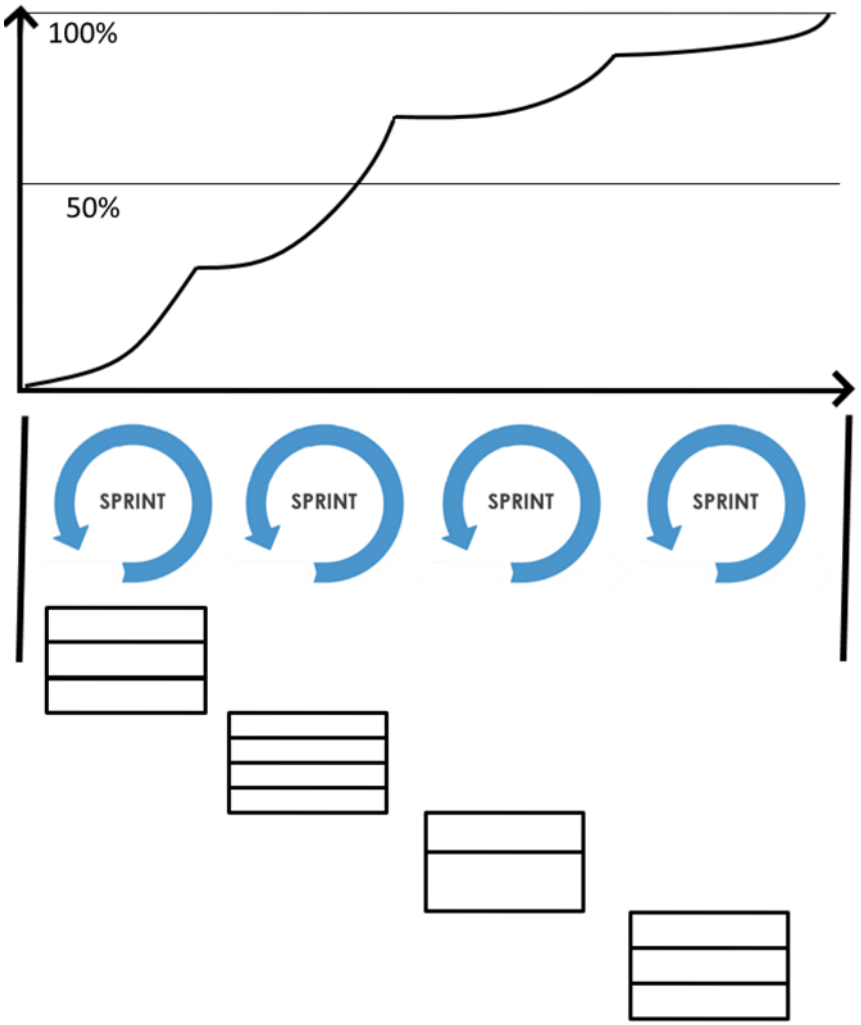
Relativer „Wert“ einzelner Feature



- Typisch für large-scale Software-Applikationen

Beispiel für Workshop

- % Umsetzung werthaltiger Feature



SCRUM

SCRUM Team

SCRUM

Verantwortlich für die Arbeit
- 7 +/-2 Personen
- interdisziplinäre Besetzung
- selbstorganisierend

Verantwortlich für die Maximierung des Wertes der Arbeit des Entwicklerteams
- Vision und Ziele der Arbeit
- endgültiger Entscheidungsträger für Inhalt und Reihenfolge des Product Backlog (Liste der Aufgaben)
- Datum und Umfang der Veröffentlichung

Verantwortlich für Führung des SCRUM-Teams um opt. Ergebnisse zu erzielen

Prozessinhaber
- Lehrt Scrum an alle
- Coacht wie man Scrum benutzt um max. Wert u. Qualität zu liefern
- Erzwingt die Regeln von SCRUM

Problemlöser
- Hilft beim Entfernen von Blockaden und unterstützt SCRUM-Team damit dies die besten Ergebnisse erzielen

Protector
- Schützt Entwicklerteam vor Störungen

SCRUM im Workshop

„Entwickler“, d.h. Architekten, Programmierer, Tester, UI Designer, Redakteure, …

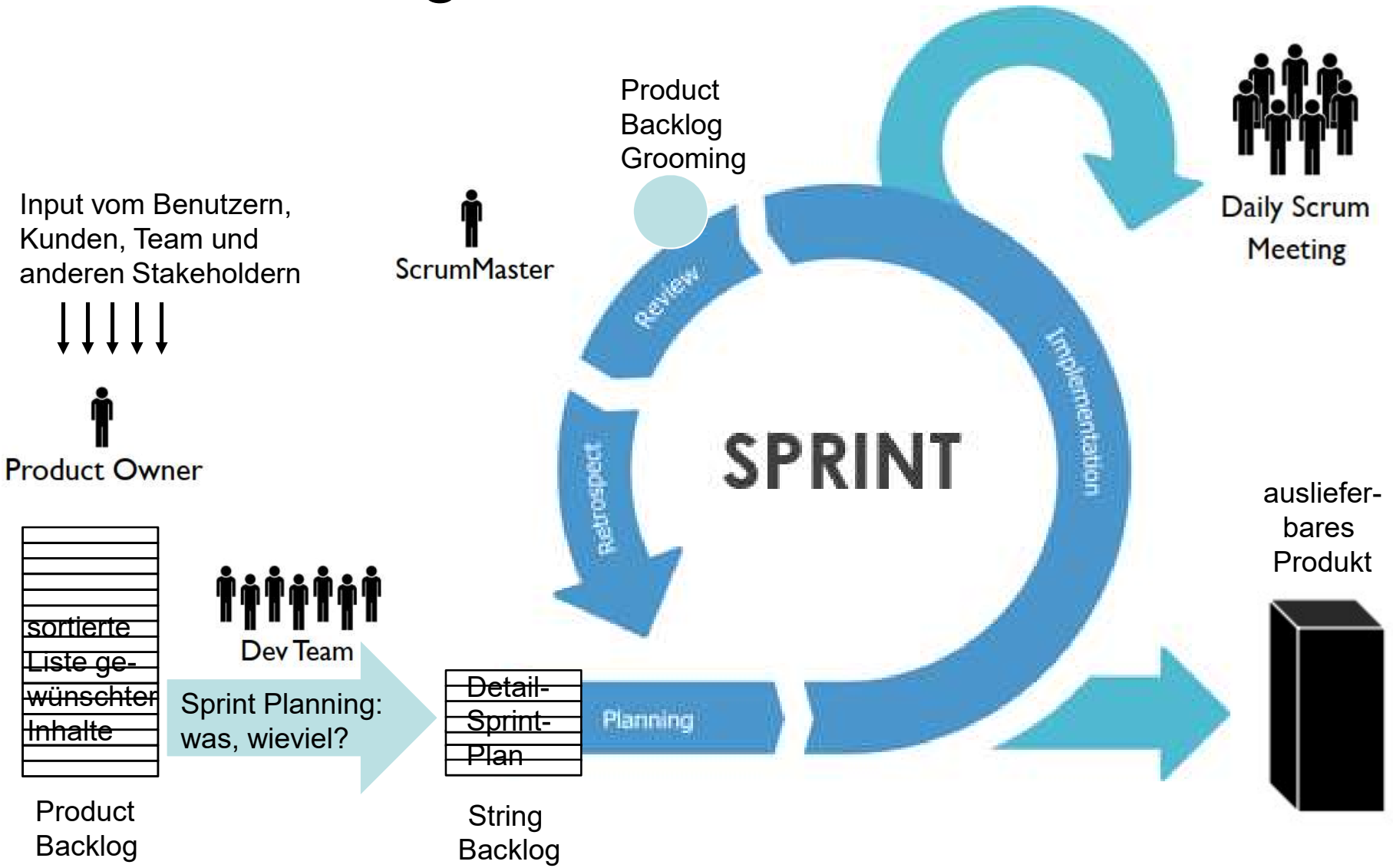
Rolle könnte vom Kunden übernommen werden, Kundenvertreter, Produkt-Manager, Programm- oder Projektmanager.
NUR hier für die Erfahrung: Übernahme der Rolle durch ein Mitglied des Dev Teams

Setzt dabei explizit einen anderen Hut auf!

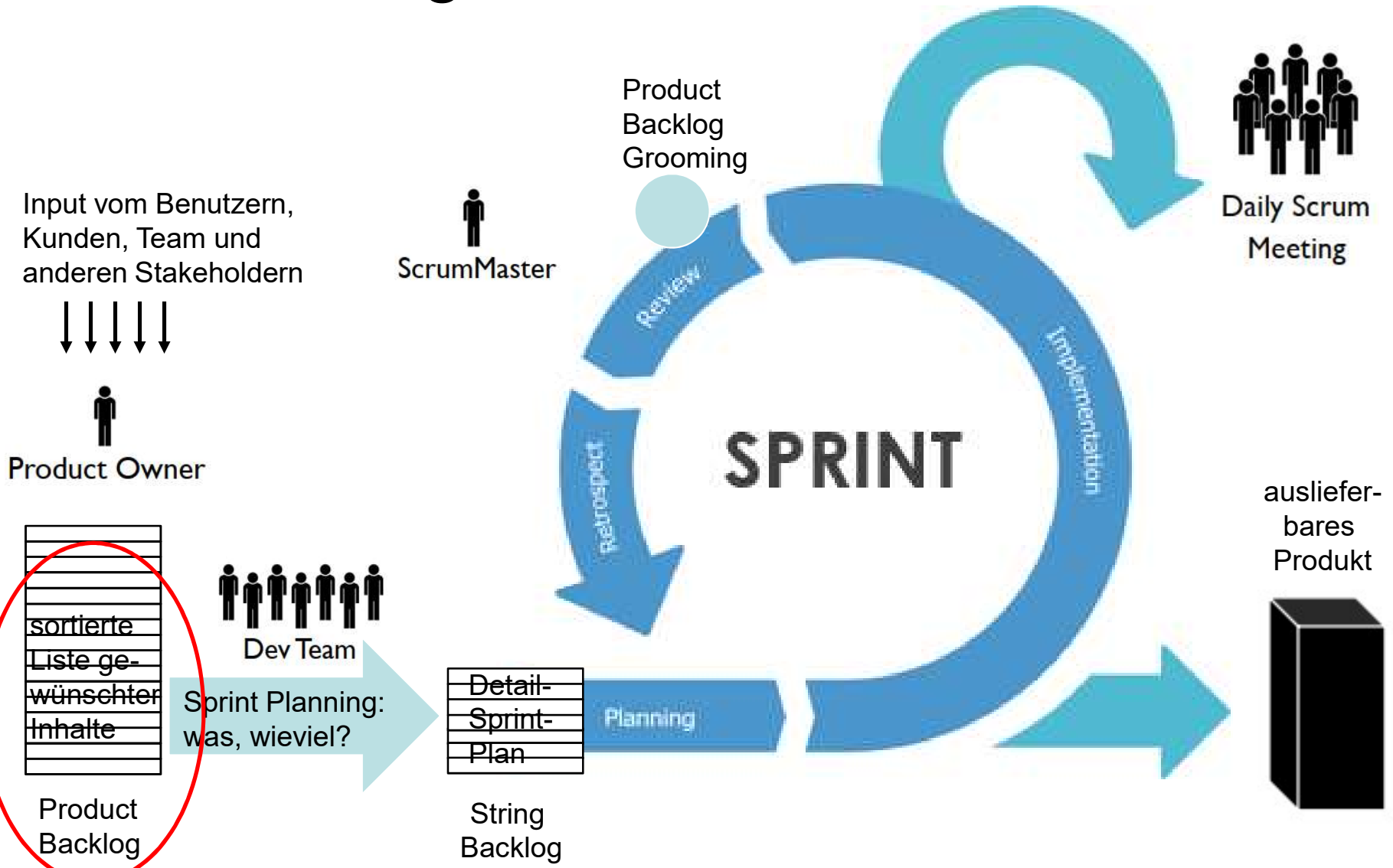
Rolle kann von einem Mitglied des Entwickler Teams übernommen werden, oder Ex-Projektleiter.

Diese Person sollte besser keiner sein, der ein „Management“ im Titel trägt.

Entwicklung in SCRUM



Entwicklung in SCRUM



Requirements für SCRUM

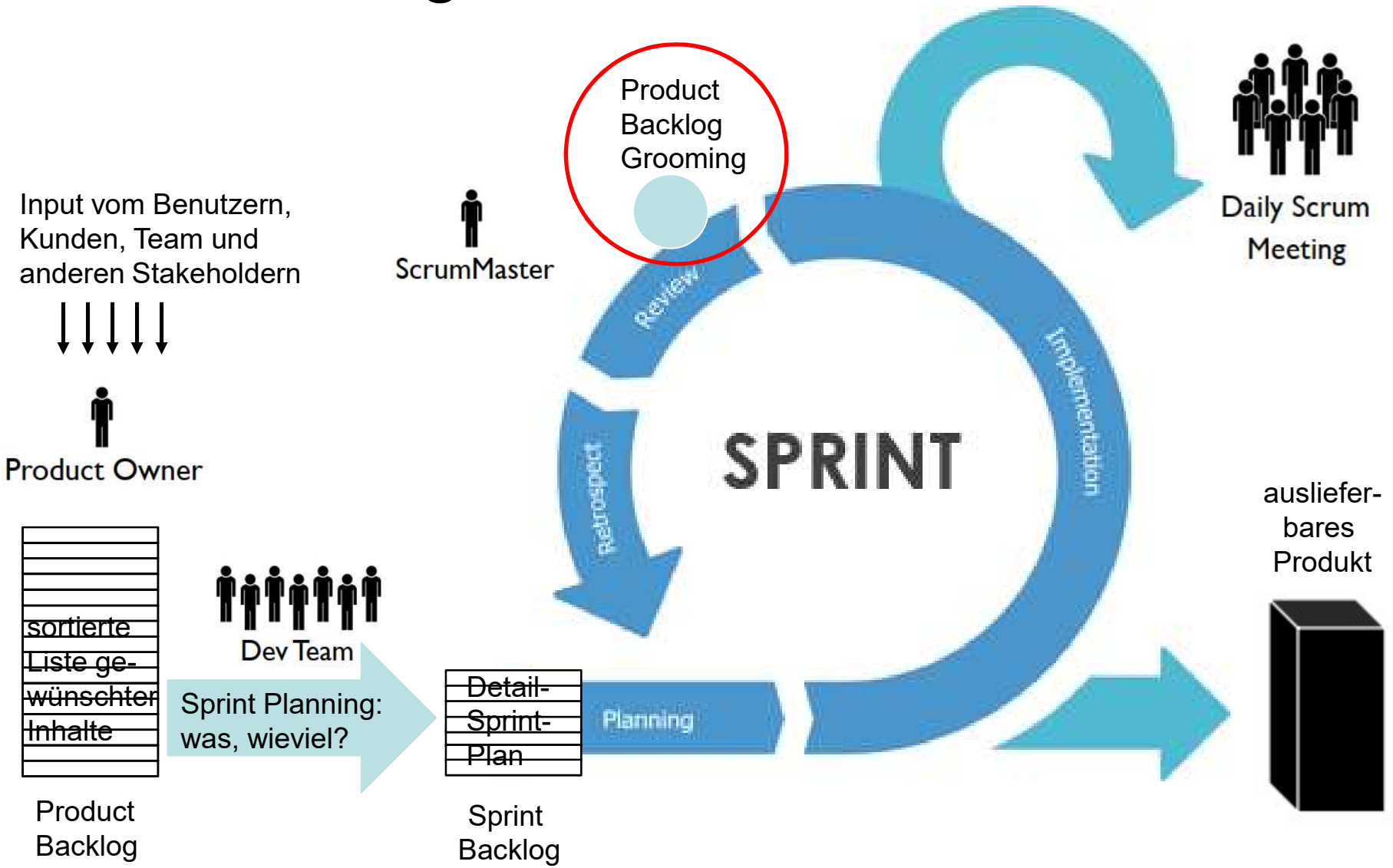
- User Stories sind ein agiler Ansatz für RE
  - Kurze textuelle Beschreibung der vom Benutzer gewünschten Inhalte; konzentriert auf WAS gebraucht wird und WARUM.
  - Jeder schreibt User Stories, der Product Owner entscheidet welche ins Product Backlog aufgenommen werden und priorisiert/sortiert diese.
  - Große Stories sind „Epics“ und müssen in einzelne User Stories heruntergebrochen werden.
  - Aufbau einer User Story:
    Als [Nutzer / Rolle] möchte ich [WAS?], dass / weil / um [WARUM? / Nutzen / Wert].

User Stories vs. Backlog Einträge

- Einträge im Backlog sollten als konkrete Aufgaben umsetzbar sein, wenn eine User Story hierfür zu groß ist muss diese heruntergebrochen werden
  - Größe = Aufwand + Komplexität + Unsicherheit
  - Abschätzung in Story Points z.B. durch Planning Poker
- Identifikation von (Teil-)Aufgaben mit <= 1-2 Tage Aufwand

- Für den Workshop mit Sprint in 1h und 4er Team müssen die Werte entsprechend skaliert werde!

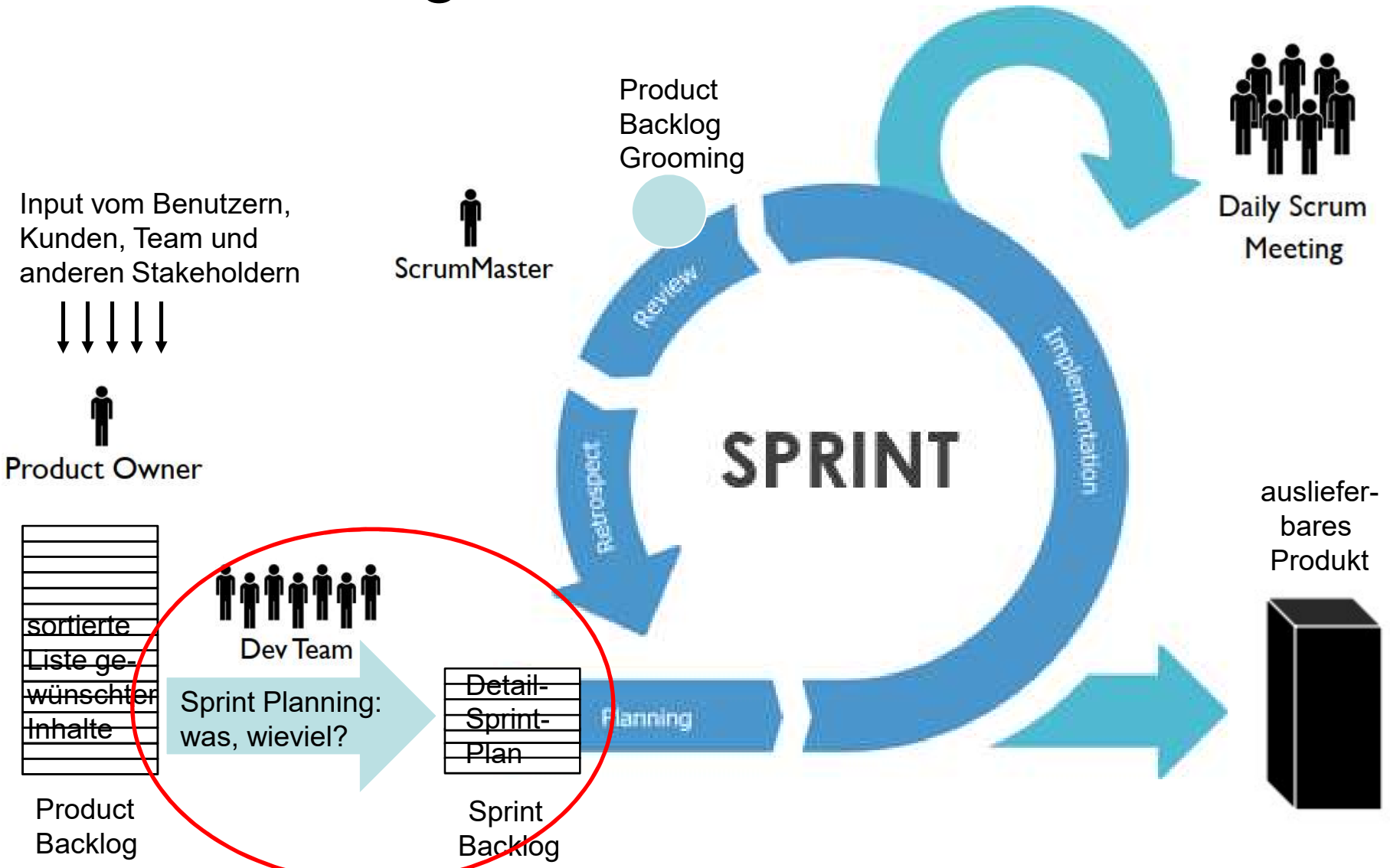
Entwicklung in SCRUM



Product Backlog Grooming

- In jeden Sprint prüfen Entwickler Team und Product Owner die im Product Backlog für die nächsten 2-3 Sprints anstehenden Einträge
- Große Backlog Einträge werden in kleinere Teile aufgeteilt, klein genug für die Bearbeitung durch 1-2 Personen in einem Sprint
- Entwickler Team bekommt besseres Verständnis der anstehenden Punkte
- Aufwand 5-10% der Arbeitszeit für einen Sprint
- Organisiert durch ScrumMaster

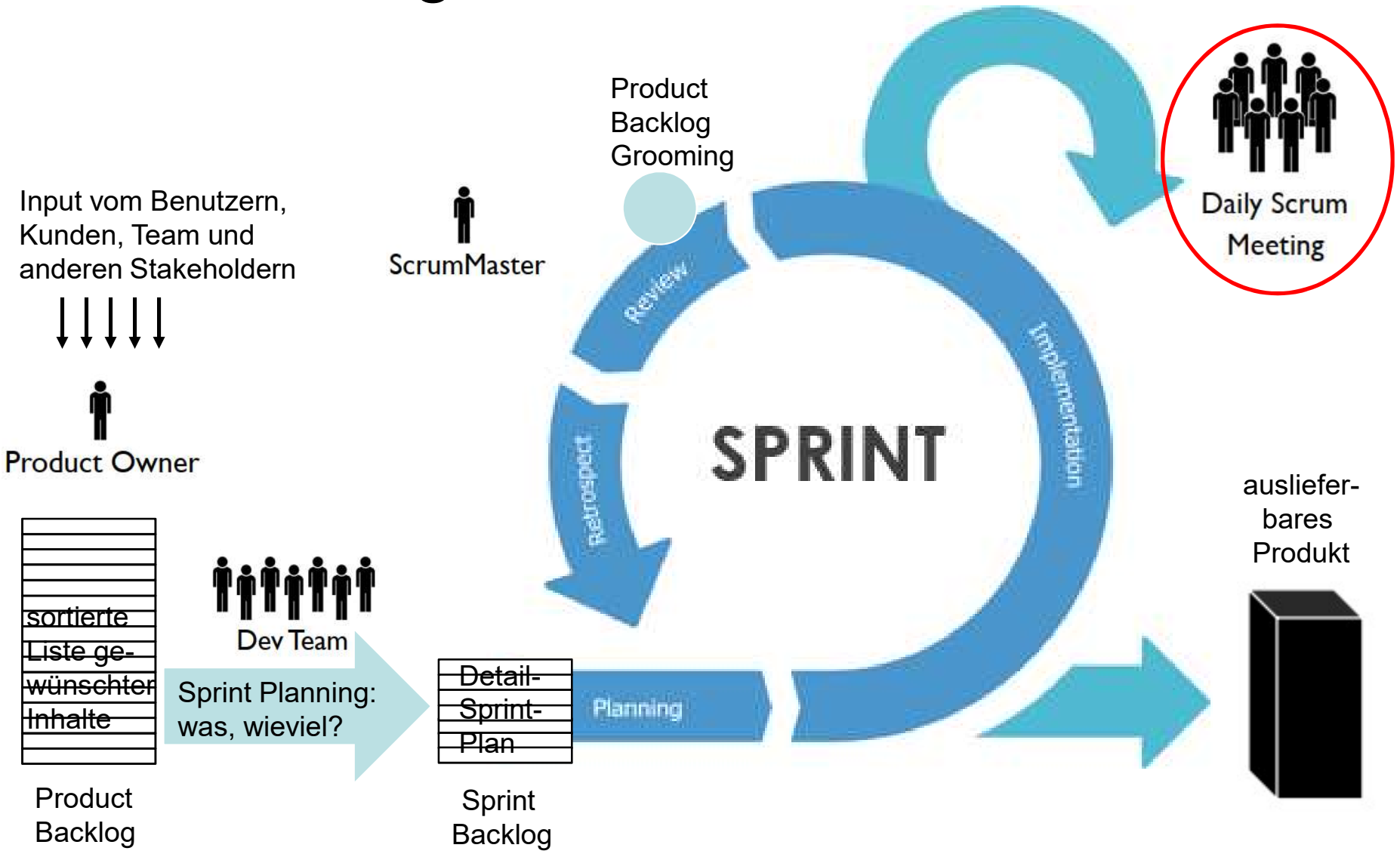
Entwicklung in SCRUM



Sprint Planning Meeting

- Gemeinsame Leitung durch Entwickler Team, Product Owner und ScrumMaster
- Oft aufgeteilt in:
  - Teil 1 – Entwickler Team entscheidet wie viel im nächsten Sprint umgesetzt werden soll
  - Teil 2 – Entwickler Team erstellt Plan wie das Ziel erreicht wird
- Mögliche Ansätze wie die Entscheidung wieviel umgesetzt werden soll ermittelt wird:
  - Geschwindigkeitsbasiert: aus historischen Daten
  - Commitment-basiert: Team schätzt seine Kapazität und commitet sich zu Aufgaben im nächsten Sprint

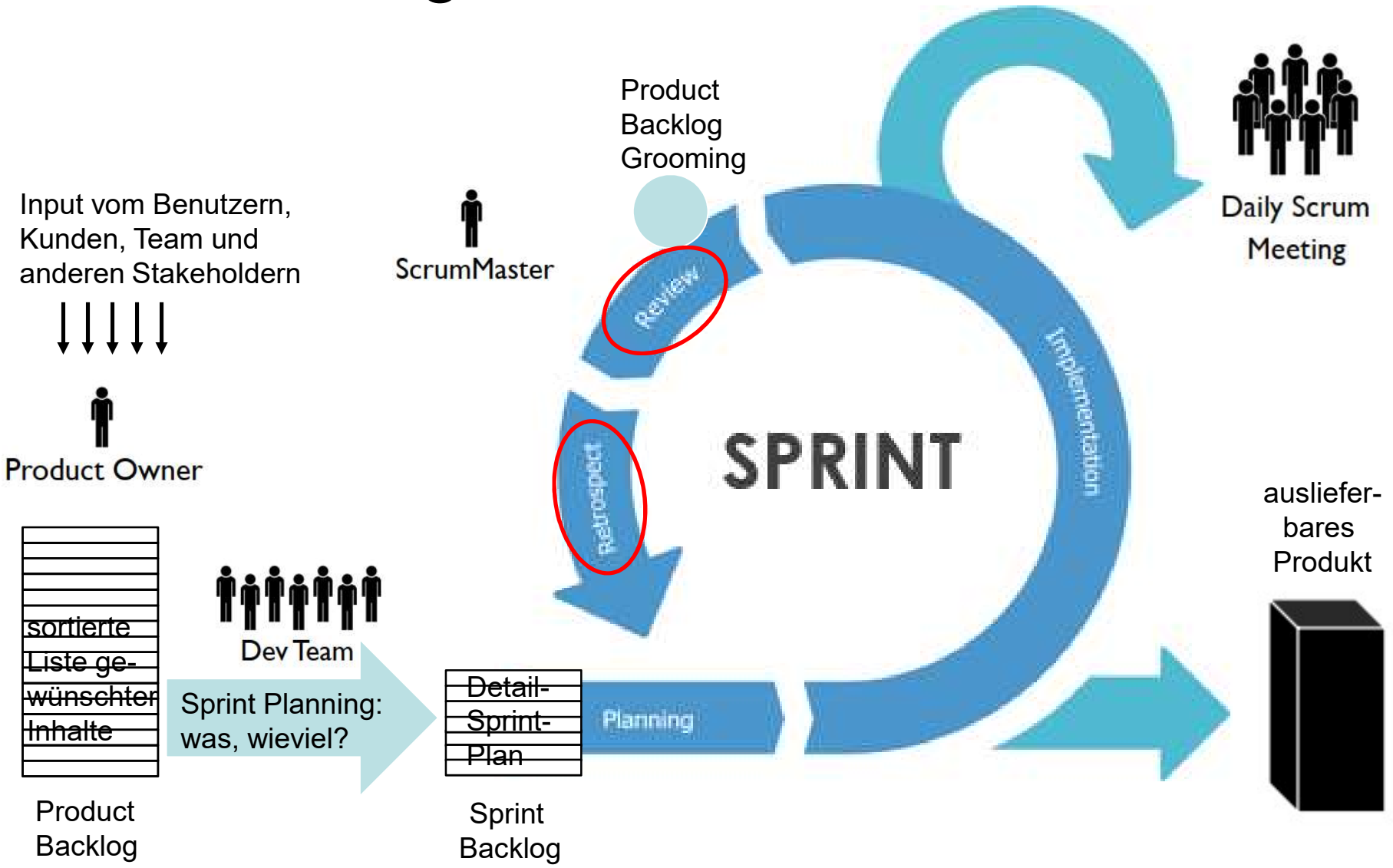
Entwicklung in SCRUM



Daily SCRUM Meeting

- tägliche (d.h. stündl.) Ziele
  - Gegenseitiges Update im Entwickler Team
  - Blockierungen für SM und Team sichtbar machen
- Bericht zu drei Inhalten
  - Was habe ich seit dem letzten Daily getan?
  - Was ist der Plan bis zum nächsten Daily?
  - Blockiert mich etwas?
- max. 15 Minuten (im Workshop nach Bedarf)
- während Meeting, nur zuhören, keine Diskussion
- nach Meeting: Diskussion bei Bedarf; SM hilft bei Blockaden
- PO kann muss aber nicht teilnehmen

Entwicklung in SCRUM



Sprint Review + Retrospective

- Review: Produkt prüfen und anpassen
  - PO, DEV, SM, Stakeholder (im Workshop evtl. AG)
  - Qualitätsprüfung neuer Inhalte in „hands on“-Workshop
  - entspricht dies den Kundenbedürfnissen?
  - Verbesserungsvorschläge für zukünftige Entwicklung
- Retro: Prozess prüfen und anpassen
  - DEV, SM
  - Was haben wir während des Sprints erlebt?
  - Aktionsplan zur Verbesserung im nächsten Sprint
  - Dies lohnt in jedem Sprint!

Definition of Done

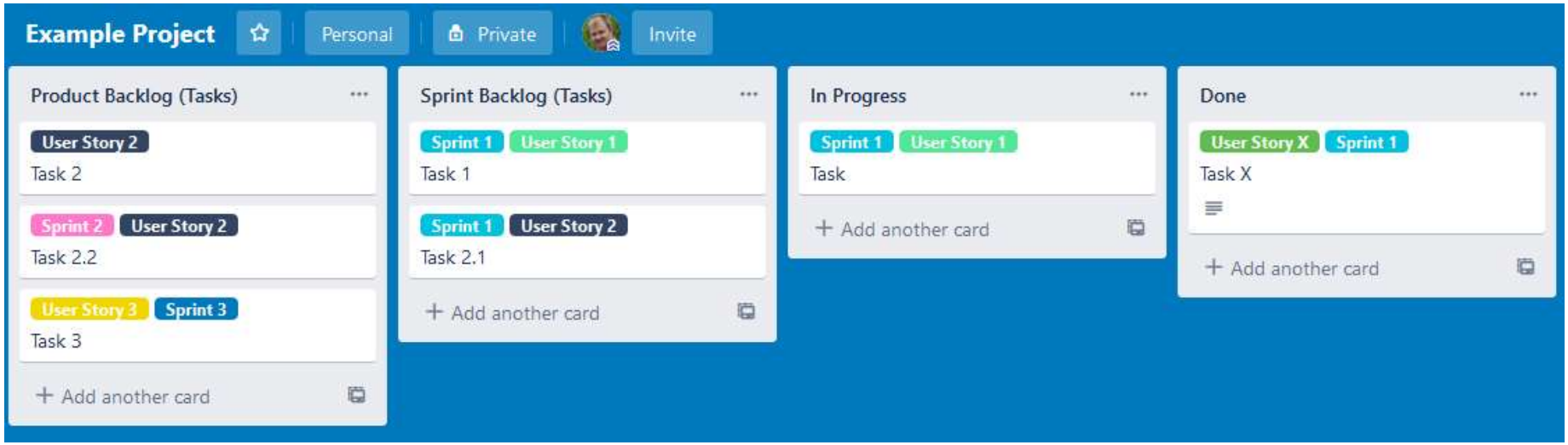
- D.o.D. definiert was es für einen Backlog Eintrag heißt, wenn dieser „erledigt“ ist.
  Mögl. Kriterien sind:
  - Code complete
  - Code reviewed
  - Unit tested
  - Integration tested
  - Acceptance tested
  - System- bzw. Benutzerdokumentation fertig
  - keine bekannte Fehler
- PO und DEV definieren D.o.D. vor dem 1. Sprint
- Es gibt evtl. Backlog Einträge außerhalb D.o.D.

Abgaben im Workshop

- Git-Repo mit
  - Backlog und Sprint-Planning über Trello.com als PDF
  - Dokument mit User Stories (Verweis aus Trello.com)
  - Quellcode (incl. Tests)
  - Benutzerhandbuch
- Einladung zum Review
  - Die AG können rein zeitlich nicht an allen Reviews teilnehmen, aber als Team solltet Ihr regelmäßig ein Review einplanen und so die Verantwortung an den PO übergeben, gerne nimmt einmal auch der AG teil.

Tools

- Trello.com bietet ein einfaches KANBAN-Board, das auch für die Verwaltung von SCRUM Backlogs geeignet ist, verwendet Labels zur Markierung von Sprints und User Stories:



- Eine Druckversion als PDF nach jedem Sprint-Planning ist Teil der Abgabe.

Teams

A) alqataa.marwa@dh-karlsruhe.de + leandro.moos@outlook.de + wittenberger.max@dh-karlsruhe.de
semling.daniel@dh-karlsruhe.de + bielefeld.moritz@dh-karlsruhe.de

B) wegmann.leander@dh-karlsruhe.de + muto.michele@dh-karlsruhe.de + lichter.ansgar@dh-karlsruhe.de
rothmann.simon@dh-karlsruhe.de + ibach.viktoria@dh.karlsruhe.de

C) selensky.charlotte@dh-karlsruhe.de + balim.beliz@dh-karlsruhe.de + scheuring.markus@dh-karlsruhe.de
nauert.domenic@dh-karlsruhe.de + topkac.gamze-nur@dh-karlsruhe.de

D) sollner.paula@dh-karlsruhe.de + swidersky.viktoria-s@dh-karlsruhe.de + pflittner.sascha@dh-karlsruhe.de
voelkel.ann-cathrin@dh-karlsruhe.de + heidsiek.alexander@dh-karlsruhe.de

E) boerkel.isabel@dh-karlsruhe.de + fichtner.patrick@dh-karlsruhe.de + ballauer.thomas@dh-karlsruhe.de
choudhery.aisha@dh-karlsruhe.de + dietz.sedric@dh-karlsruhe.de

F) illner.johanna@dh-karlsruhe.de + hollmann.justin@dh-karlsruhe.de + zelinka.sven@dh-karlsruhe.de
becker.julia@dh-karlsruhe.de + wolpert.julian@dh-karlsruhe.de

G) schneider.tim2@dh-karlsruhe.de + goeller.andre@dh-karlsruhe.de + kaspar-mueller.mathi@dh-karlsruhe.de
sasse.nico@dh-karlsruhe.de + bichlmaier.monika@dh-karlsruhe.de

H) frietsch.tobias@dh-karlsruhe.de + schmid.kai@dh-karlsruhe.de
wade.patrick@dh-karlsruhe.de + brandl.vanessa@dh-karlsruhe.de + geyer.florian@dh-karlsruhe.de

I) mitschke.jonas@dh-karlsruhe.de + hupe.fabian@dh-karlsruhe.de
beda.armin@dh-karlsruhe.de + akcay.melise@dh-karlsruhe.de