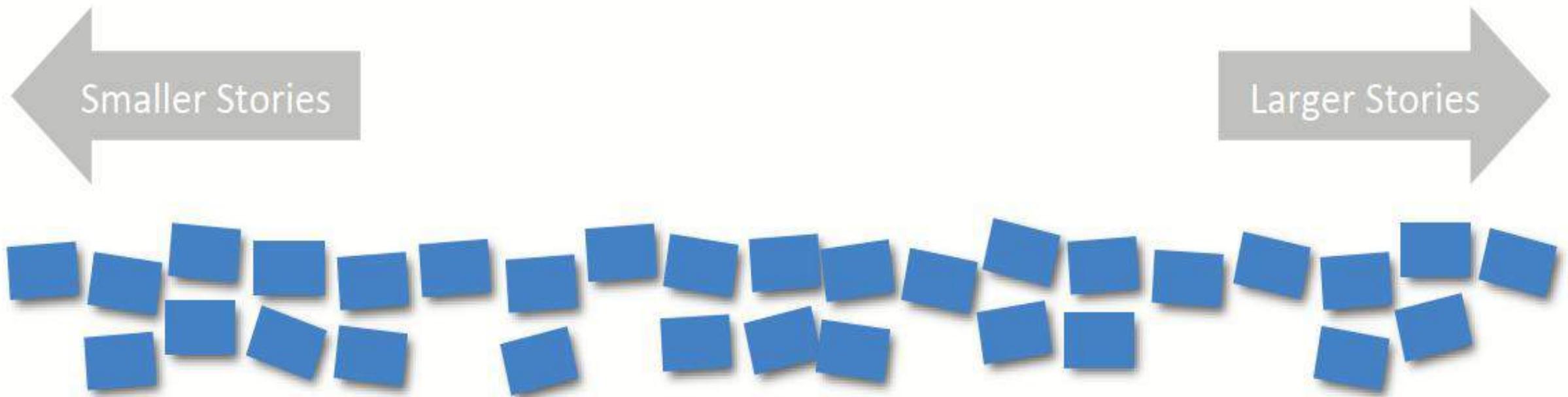


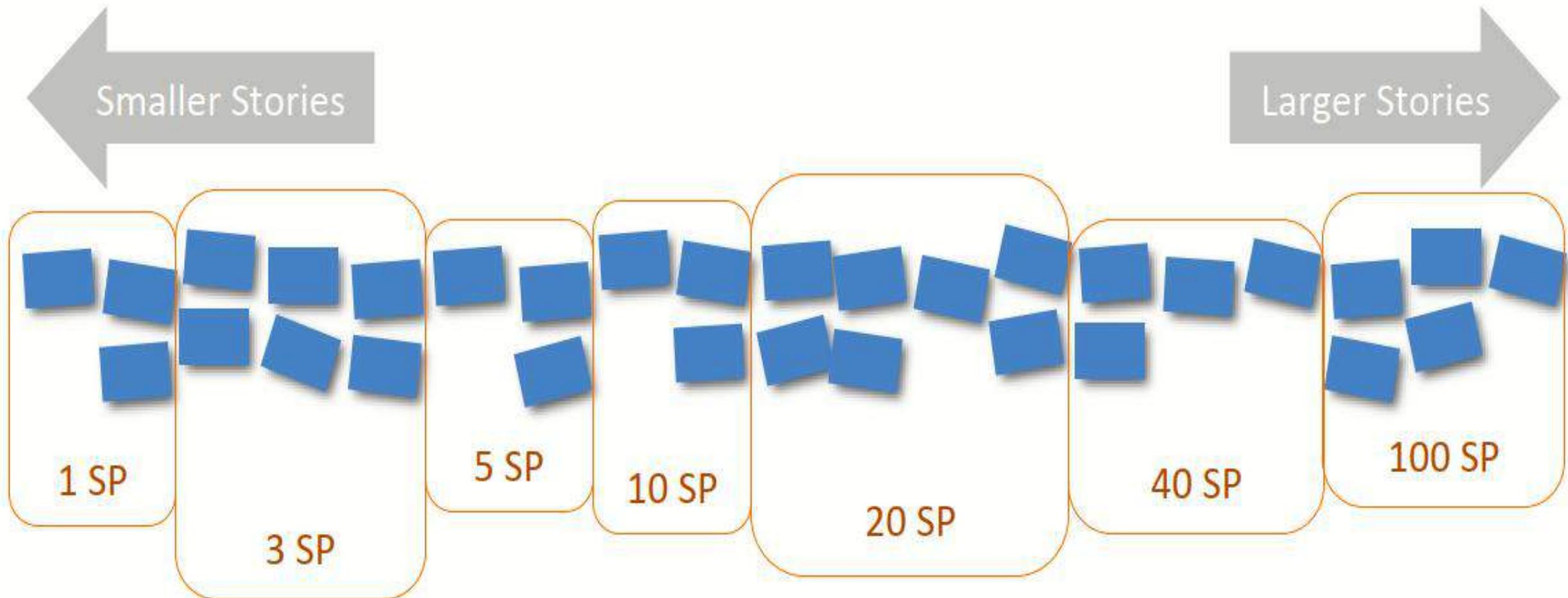
# AUFWAND SCHÄTZEN: AFFINITY ESTIMATION

## SCHRITT 1: STORIES NACH GRÖÙE SORTIEREN



# AUFWAND SCHÄTZEN: AFFINITY ESTIMATION

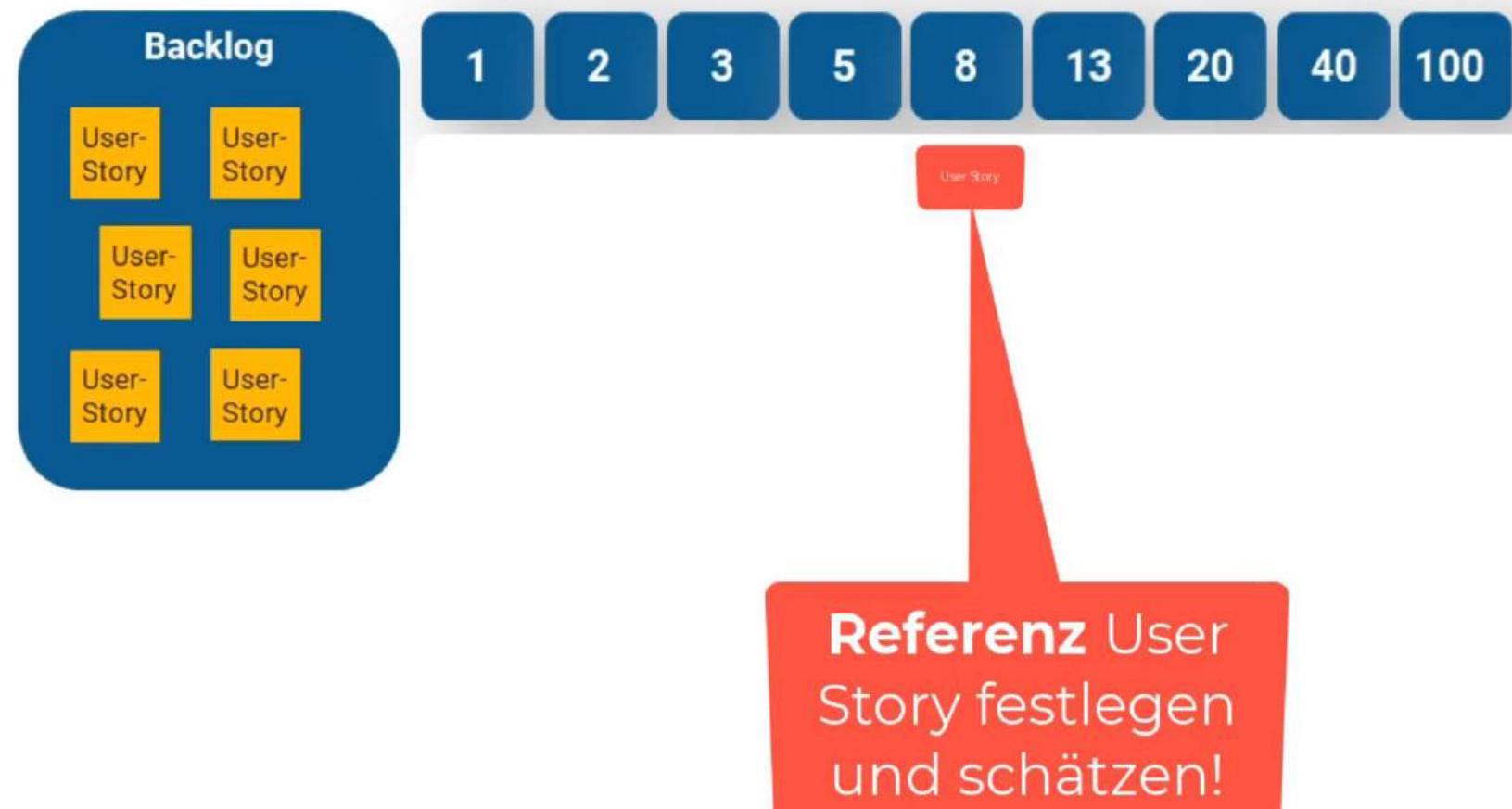
## SCHRITT 2: STORIES NACH IN GRUPPEN EINTEILEN



# SCHÄTZVERFAHREN SWIMLANE PLANNING

## Schätzverfahren Swimlane Planning

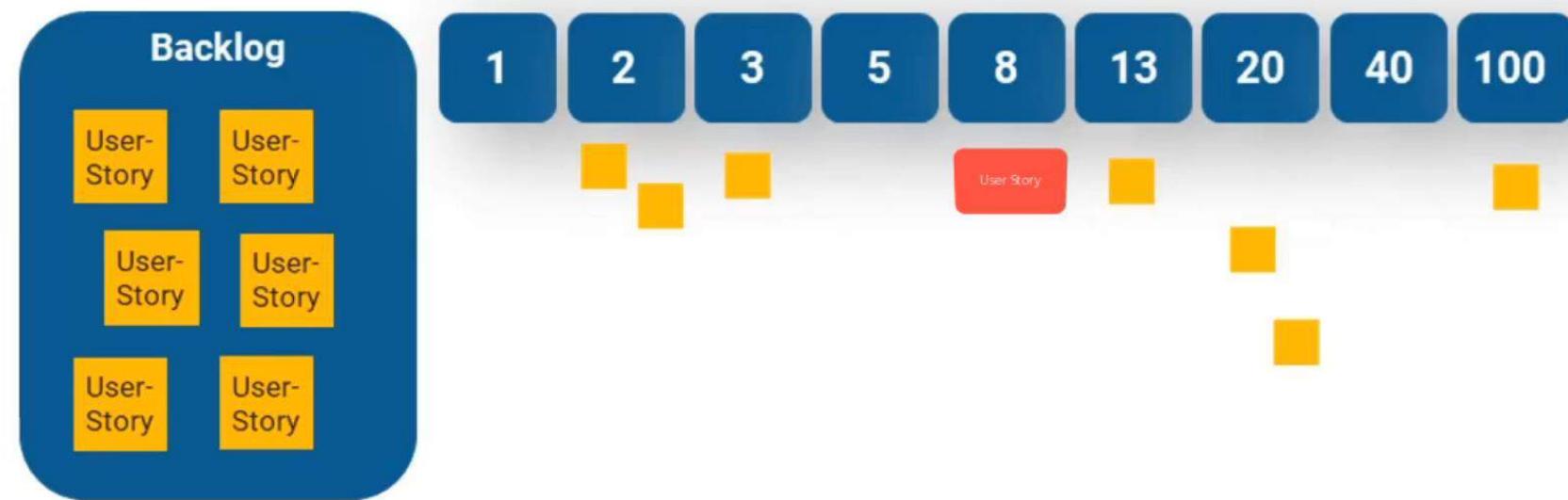
Das Swimlane Planning eignet sich für eine schnelle Einschätzung vieler User Stories



# SCHÄTZVERFAHREN SWIMLANE PLANNING

# Schätzverfahren Swimlane Planning

Das Swimlane Planning eignet sich für eine schnelle Einschätzung vieler User Stories



# SCHÄTZVERFAHREN ZUSAMMENFASSUNG

## Grundlagen

- Nur das **Entwicklungsteam** schätzt!
- Schätzungen erfolgen mit relativen Werten
- Schätzen erfolgen mit **Story Points**
- Story Points **messen nicht allein die Umsetzungsdauer!**  
Folgende Aspekte können / sollten mit einfließen:
  - Komplexität,
  - die Menge an Aufgaben (tasks)
  - mögliche Risiken und Unsicherheiten

## Vorteil relativer Schätzungen

- Relative Schätzungen **bleiben aussagekräftig**, auch wenn das Team in der Umsetzung „**schneller**“ wird
- Relative Schätzungen erlauben eine **direkte Vergleichbarkeit** zwischen User Stories (eine User Story mit 5 Story Points ist ungefähr doppelt so aufwändig, wie eine User Story mit 3 Story Points)

## Techniken

### Swimlane Planning

- **Schnell** eine Vielzahl von User Stories schätzen (**grob**)
- Schätzungen finden **parallel** statt
- Gut für den Projektstart

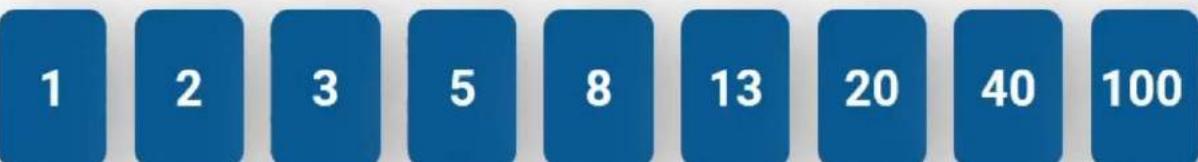
### Planning Poker

- **Wenige** Anforderungen **genauer** schätzen
- **Detaillierte Auseinandersetzung** mit **jeder einzelnen** Anforderung
- **Kartensets** sind **online** erhältlich, oder als **App**

## Fibonacci - Folge

Relative Schätzungen brauchen einen eindeutigen Referenzwert!

Die zunehmenden Abstände zwischen den Werten repräsentieren die **steigende Unsicherheit** in Relation zur Größe



## T-Shirt Schätzung



Als Alternative zur Fibonacci Folge wird auch gerne auf das „T-Shirt Sizing“ zurückgegriffen. Das Prinzip bleibt auch hier gleich und soll die Schätzung vereinfachen. Ein T-Shirt der Größe „M“ entspricht hierbei ca. 2x der Größe „S“, welches wiederum ca. 2x der Größe „XS“ entspricht

## Story Points = Stunden?

**Nein!** Story Points sind relative Werte, die zudem im Projektverlauf konstant bleiben und nicht verändert werden.  
**Vorteil:** Zunehmende Geschwindigkeit des Entw.teams kann dargestellt werden

Sprint 1	
Team Kapazität	10 Stunden
User Story A	5 Story Points
User Story B	3 Story Points
User Story C	5 Story Points
Summe Story Points	13 Story Points
Story Points pro Arbeitsstunde	1,3 Story Points (13/10)

Sprint 2	
Team Kapazität	10 Stunden
User Story D	8 Story Points
User Story E	5 Story Points
User Story F	3 Story Points
User Story G	5 Story Points
Summe Story Points	21 Story Points
Story Points pro Arbeitsstunde	2,1 (21/10)

## Technische Schulden

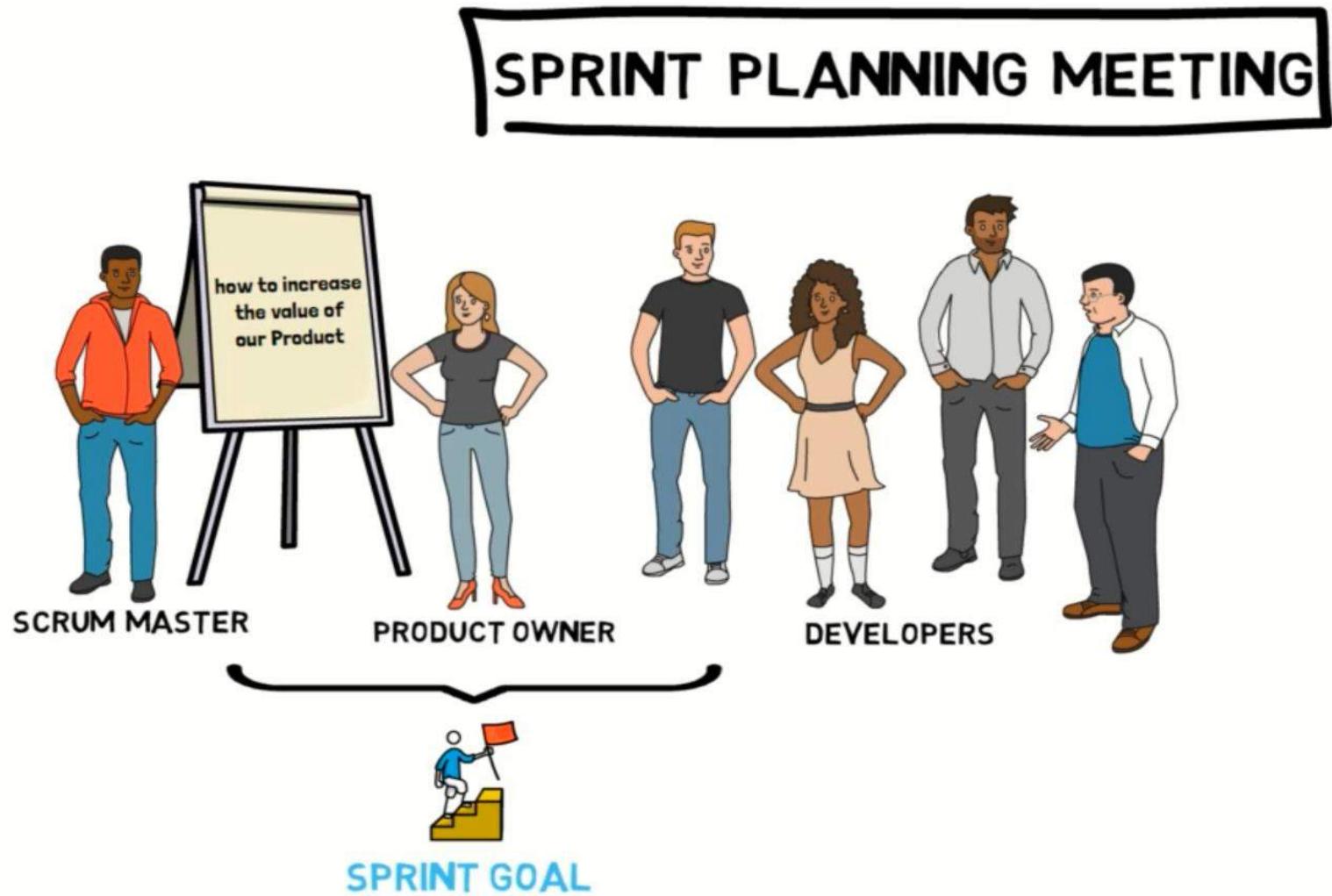


Als Product Backlog Item aufnehmen, schätzen und priorisieren

Pauschal Zeit für die Behebung technische Schulden reservieren

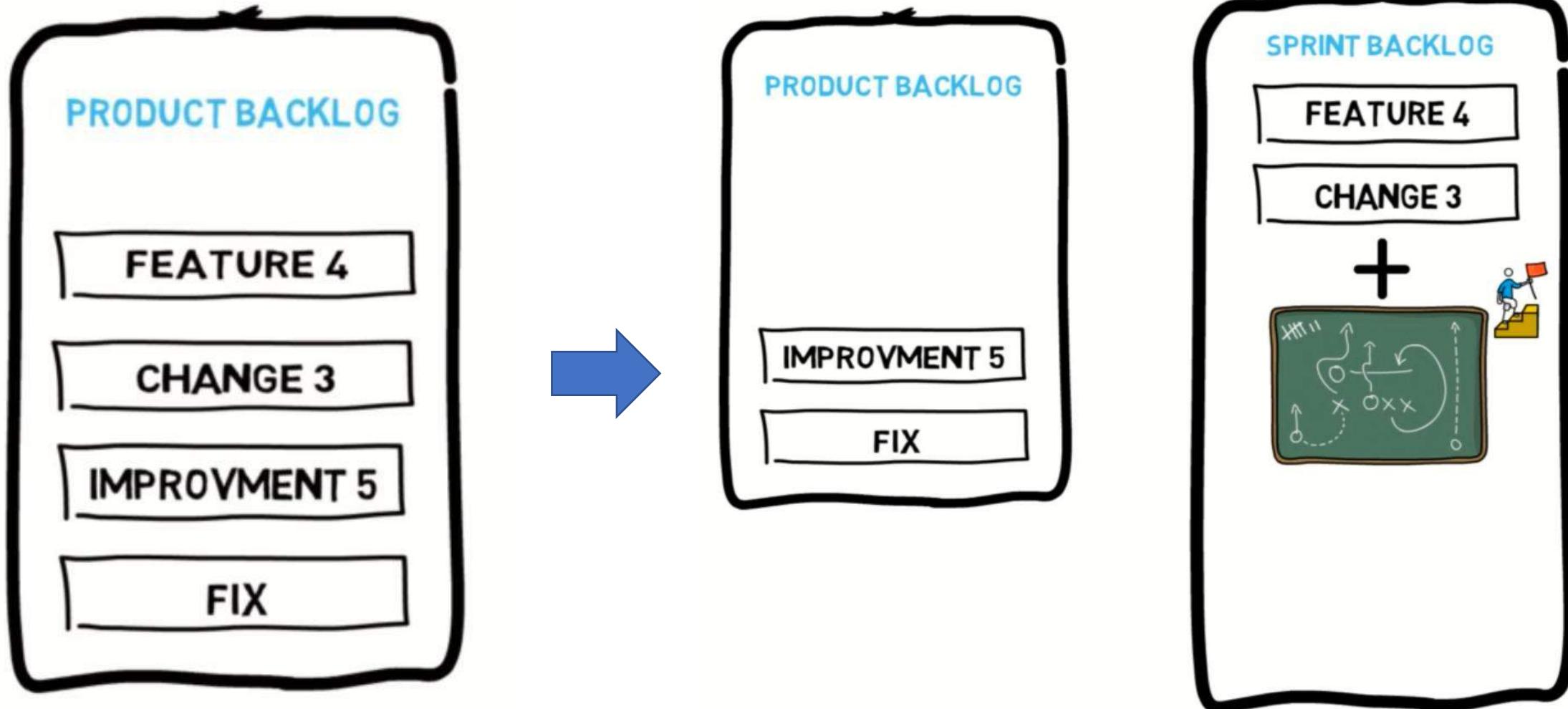
# SCRUM ARTEFAKTE

## SPRINT BACKLOG



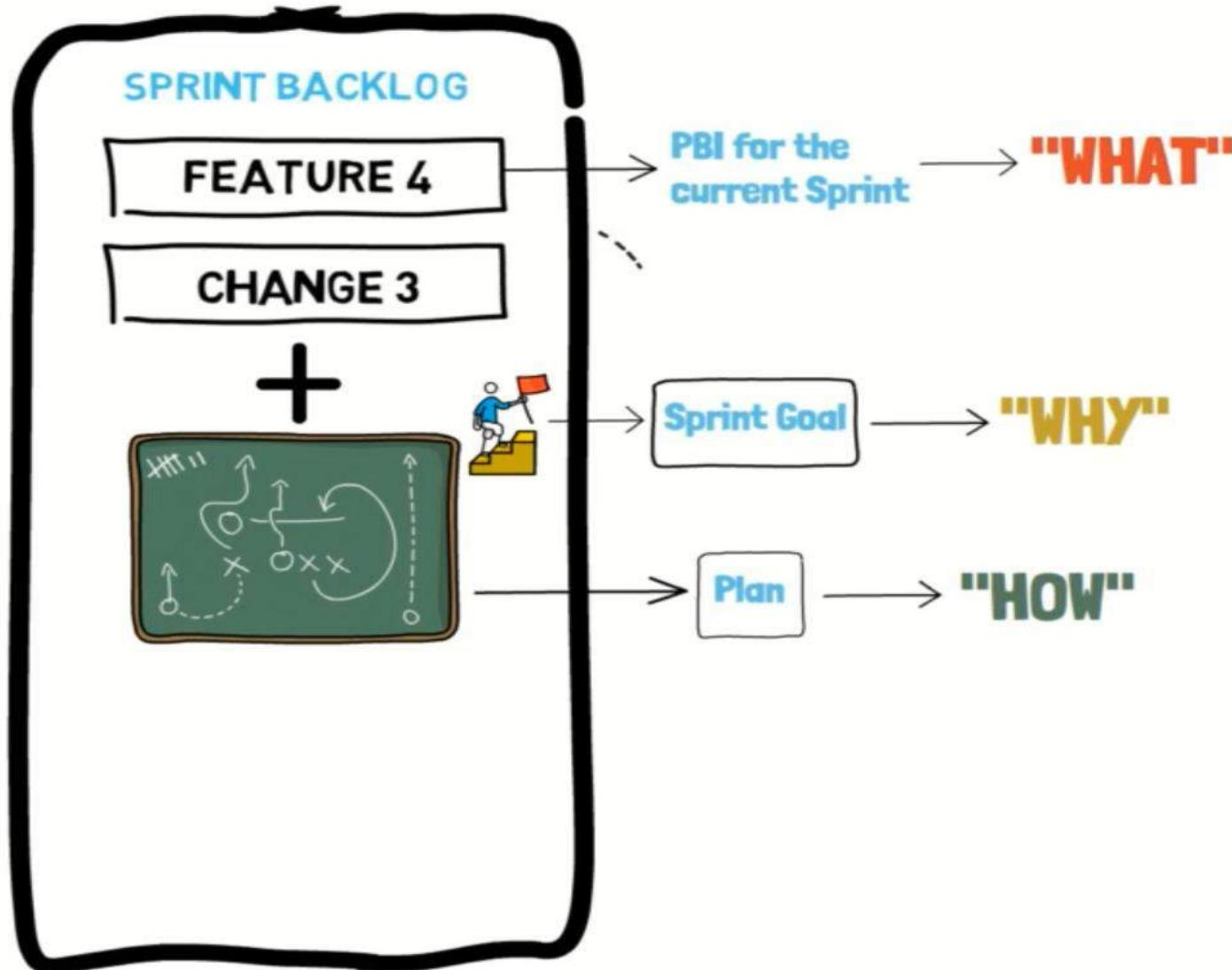
# SCRUM ARTEFAKTE

## SPRINT BACKLOG / SPRINT PLANNING



# SCRUM ARTEFAKTE

## SPRINT BACKLOG / SPRINT PLANNING



PBI  
=  
Product Backlog item  
=  
Product Backlog Gegenstand

# SCRUM ARTEFAKTE

## SPRINT BACKLOG

Das Sprint Backlog enthält jene Anforderungen die im **nächsten** Sprint umgesetzt werden sollen **und** den **Plan** hierzu

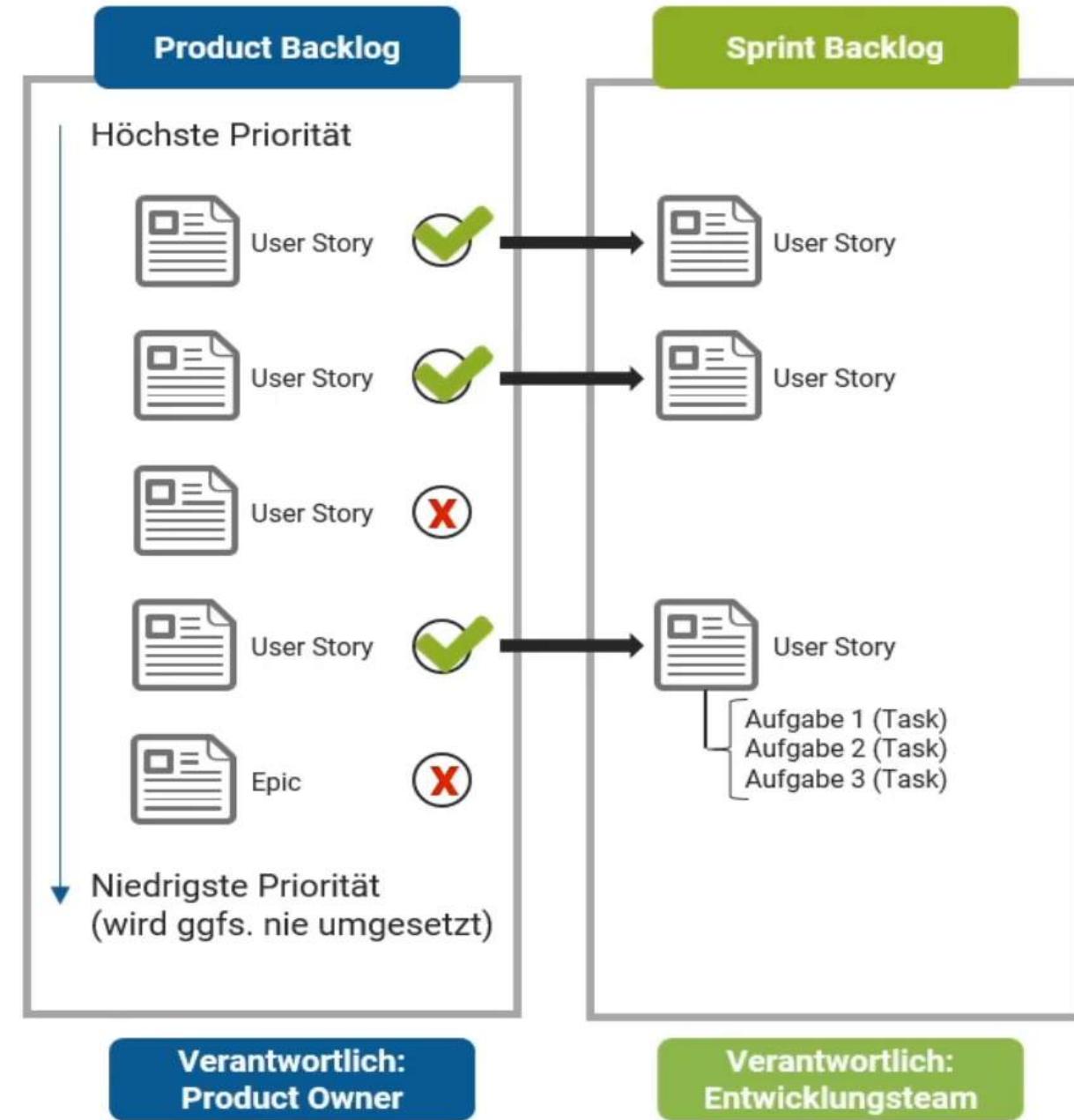
**SCRUM ARTEFAKTE**

**SPRINT BACKLOG**

Das  
**Sprintziel**  
wird vom  
**Scrum  
Team**  
festgelegt

# SPRINT BACKLOG

## BEDEUTUNG DER „DEFINITION OF READY“



# SCRUM ARTEFAKTE

## SPRINT BACKLOG / SPRINT PLANNING



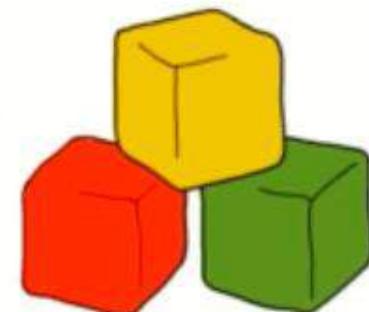
# SCRUM ARTEFAKTE

## PRODUCT INCREMENT / DEFINITION OF DONE

### PRODUCT INCREMENT

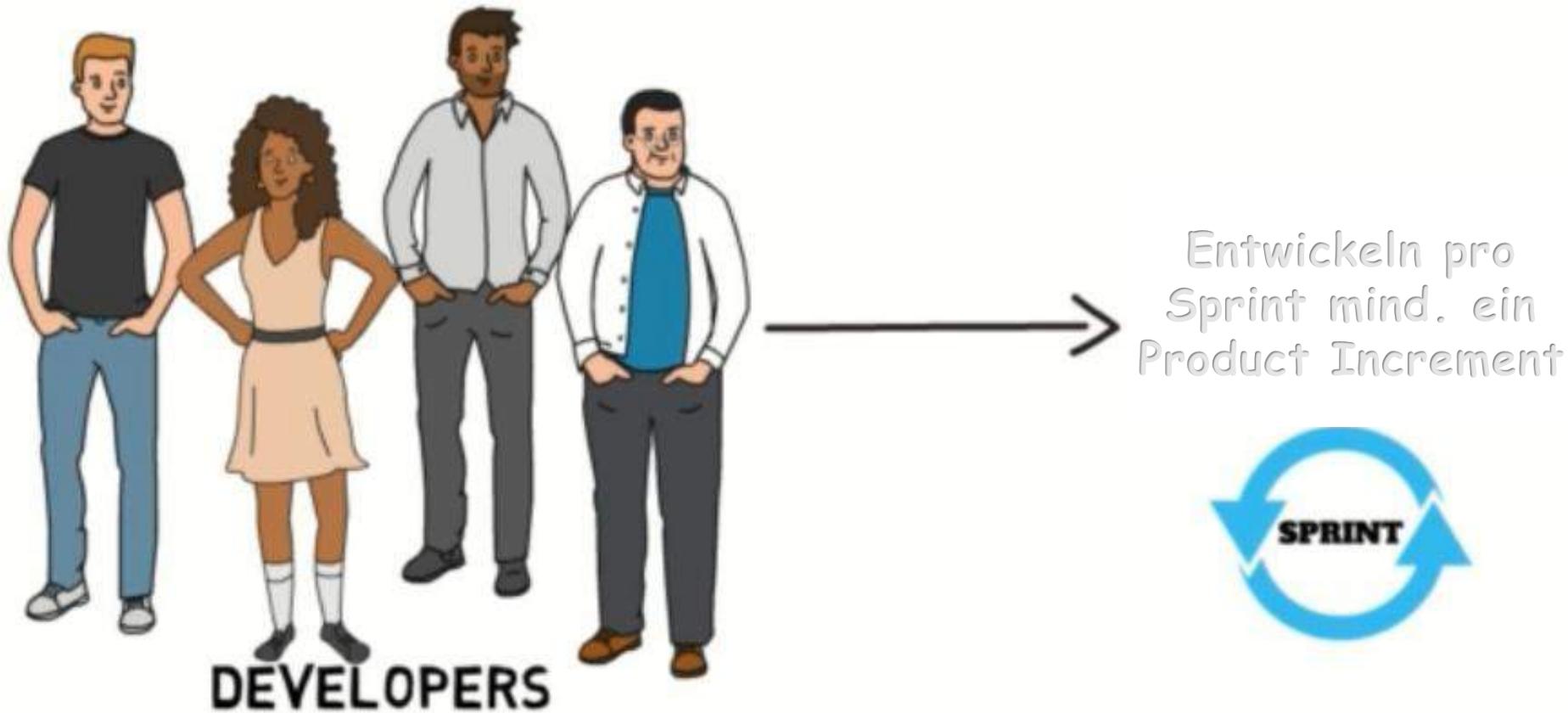
= eine neue Version des Produkts

-> ein ergänzendes Feature zu allen  
vorherigen Inkrementen der vergangenen  
Sprints



# SCRUM ARTEFAKTE

## PRODUCT INCREMENT / DEFINITION OF DONE



# SCRUM ARTEFAKTE

## PRODUCT INCREMENT / DEFINITION OF DONE



Jedes Inkrement muss ohne weitere Entwicklung  
/ Arbeit nutzbar sein  
(z.B. kein zus. Testing)



Der Product  
Owner  
entscheidet,  
ob das  
Inkrement  
veröffentlicht  
(auf den Markt  
gebracht) wird

# SCRUM ARTEFAKTE

## PRODUCT INCREMENT

RELEASE:  
JA ODER NEIN?

### **Beispiel:**

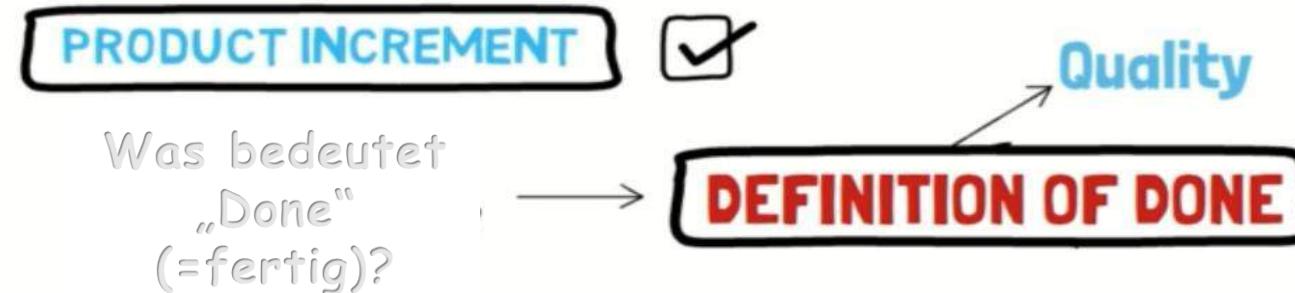
Landing Page  
für eine Website

**Rollout** um erste  
Aufmerksamkeit  
zu erzeugen?

**Warten** bis  
weitere Inhalte  
bereitstehen?

# SCRUM ARTEFAKTE

## PRODUCT INCREMENT / DEFINITION OF DONE



Die Qualitätsstandards des Unternehmens  
müssen mindestens beachtet werden!

**Jack & Mary's minimum requirements:**



# DEFINITION OF DONE: SIND WIR ENDLICH FERTIG?

## Definition of Done

Die „Definition of Done“ ist eine Liste von Kriterien, welche erfüllt sein müssen, bevor eine User Story als „erledigt“ betrachtet werden kann.

Die „Definition of Done“ dient der Qualitätssicherung und sorgt für ein gemeinsames Verständnis, wann die Arbeit an einer User Story als abgeschlossen betrachtet werden kann und welche Voraussetzungen hierfür erfüllt sein müssen.



# DEFINITION OF DONE

## BEISPIEL

### Definition of Done

#### - Ein Beispiel -

- Die Entwicklung von Code / Inhalt für die jeweilige Anforderung ist abgeschlossen und liegt vollständig vor (alle „tasks“ erledigt)
- Die User Story wurde hinsichtlich der definierten Akzeptanzkriterien getestet und alle bekannten Fehler behoben
- Die fachliche Abnahme durch den Product Owner ist erfolgt und die Akzeptanzkriterien erfüllt

---

*Technische, oder IT User Stories können darüber hinaus bspw. noch deutlich konkreter auf bst. Tests eingehen*

---

- Unit Tests abgeschlossen und 100% erfolgreich
- Integration Test abgeschlossen und 100% erfolgreich
- Regression Test abgeschlossen und 100% erfolgreich
- Performance Test abgeschlossen und 100% erfolgreich

# TECHNISCHE SCHULDEN (TECHNICAL DEBT)

WIE GEHE ICH  
MIT BUGS UM?

## Technische Schulden...

### Was ist ein Bug?



### Was passiert wenn wir Bugs schätzen?

Variante 1: Bug wird nicht geschätzt



Die Story Points steigen, obwohl kein zusätzlicher Mehrwert geschaffen wurde!  
Fokus auf technische Exzellenz!

Variante 2: Bug **wird** geschätzt



# TECHNISCHE SCHULDEN (TECHNICAL DEBT)

WIE GEHE ICH  
MIT BUGS UM?

## Technische Schulden...

### Frage

„Des weiteren hätte ich die Frage ob du der Meinung bist das man auch **Bugs** schätzen sollte? Ich bin davon überzeugt das es geht, aber es gibt immer wieder Stimmen die das gegenteilige behaupten. Es fehlen mir die Alternativen?!“



- Technische Schulden „gehören“ dem **Entwicklungsteam**



- Das Entwicklungsteam hat zum Sprintende ein **technisch einwandfreies Inkrement** zu liefern. Bugs / Defects sind im laufenden Sprint sofort zu beheben.



- Die **Definition of Done** sollte sicherstellen, dass keinerlei technische Schulden entstehen

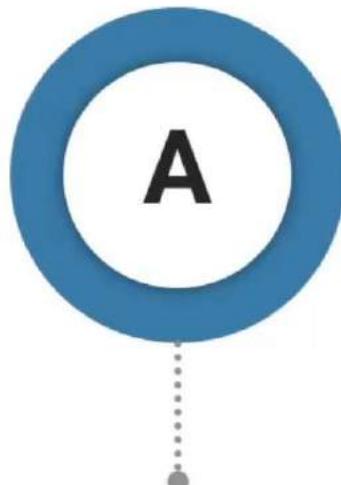


- Eine **Ansammlung technischer Schulden ist unbedingt zu vermeiden**, da die Lösung zu späteren Zeitpunkten immer schwieriger wird und mehr Zeit für die Umsetzung neuer Funktionalitäten fehlen könnte

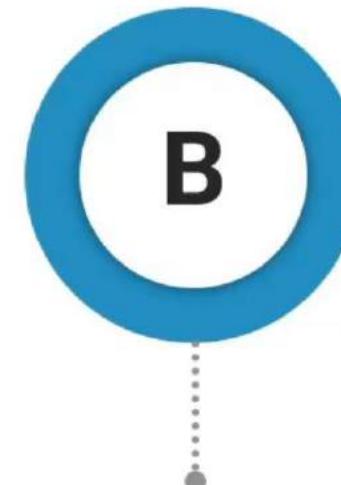
## TECHNISCHE SCHULDEN (TECHNICAL DEBT)

WIE GEHE ICH  
MIT BUGS UM?

# Technische Schulden...



**Technische Schulden als  
Product Backlog Item**  
Technische Schulden werden  
beschrieben und geschätzt und ins  
Product Backlog aufgenommen.  
Dort erfolgt eine Priorisierung ggü. allen  
anderen Anforderungen



**Sprintkapazität  
reservieren**  
Eine bestimmte Sprintkapazität wird  
pauschal "geblockt" um in jedem Sprint  
etwas Zeit für die Behebung  
technischer Schulden einzuplanen

# SCRUM ARTEFAKTE ZUSAMMENFASSUNG

## Artefakte

Als „Artefakt“ werden Elemente beschrieben, die zur Steuerung des Vorhabens erforderlich sind. Die Inhalte dieser Artefakte verändern sich regelmäßig

## Anforderungen im Backlog

Anforderungen im Product Backlog sollten drei Eigenschaften enthalten:

1. Inhaltlich beschrieben (description)
2. Priorisiert (value, order)
3. Geschätzt (estimated)

## Product Backlog



- Liste aller gewünschten Anforderungen
- Anforderungen sind bestenfalls als „User Story“ beschrieben und verfügen über Akzeptanzkriterien
- Anforderungen sind priorisiert nach Business Nutzen
- Anforderungen sind geschätzt
- Sichtbar und Transparent für alle Stakeholder und Scrum Team Mitglieder
- Regelmäßig durch den Product Owner überprüft und aktualisiert
- „Lebendiges“ Artefakt, welches stets durch neue Anforderungen ergänzt werden kann

## Commitment Product Goal



## Product Backlog

Auswahl der **wichtigsten** Anforderungen (Product Backlog Items) für den nächsten Sprint!

## Sprint Backlog



- Besteht aus jenem Teil der Anforderungen des Product Backlog, welche für den nächsten Sprint relevant sind und dem Plan zur Umsetzung
- Inhalte des Sprint Backlogs werden zwischen Product Owner und Entwicklungsteam ausgehandelt, hierbei spielen Aufwand und verfügbare Kapazität eine wichtige Rolle
- Wird vom Entwicklungsteam in einzelne Tasks heruntergebrochen und täglich nachverfolgt (bspw. Kanban-Board)

## Commitment Sprint Goal



Planung der operativen Umsetzung (Tasks)

## Product Increment



- Ergebnis des aktuellen und aller vorherigen Sprints, welches potenziell ausgeliefert werden könnte.
- Muss getestet und einsatzbereit sein
- Das Product Increment wächst mit jedem Sprint und erweitert, oder verbessert die bereits bestehenden Funktionalitäten
- Ist die Grundlage für Feedback in den regelmäßigen Sprint-Review Meetings

## Commitment Definition of Done



## Product Increment

Potentiell auslieferbares Ergebnis (Increment)

# ROLLEN UND VERANTWORTUNGEN IN SCRUM

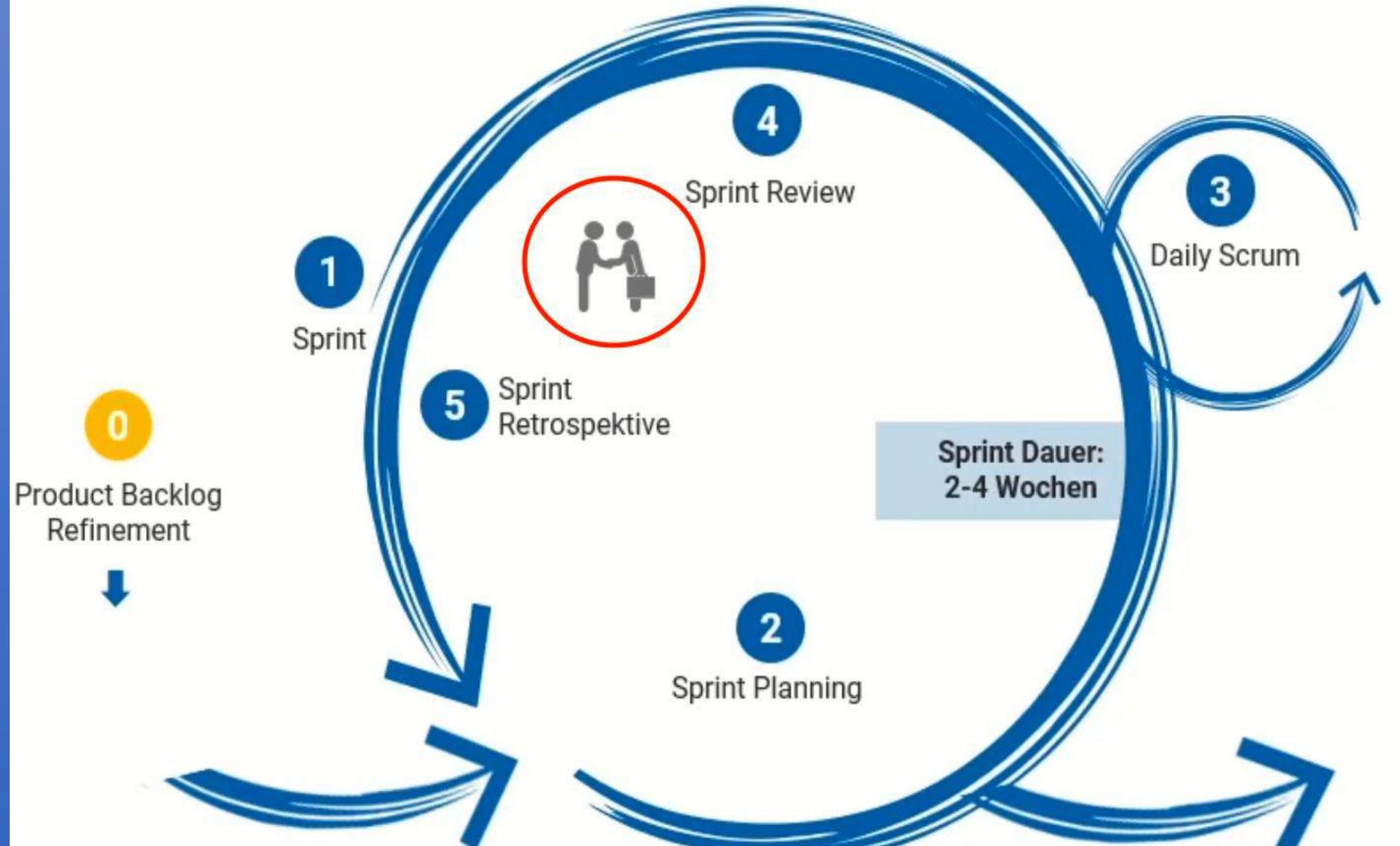


Agile Teams bestehen aus einem Scrum Master, einem Product Owner und dem Entwicklungsteam. Stakeholder (bspw. aus betroffenen Fachbereichen sind keine offizielle Gruppe, spielen aber dennoch eine wichtige Rolle)



## ROLLEN IN SCRUM

### STAKEHOLDER



Product  
Backlog



Sprint  
Backlog



Product  
Increment

# ROLLEN UND VERANTWORTUNGEN IN SCRUM

WIRTSCHAFTLICHER ERFOLG



Product Owner (PO)

OPTIMALE ANWENDUNG DER  
PROZESSE



Scrum Master

TECHNISCHE QUALITÄT



Team

# ROLLEN UND VERANTWORTUNGEN IN SCRUM

**Product Owner**



**Scrum Master**



**Development Team**

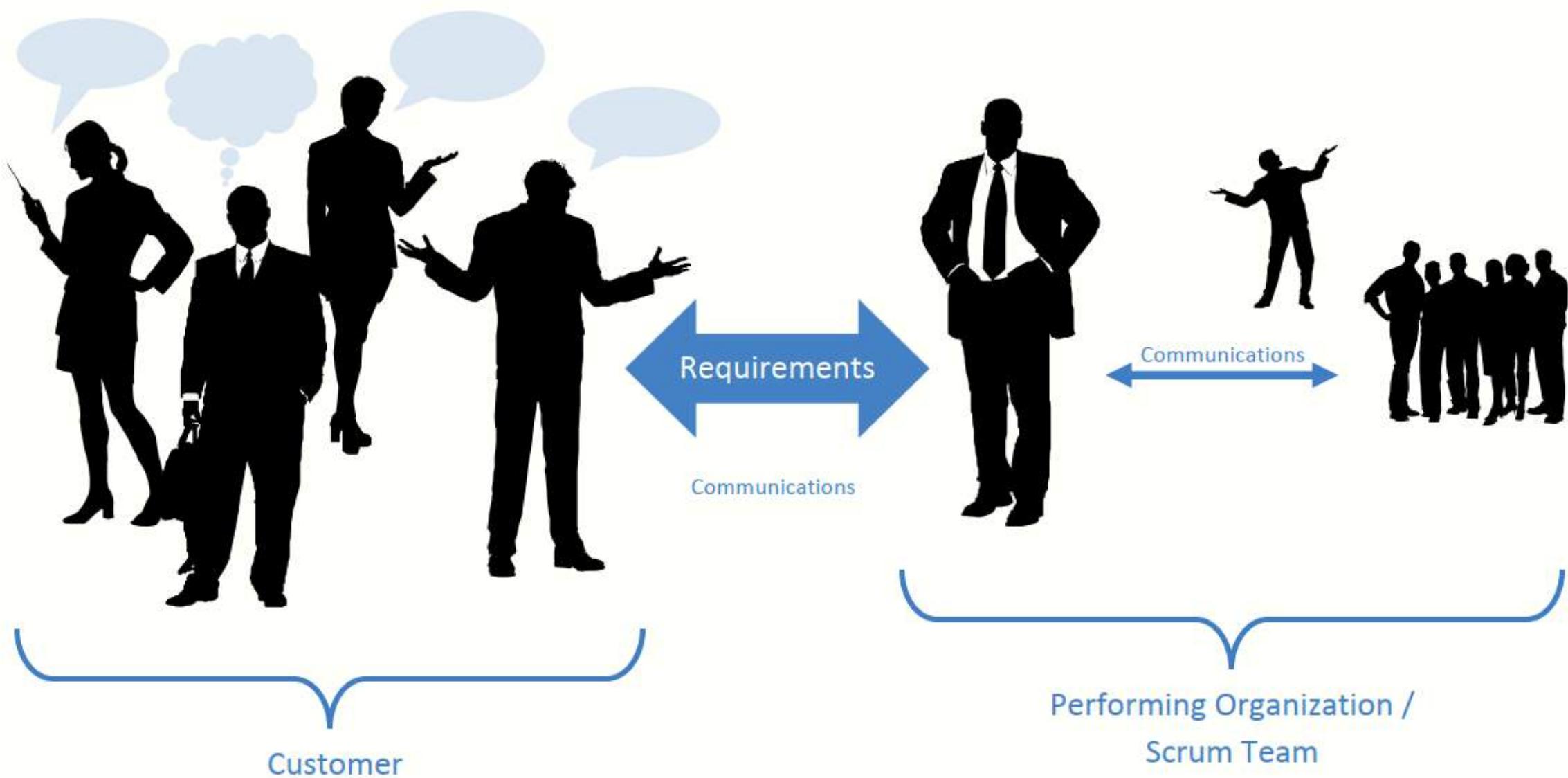


1 person  
Full-time or part-time  
Business-oriented

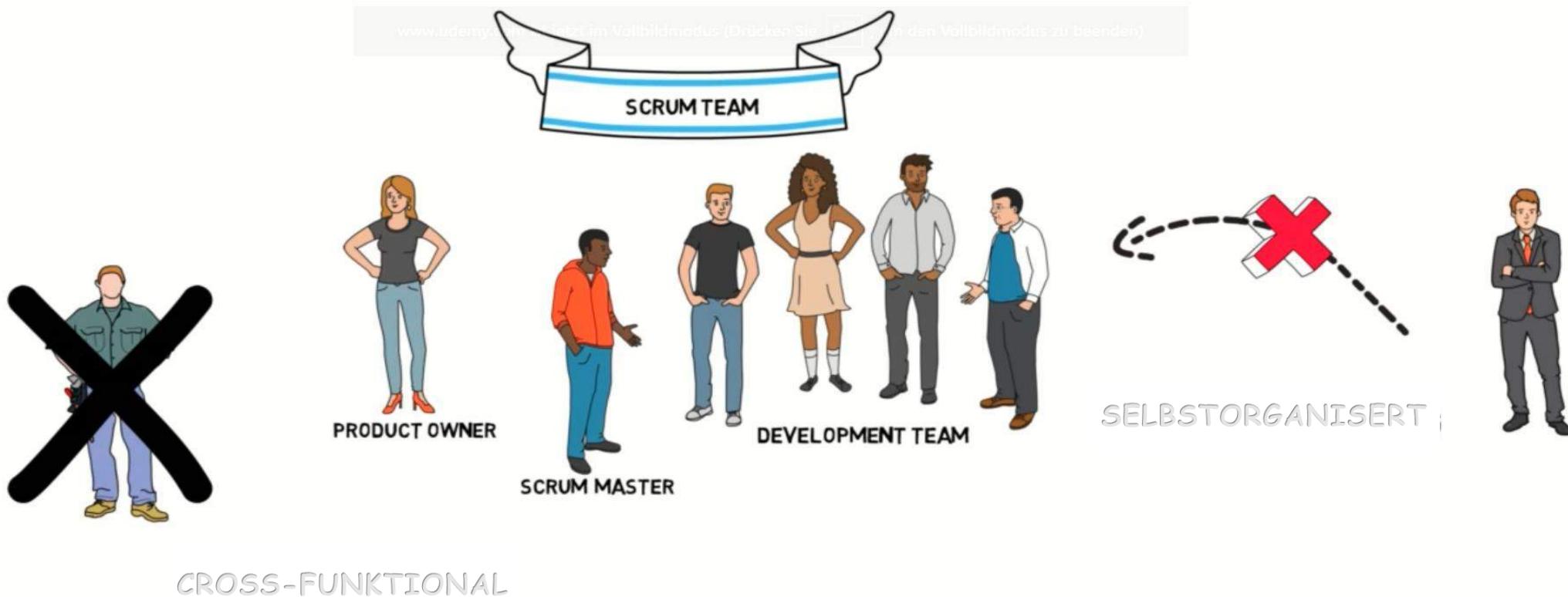
1 person  
Full-time or part-time  
Scrum coach and facilitator

3 to 9 people  
Full-time (recommended)  
Specialists

# ROLLEN UND VERANTWORTUNGEN IN SCRUM



# SCRUM ROLLEN DAS SCRUM TEAM



# SCRUM ROLLEN DAS SCRUM TEAM

DAS SCRUM TEAM IST DARAUF AUSGERICHTET, EIN MAXIMUM IN FOLGENDEN BEREICHEN ZU ERLANGEN:



FLEXIBILIÄT



KREATIVITÄT



PRODUKTIVITÄT

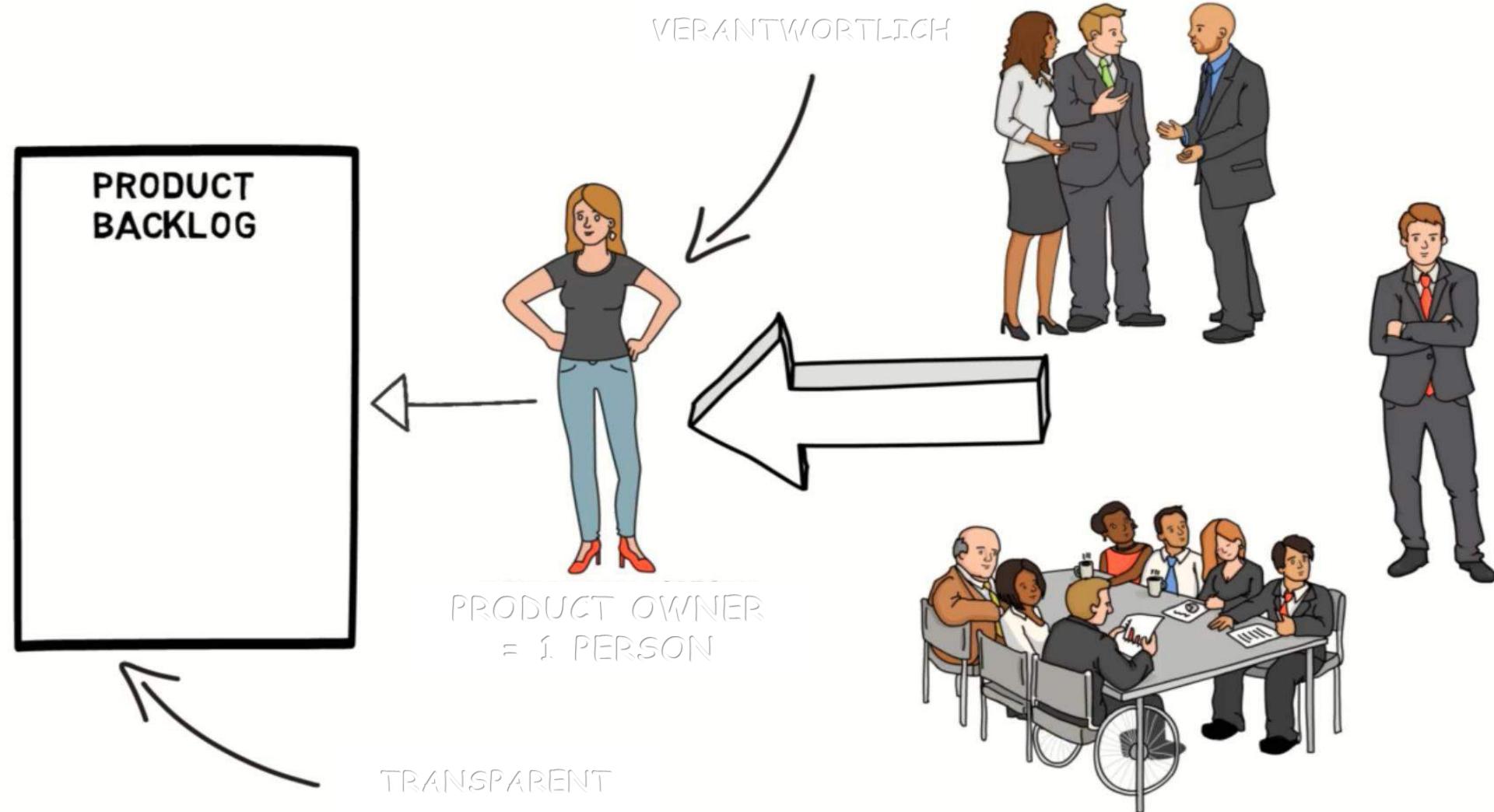
# 3 SCRUM ROLLEN: DER PRODUCT OWNER (PO)

## Product Owner

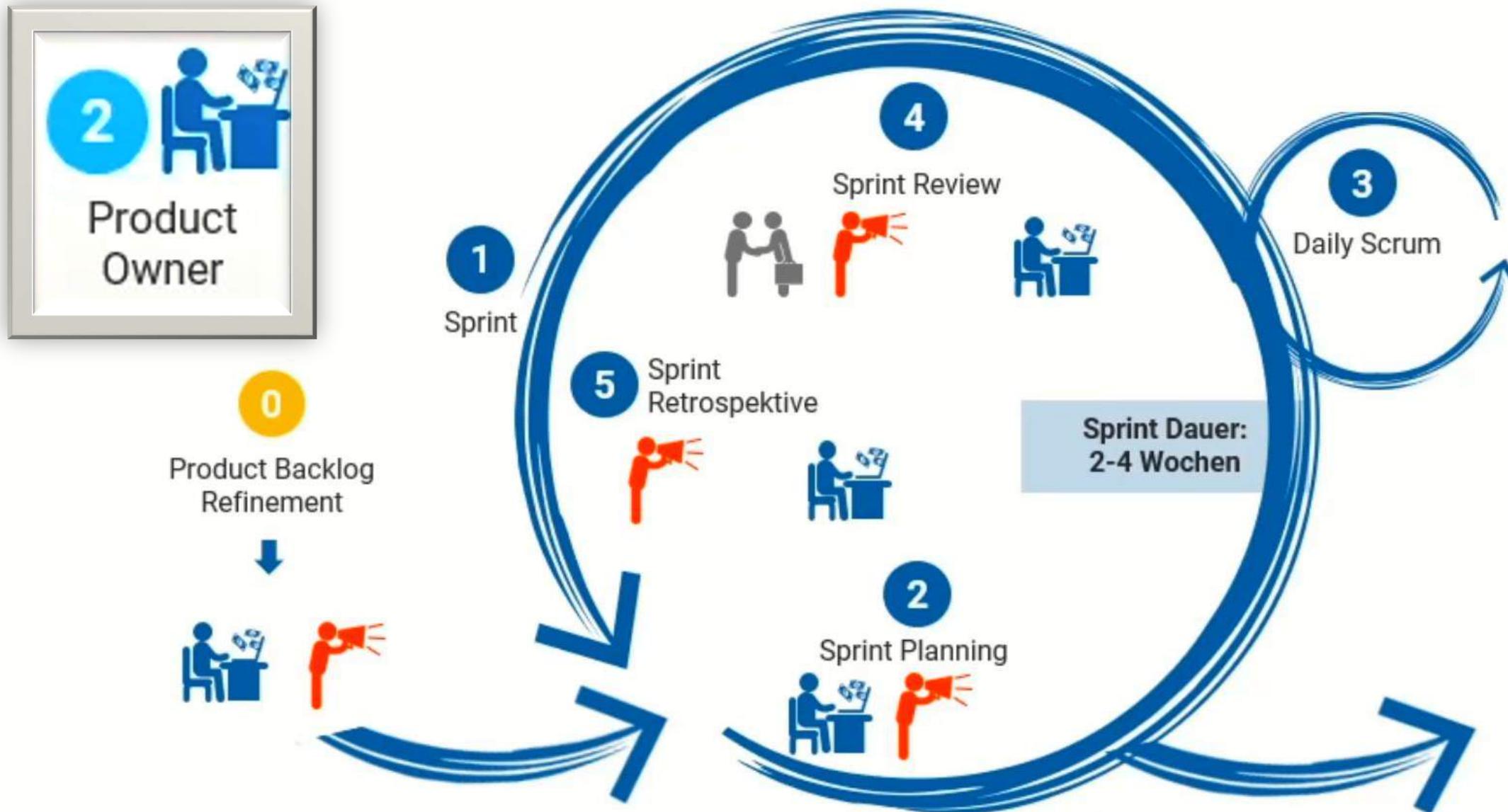
Der Produkt Owner ist für den wirtschaftlichen Erfolg des Produkts und die Maximierung des Business Nutzens verantwortlich. Er hat das letzte Wort zu allen Anforderungen und Produkteigenschaften



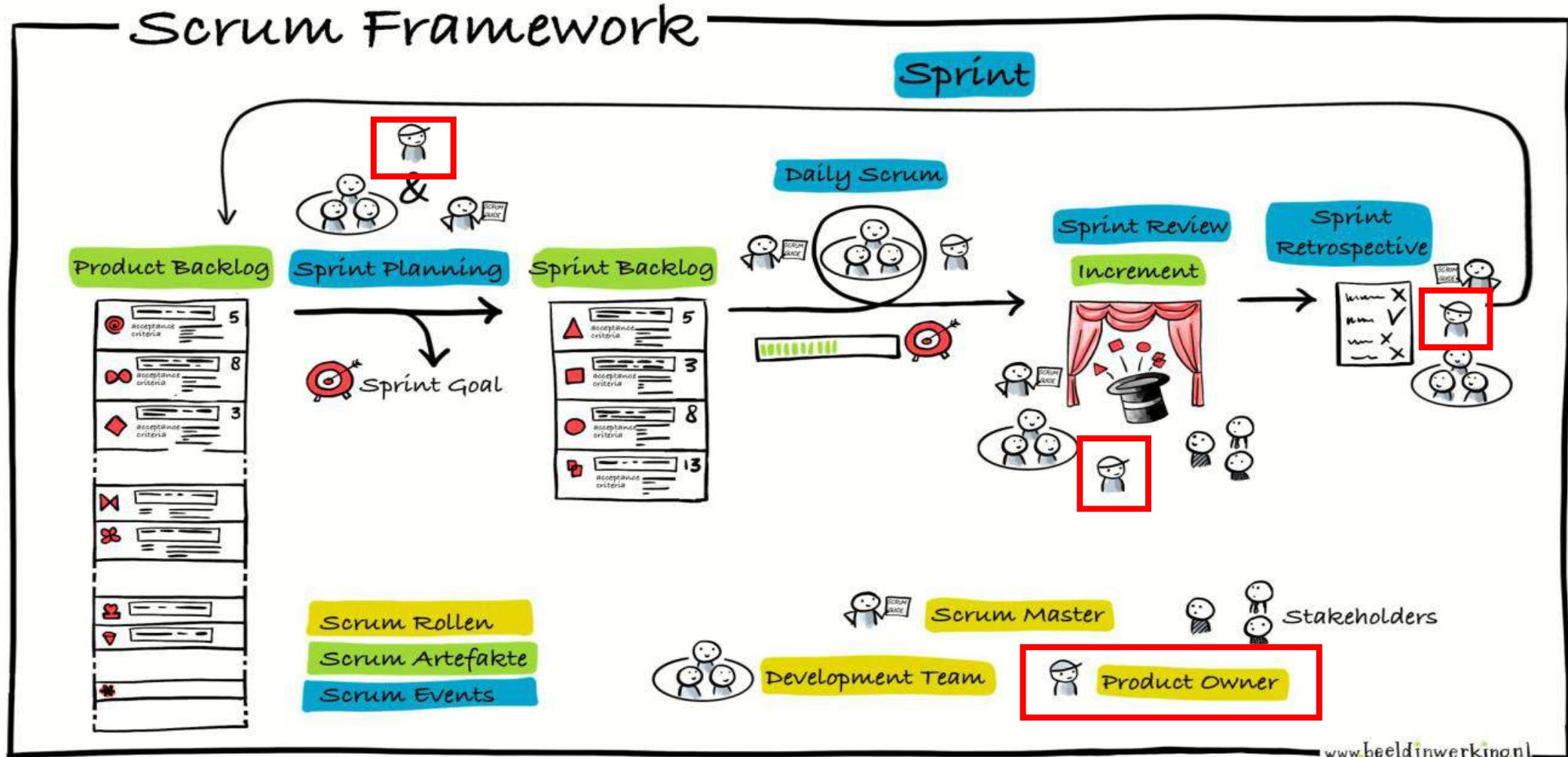
# SCRUM TEAM / SCRUM ROLLEN DER PRODUCT OWNER



# 3 SCRUM ROLLEN: DER PRODUCT OWNER (PO)



# SCRUM TEAM / SCRUM ROLLEN DER PRODUCT OWNER



# 3 SCRUM ROLLEN: DER PRODUCT OWNER (PO)

- Fachexperte
- Schnittstelle zwischen Team, Kunde und Nutzer
- Erstellt und pflegen das Product Backlog
- Priorisiert Anforderungen
- Nimmt an folgenden Events teil
  - Sprint Planning
  - Backlog Grooming
  - Review



# ROLLEN UND VERANTWORTUNGEN IN SCRUM

## PRODUCT OWNER

### Product Owner



- Der Product Owner ist für die **Maximierung des Geschäftswerts** (value of the product) verantwortlich
- Der Product Owner ist auch für den **bestmöglichen Einsatz** des Entwicklungsteams verantwortlich
- **Einzelne Person**, die als "Stimme des Kunden" agiert

# PRODUCT OWNER

# Product Owner

Der Produkt Owner ist für den wirtschaftlichen Erfolg des Produkts und die Maximierung des Business Nutzens verantwortlich. Er hat das letzte Wort zu allen Anforderungen und Produkteigenschaften

## Anforderungen

- Gutes Verständnis, bzw. klare **Vision vom Endprodukt und Marktumfeld**
- **Wirtschaftliche Denkweise** und ggf. technisches Verständnis
- Muss sich in die **Lage der Stakeholder** und des **Entwicklungsteams** versetzen können und diese einbinden
- **Fähigkeit zu priorisieren** und **Prioritäten** klar zu **kommunizieren**
- **Vollzeit verfügbar** und aktiv involviert (bspw. für Rückfragen und Abnahmen)
- **Akzeptanz** im Unternehmen und Fähigkeit kurzfristige Entscheidungen für das Produkt zu treffen

## Verantwortung

- **Product Backlog Management**
- **Product Backlog Items** (User Stories, Epics) werden klar **formuliert**
- **Gewährleistung**, dass die Product Backlog Items vom Scrum Team **verstanden** werden
- **Priorisiert Anforderungen** um das Produktziel bestmöglich umzusetzen
- **Sinnvoller Einsatz** des Entwicklungsteams
- Gewährleistung der **Transparenz** des Produkt Backlogs

**Entwicklung und Kommunikation des Product Goal**

## Aufgaben

- **Sicherstellung**, dass die **richtigen** (am höchsten priorisierten) User Stories umgesetzt werden
- **Regelmäßige Überprüfung des Product Backlogs** hinsichtlich
  - Prioritäten
  - Aktualität
  - Qualität von User Stories
  - Einbeziehung v. Stakeholdern
- **Kontinuierliche Abstimmung** mit dem Entwicklungsteam
- **Identifikation und Erstellung** von User Stories (dürfen auch andere Rollen!)
- Definition und Überprüfung von **Akzeptanzkriterien**

# SCRUM TEAM / SCRUM ROLLEN DER PRODUCT OWNER

Product Owner

Die Hauptverantwortung des Product Owner ist die

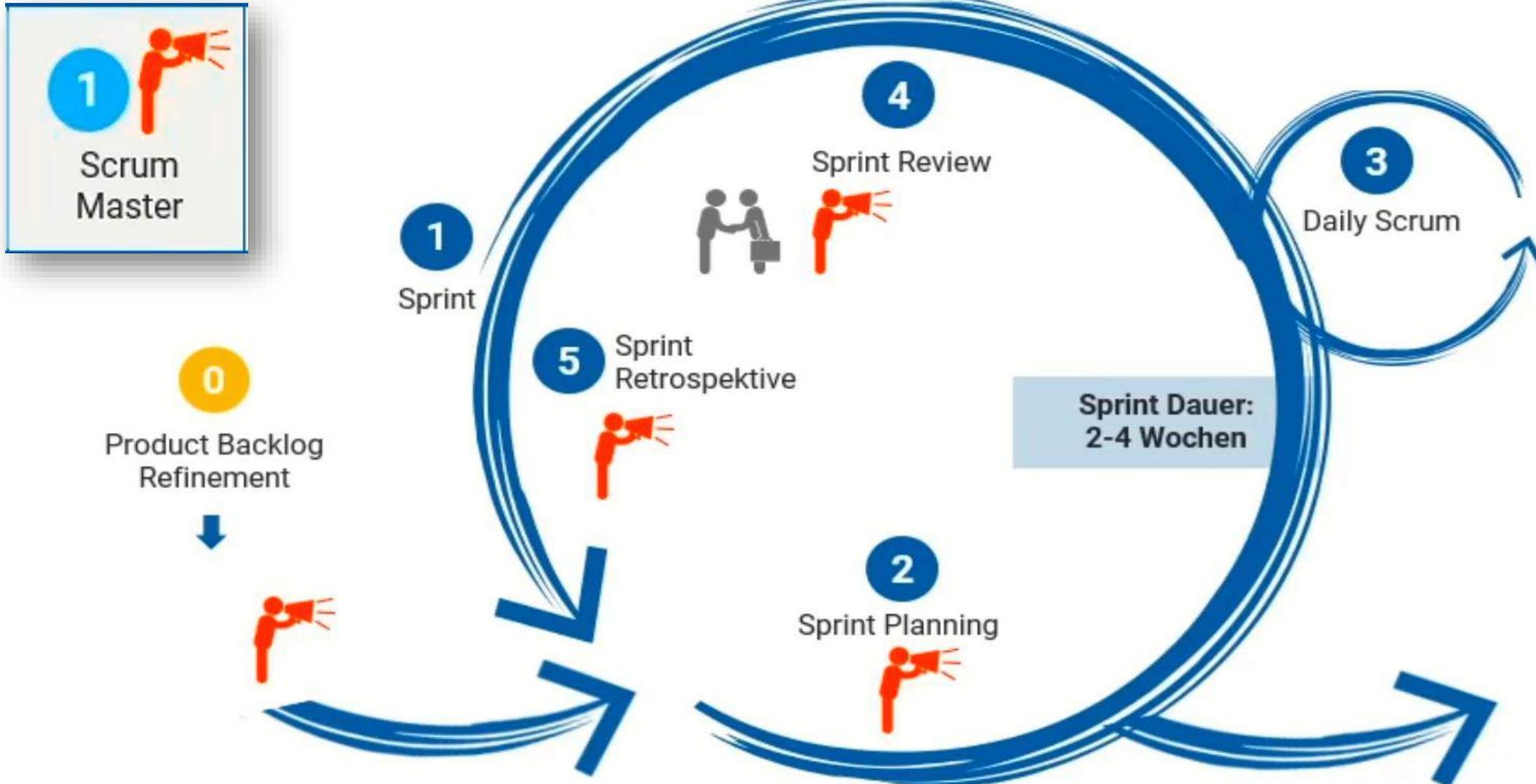
Maximierung des Wertes  
der Entwicklungsarbeit im Team

Was bedeutet Wert?

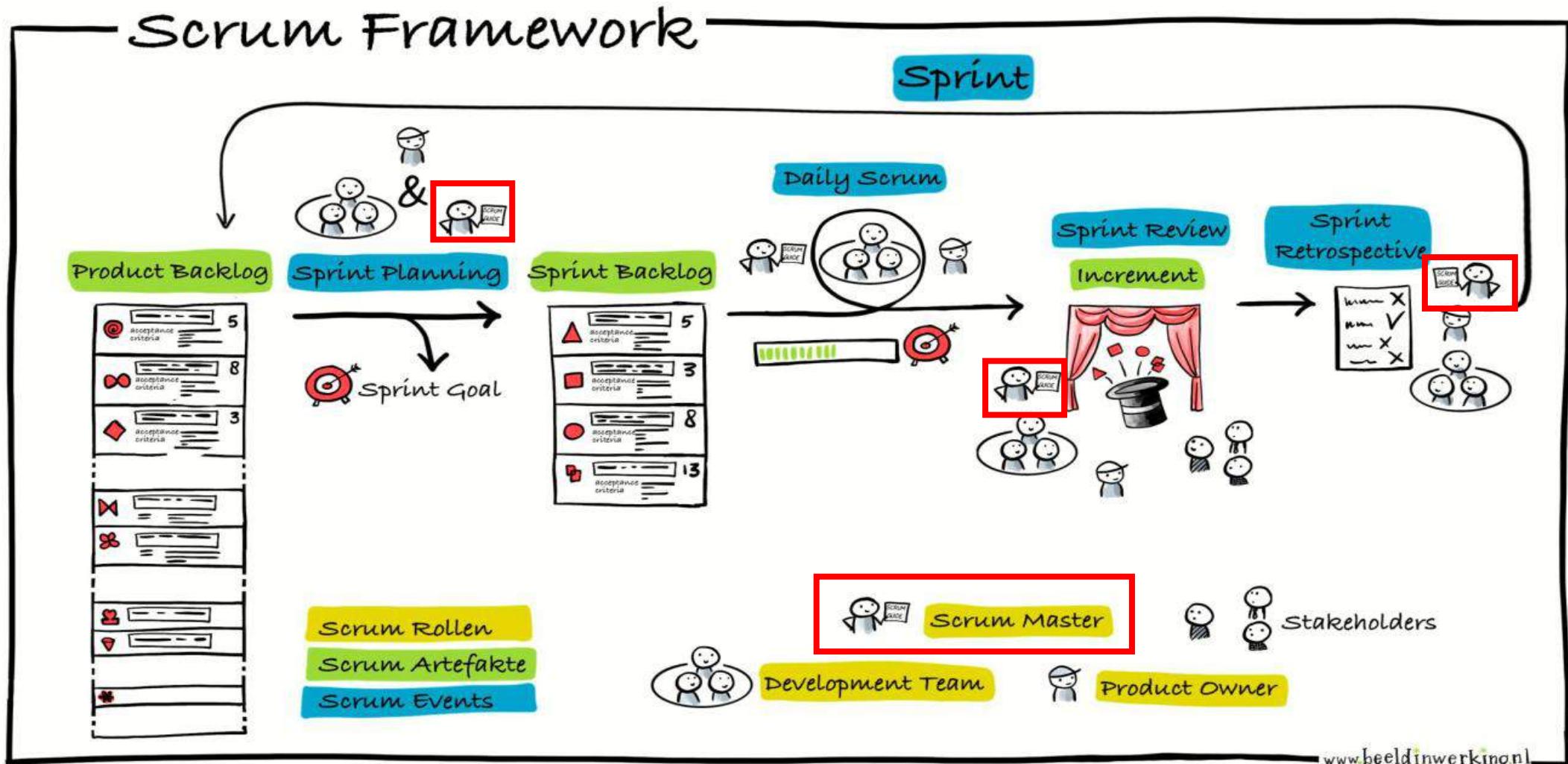
- Business Wert
- Wert für den Kunden
- Technischer Wert

Es gibt  
KEINE PROJEKTMANAGER  
Rolle in Scrum

# 3 SCRUM ROLLEN: DER SCRUM MASTER



# SCRUM TEAM / SCRUM ROLLEN DER SCRUM MASTER



# ROLLEN UND VERANTWORTUNGEN IN SCRUM

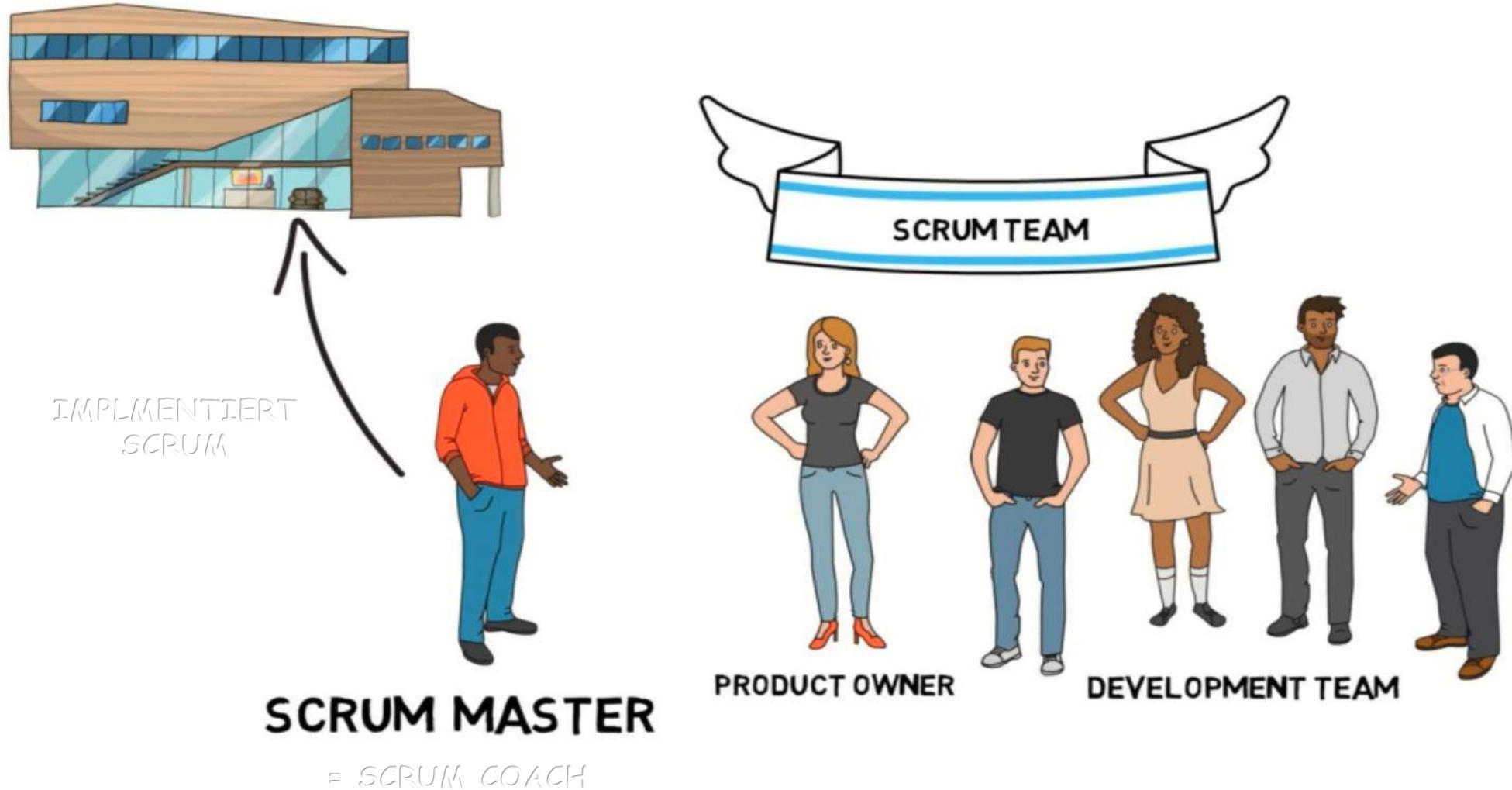
## SCRUM MASTER

### Scrum Master

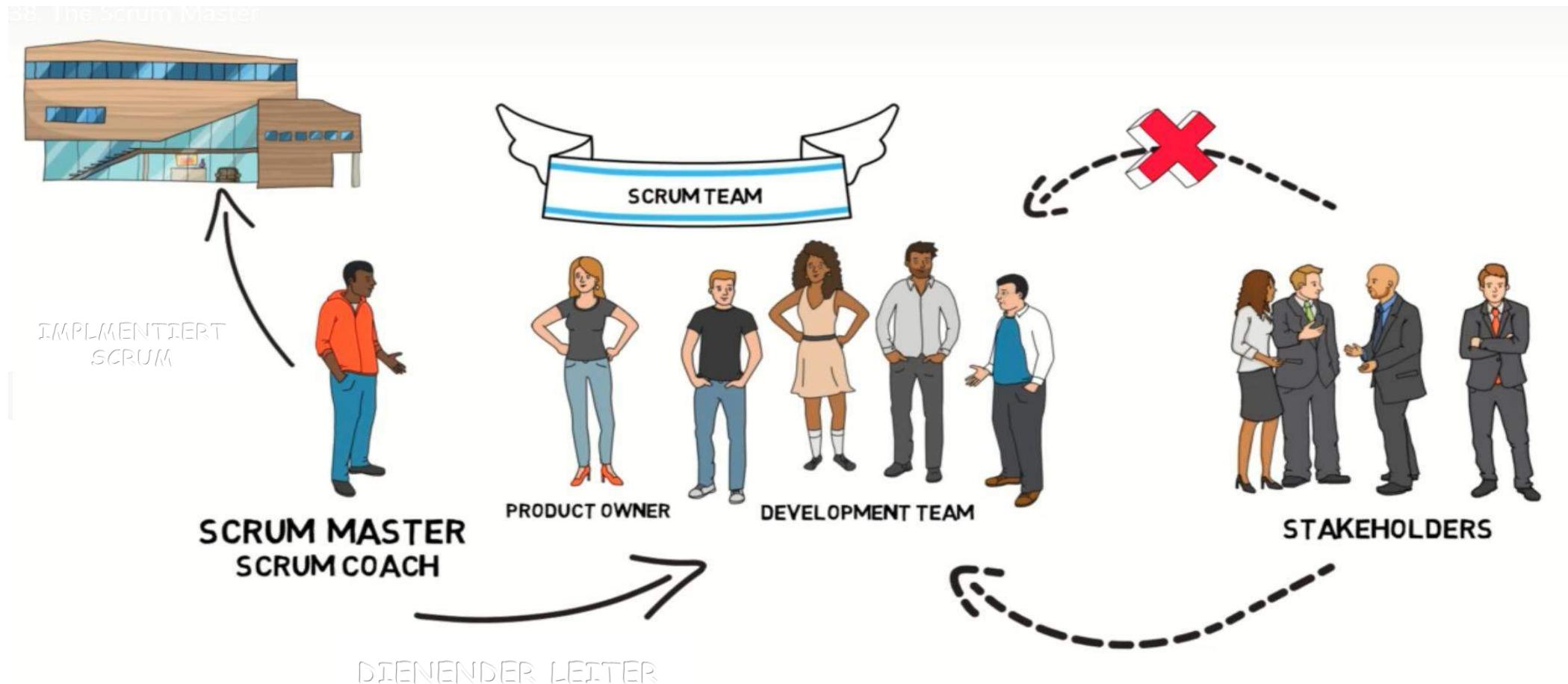


- Verantwortlich für das Verständnis und die **Einhaltung** von **Scrum Prozessen** und **festgelegten Regeln**
- **Servant-Leader** für das Scrum Team
- Hilft **Außenstehenden** bei der Interaktion mit dem Scrum Team und hilft den **Nutzen** dieser **Interaktion** zu maximieren
- Schafft ein **Umfeld** in dem das **Entwicklungsteam selbst-organisiert** arbeiten kann

# SCRUM TEAM / SCRUM ROLLEN DER SCRUM MASTER



# SCRUM TEAM / SCRUM ROLLEN DER SCRUM MASTER



# 3 SCRUM ROLLEN: DER SCRUM MASTER

## Scrum Master – Verantwortung

- Verantwortlich für Scrum-Prozess
- Sogar verantwortlich für Implementierung von Scrum im Unternehmen
- Schafft Umfeld, in dem Scrum funktionieren kann
- Pioneer für das Thema Scrum im Unternehmen
- Moderiert Change-Prozess

# 3 SCRUM ROLLEN: DER SCRUM MASTER

## Servant Leader

- Scrum Master als „dienende Führungskraft“
- Völlig neues Verständnis von Führung
- Führung ohne Vorschreiben von Regeln, sondern durch Fokussierung auf Interessen des Teams/den Mitarbeiter
- Fordert Agiles Methodenset

# 3 SCRUM ROLLEN: DER SCRUM MASTER

## Aufgaben des Scrum Masters

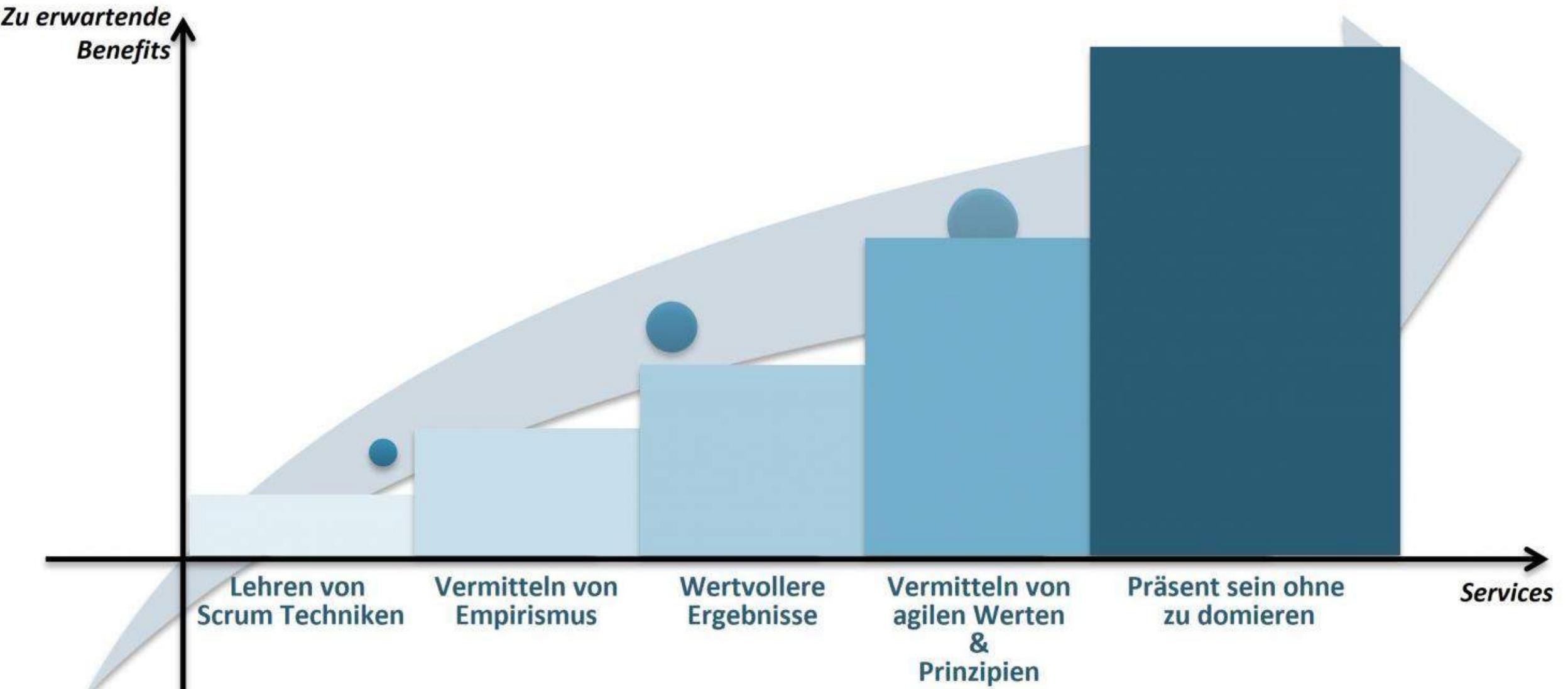
Scrum Master

Service dem Team  
gegenüber

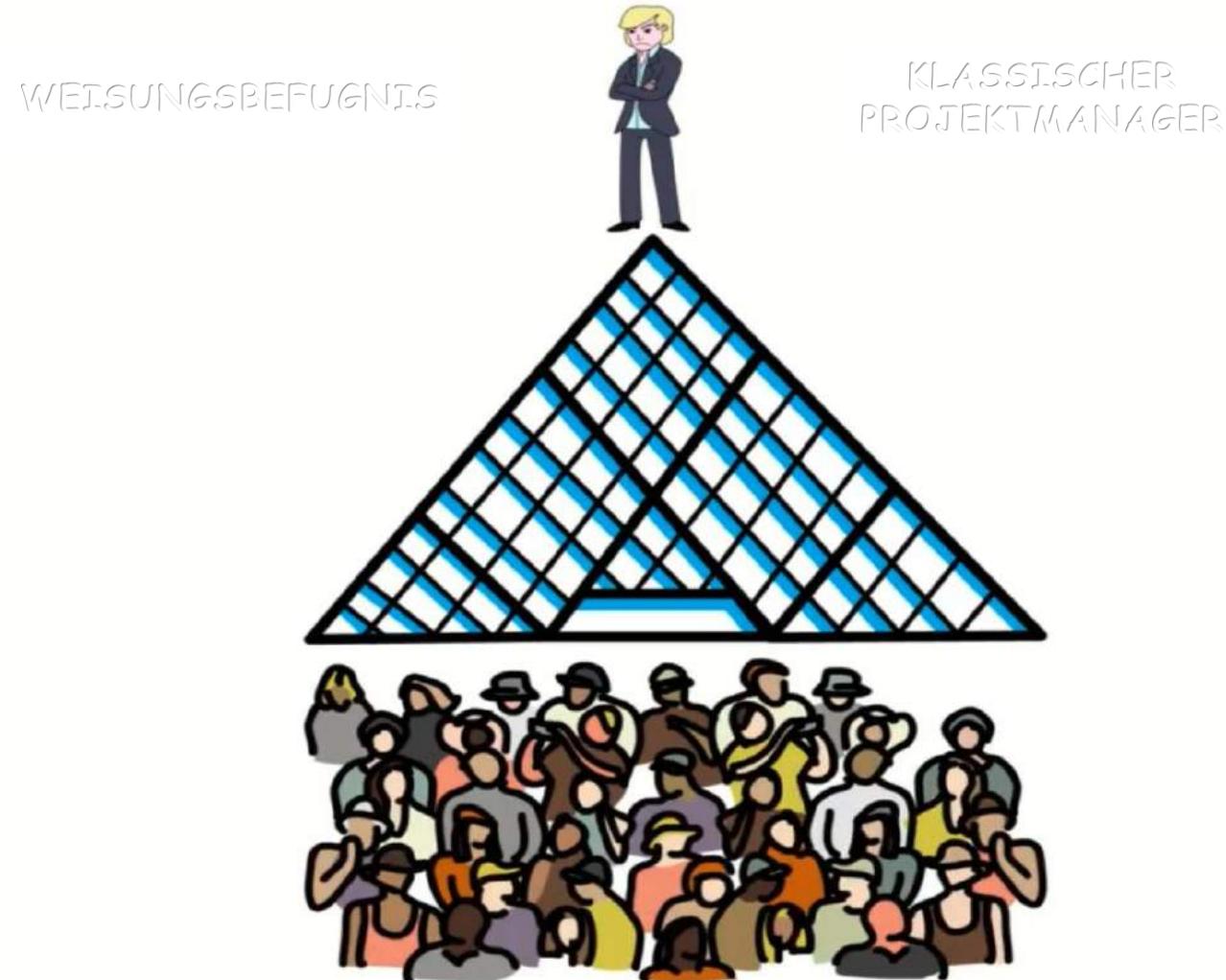
Service dem  
Product Owner  
gegenüber

Service der  
Organisation  
gegenüber

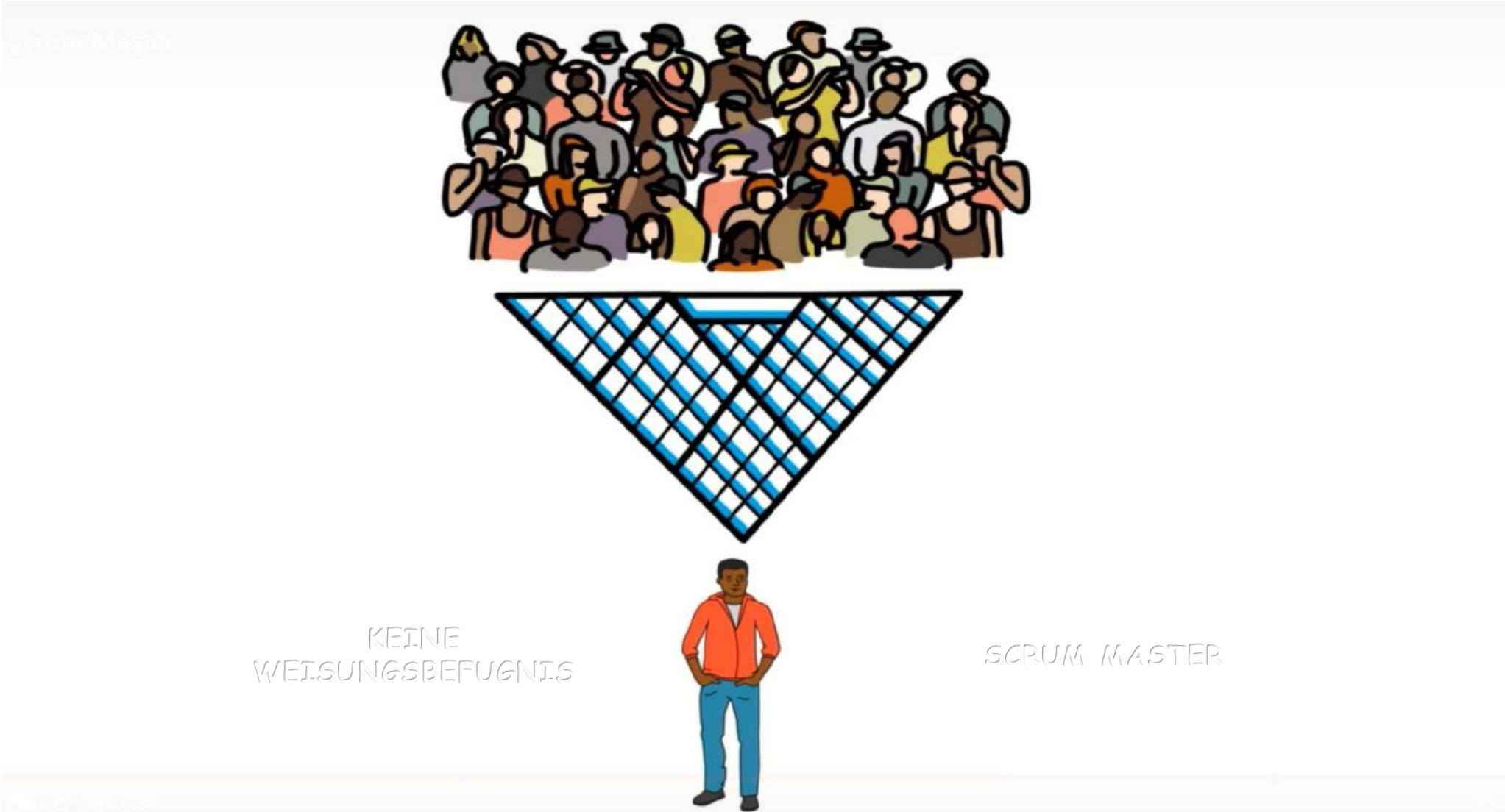
# DER SCRUM MASTER: EIN „SERVANT LEADER“



# SCRUM TEAM / SCRUM ROLLEN DER SCRUM MASTER



# SCRUM TEAM / SCRUM ROLLEN DER SCRUM MASTER



# SCRUM MASTER

## Scrum Master

Der Scrum Master ist für die Einhaltung der Scrum Prozesse verantwortlich und stellt sicher, dass das Scrum Team effektiv arbeiten kann

### Anforderungen

- **Kenntnis von Scrum Prozessen** und der praktischen Umsetzung
- „**Dienstleister**“ für das Entwicklungsteam und den Product Owner
- Fähigkeit „**Brücken zu bauen**“ und zwischen verschiedenen Parteien zu vermitteln
- **Lösungsorientierung** und pragmatische Denkweise
- Fähigkeit Teammitglieder **zu motivieren** und einzubinden
- Muss zuhören können und ein **gutes Gespür für zwischenmenschliche Spannungen** haben

### Verantwortung

- **Servant – Leader**: Fokus auf die Bedürfnisse des Scrum Teams und seiner Kunden
- **Facilitator**: Sicherstellung, dass das Entwicklungsteam in der Lage ist effektiv zu arbeiten
- **Coach / Mentor**: Coaching mit Fokus auf Mindset und Verhalten hinsichtlich kont. Verbesserung und hinsichtlich der Scrum Prozesse
- **Moderator**: spricht unproduktives Verhalten an, hilft bei der Festlegung des Sprintziels
- **Manager**: Management von Prozessen, Zeit, Impediments, unprod. Verhalten, Selbst-Organisation und Teamkultur

### Aufgaben

- Service / Unterstützung für den **Product Owners**
- Service / Unterstützung für das **Entwicklungsteam**
- Service für das **Unternehmen**

# SCRUM MASTER EIN „SERVANT LEADER“

# SCRUM MASTER

## Scrum Master - Services

Der Scrum Master ist für die Einhaltung der Scrum Prozesse verantwortlich und stellt sicher, dass das Entwicklungsteam effektiv arbeiten kann

### Service für den Product Owner

- Sicherstellen, dass **Ziele, Inhalte** und das **Arbeitsumfeld** vom gesamten Scrum Team verstanden werden
- **Auswahl von Techniken** zum managen des **Product Backlogs**
- **Hervorheben** der Notwendigkeit von klaren und **prägnanten Product Backlog Items**
- Unterstützung bei der **Product Planung** und der **Organisation** des **Product Backlogs** zur **Maximierung** des Nutzens
- **Vorbereitung und Moderation** von Scrum Events je nach Bedarf

### Service für das Entwicklungsteam

- **Coaching** in den Bereichen „selbst-organisation“ und „cross-funktionalität“
- **Coaching** des Entwicklungsteams hinsichtlich der Vorgehensweisen
- **Unterstützung** bei der Erzeugung von Produkten mit hohem Wert
- Beseitigung von „**Impediments**“
- **Vorbereitung und Moderation** von **Scrum Events** je nach Bedarf

### Service für das Unternehmen

- **Beratung, Planung** und ggfs. **Leitung** von Scrum Einführungen
- **Unterstützung** für **Mitarbeiter** und Stakeholder beim **Verständnis** von **Scrum Prozessen** und emp. Produktentwicklung
- Bewirkung von Veränderungen mit positivem Einfluss auf die Team Produktivität
- Zusammenarbeit mit anderen Scrum Mastern um die Effektivität von Scrum Prozessen im Unternehmen zu stärken

# SCRUM TEAM / SCRUM ROLLEN DER SCRUM MASTER

WIE UNTERSTÜTZT DER  
SCRUM MASTER  
  
DAS DEVELOPMENT  
TEAM?

Coacht die **Selbstorganisation** und **Cross-Funktionalität**

Hilft dabei, **hochwertige Produkte** zu entwickeln

Räumt **Hindernisse (Impediments)** aus dem Weg

Implementiert die **Scrum Events**

# SCRUM TEAM / SCRUM ROLLEN DER SCRUM MASTER

WIE UNTERSTÜTZT DER  
SCRUM MASTER  
  
DIE  
ORGANISATION?

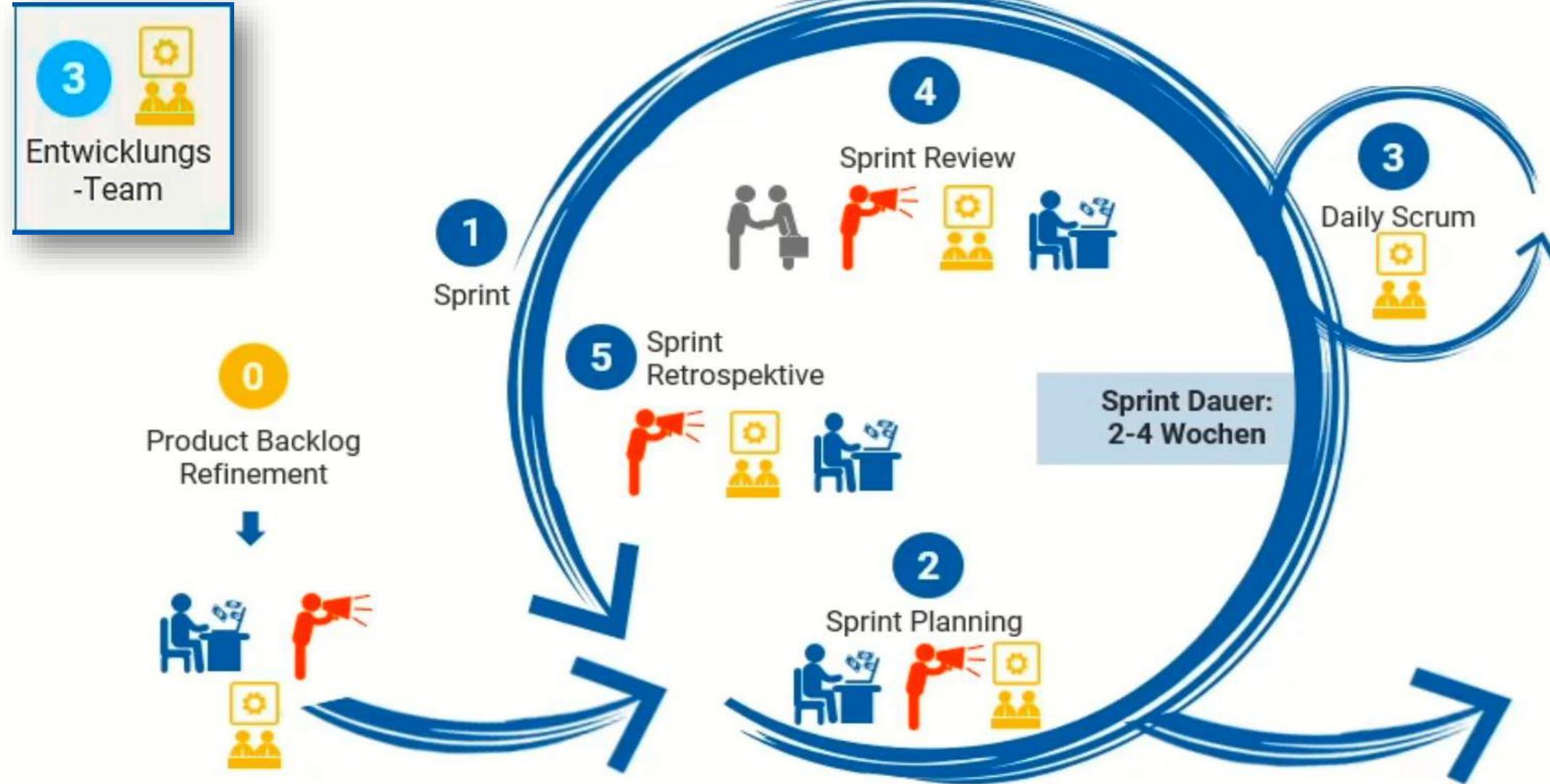
Coacht die Organisation

Zusammenarbeit mit  
anderen Scrum Mastern

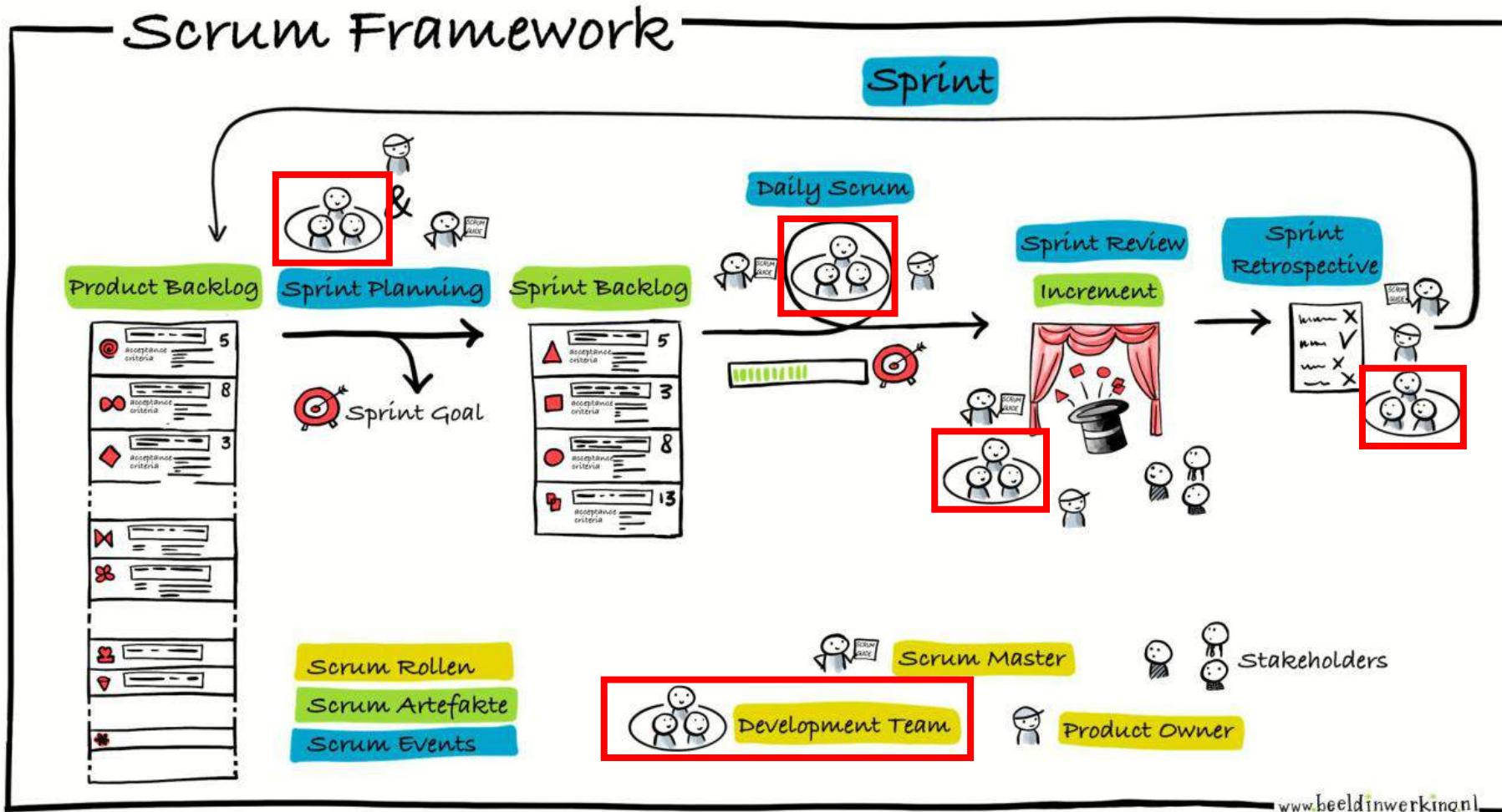
Implementiert Veränderungen zur Erhöhung der  
Produktivität  
im Scrum Team

Coacht Scrum und  
empirische Produktentwicklung

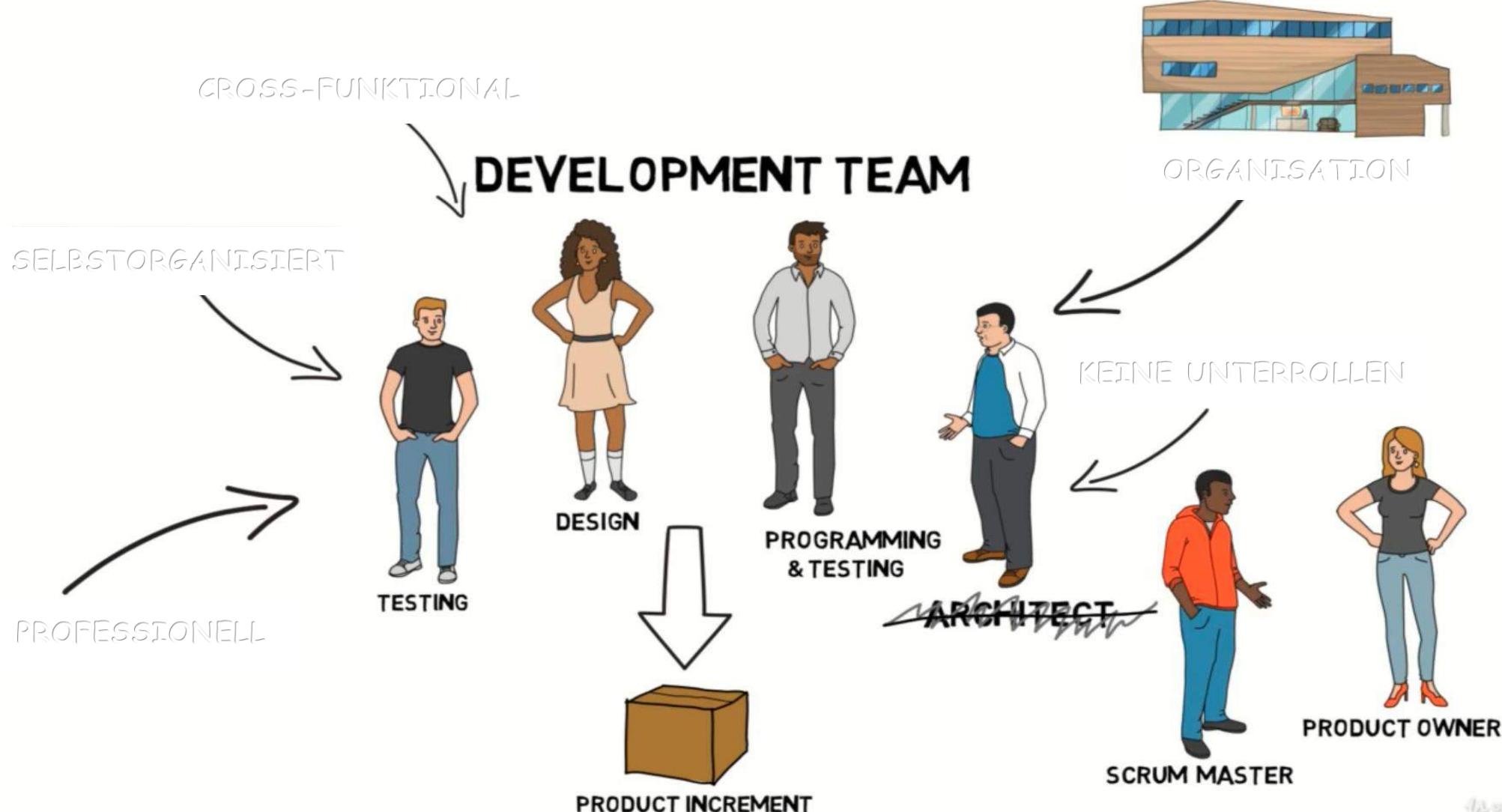
# DAS DEVELOPMENT TEAM



# SCRUM TEAM / SCRUM ROLLEN DAS DEVELOPMENT TEAM

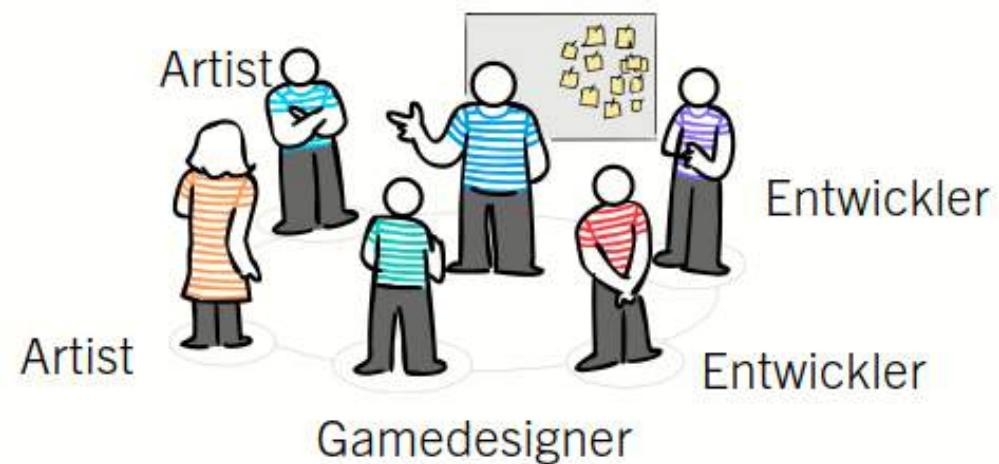


# SCRUM TEAM / SCRUM ROLLEN DAS DEVELOPMENT TEAM



# DAS DEVELOPMENT TEAM

- Entscheiden das „Wie“
- Crossfunktional
- Trägt Verantwortung
- Hat dafür die Befugnisse
- Selbstorganisation
- Innerhalb des Teams wird auf Augenhöhe miteinander umgegangen



Beispiel: Ein crossfunktionales  
Teams zum Erstellen eines Games

# ROLLEN UND VERANTWORTUNGEN IN SCRUM

## DEVELOPMENT TEAM

### Entwicklungsteam



- Besteht i.d.R. aus **3-9 Personen**
- **Cross-funktional** aufgebaut (Entwickler, Tester, Business Analysten, etc. in einem Team)
- **Keine Titel und keine Unterteams!**
- Verantwortlich für die **technische Umsetzung** der Anforderungen
- **Selbst-organisiert**, team-orientiert und **eigenverantwortlich** dafür, wie Sprintziele erreicht werden
- Dem **Sprintziel** verpflichtet!

# DEVELOPMENT TEAM

## Entwicklungsteam

Das Entwicklungsteam ist für die Umsetzung der Anforderungen zuständig

### Anforderungen

- Cross-funktional (bereichsübergreifend) und bestehend aus **3-9 Personen**
- Muss in der Lage sein **alle anfallenden** Entwicklungstätigkeiten abzuwickeln inkl. **Test**
- **Vollzeit verfügbar**
- **Intrinsisch motiviert und eigenständig**
- **Agiles Mindset**
- **Gutes Zeitmanagement und Bereitschaft** Verantwortung für zusagte Inhalte zu übernehmen

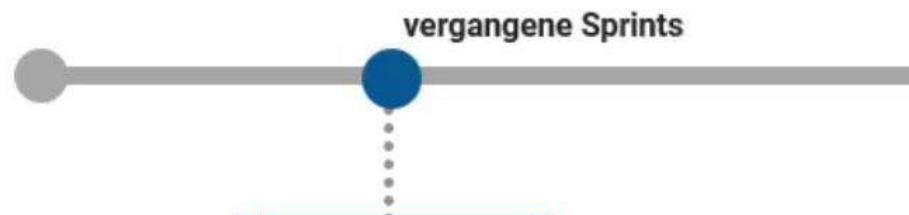
### Verantwortung

- Trotz **spezialisierter Skills** zählt nur die Gesamtverantwortung des Entwicklungsteams!
- **Liefert** das jeweilige **Produktinkrement** zum Sprintende
- **Selbstständige** Aufgabeneinteilung (verantw. **Sprint Backlog**) innerhalb des Sprints mit Fokus auf **Sprintziel**
- **Kompetenzübergreifendes** Engagement
- Ausführlicher **Test** und **Qualitätssicherung**
- Identifikation und Kommunikation von **Impediments**
- **Offene Kommunikation** bei Unklarheiten zu User Stories und Sprint Planung

### Aufgaben

- Schätzung von **User Stories**
- **Planung, Design, Entwicklung** und **Test** von technischen Lösungen
- **Aktualisierung** von Kanban, oder Burndown Charts
- **Dokumentation** von Ergebnissen, oder Code
- Teilnahme an Scrum Aktivitäten (**Backlog Refinement, Planning, Daily, Review, Retrospektive**)
- Einbindung des **Product Owners** für **Abnahmen** vor Sprintende
- Vorstellung des **Sprintergebnisses** im Sprint Review
- Erstellung einer **Definition of Done**

# VELOCITY UND KAPAZITÄT



## Velocity

Die „Velocity“ blickt auf die letzten 3 Sprints zurück und ermittelt den laufenden Durchschnitt der erfolgreich umgesetzten Story Points

**(Achtung: Messung in Story Points!)**



## Kapazität

Die Kapazität blickt in die Zukunft und ermittelt, die verfügbare Arbeitszeit des Entwicklungsteams

**(Achtung: Messung in Arbeitsstunden!)**

# KAPAZITÄT

## ERMITTlung DER STANDAR-KAPAZITÄT

# Kapazität

Die Kapazität dient zur Planung des nächsten Sprints

### Kapazität (Standard Kapazität)

- Unter „Kapazität“ versteht man die tatsächlich zur Verfügung stehende Zeit, die für die Umsetzung von User Stories genutzt werden kann
- Bei der Berechnung der Kapazität sollte Aufwand für Nicht-Entwicklungstätigkeiten (Scrum Events, E-Mail Bearbeitung, ggf. technische Schulden abgezogen werden)

### Beispiel:

Entwicklungsteam aus 6 Mitarbeitern

Sprintdauer: 2 Wochen (10 Tage)

Arbeitstag: 8 Stunden, abzgl. 25% für Meetings und sonstige Tätigkeiten = 6 Std.

$$\text{Anzahl Entwickler} \times \text{Sprinttage} \times \text{Anzahl Arbeitsstunden} = \text{Standard Kapazität}$$
$$6 \times 10 = 60 \text{ (Tage)}$$
$$8 \times 0,75 \text{ (Arbeitsstunden - 25\%)} = 360 \text{ (Std)}$$

Dem Entwicklungsteam stehen 360 Stunden zur Verfügung

# STANDARD KAPAZITÄT

**Beispiel:**

**Entwicklungsteam aus 6 Mitarbeitern**

**Sprintdauer: 2 Wochen (10 Tage)**

**Arbeitstag: 8 Stunden, abzgl. 25% für Meetings und sonstige Tätigkeiten = 6 Std.**

## **Zeitansatz Events**

(2 Wochen Sprint)

Refinement: 2-6 Std.

Sprint Planning: 4 Std.

Sprint Review: 2 Std.

Sprint Retrospektive: ca. 2 Std.

**Summe: 10 - 14 Std. pro Entwickler!**  
**(Schwankungen je Team möglich)**

# SPRINT KAPAZITÄT

## Beispiel:

Im kommenden Sprint fällt ein Mitarbeiter aufgrund von Urlaub für 2 Wochen aus (-10 Tage)  
Zusätzlich wird das restliche Team (5 Mitarbeiter) einen Tag lang auf Schulung sein (-5 Tage)  
Dem Entwicklungsteam fehlen somit im nächsten Sprint 15 Tage ggü. der Standardkapazität

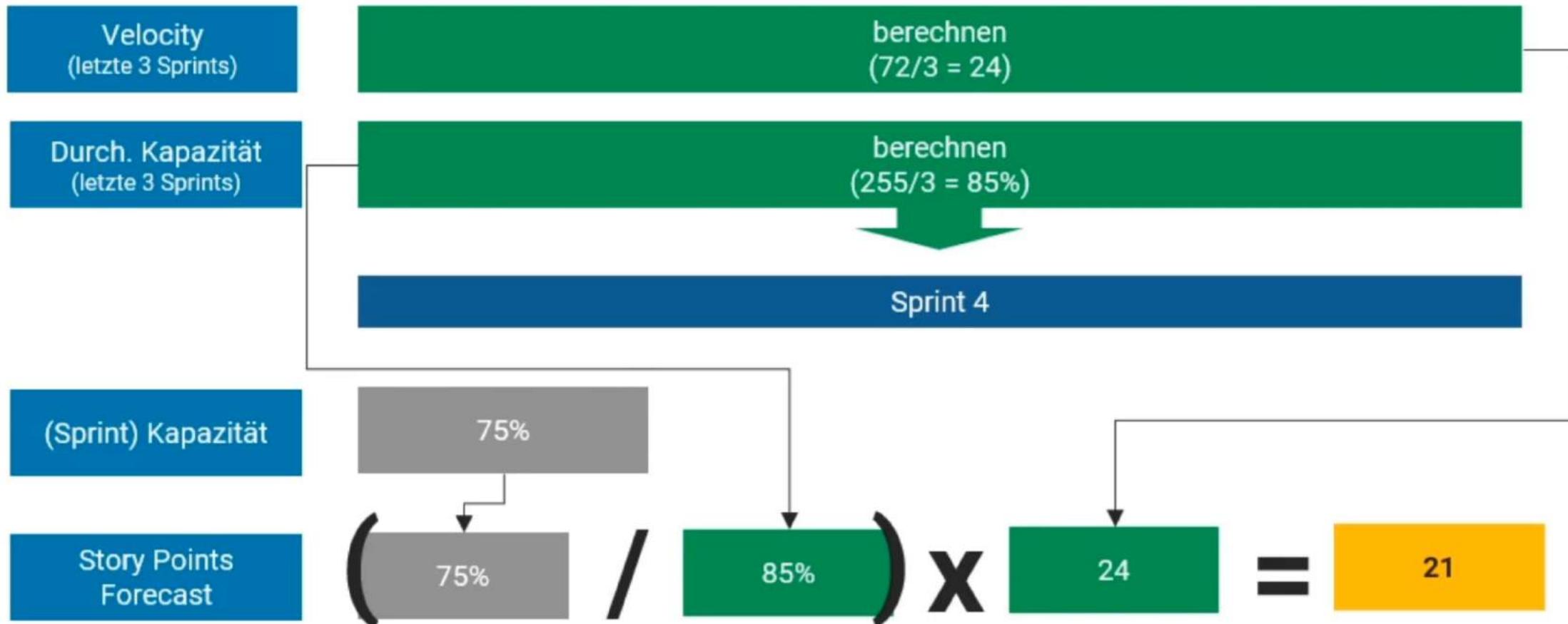


Das Entwicklerteam hat ca. 75% der Standard-Kapazität  
im nächsten Sprint verfügbar

# PROGNOSEN MIT VELOCITY UND KAPAZITÄT

	Sprint 1	Sprint 2	Sprint 3
Sprint-Kapazität	75%	85%	95%
Story Points erledigt	Zählen (25)	Zählen (20)	Zählen (27)
Velocity (Durchschnitt letzte 3 Sprints)		Berechnen $(72/3 = 24)$	
Durchsch. Kapazität (letzte 3 Sprints)		Berechnen $(255/3 = 85\%)$	

# PROGNOSEN MIT VELOCITY UND KAPAZITÄT



Das Entwicklerteam wird vermutlich ca. 21 Story Points im  
nächsten Sprint umsetzen können

# VELOCITY / KAPAZITÄT ZUSAMMENFASSUNG

## Velocity

Die „Velocity“ blickt auf die **letzten 3 Sprints zurück** und ermittelt den laufenden **Durchschnitt** der erfolgreich umgesetzten **Story Points** (Messung in relativen Story Points!)

## Story Point Prognose

### Annahmen:

- Entwicklungsteam aus **6 Mitarbeitern**
- Sprintdauer **2 Wochen (10 Tage)**
- Arbeitstag **8 Std. abzgl. 25% für übergreifende Tätigkeiten**
- Velocity: **24**
- Durchschnittliche Kapazität: **85%**

### Schritt 1: Standard Kapazität berechnen

- 6 (MA) x 10 (Tage) x 6 (8 Arbeitsstunden – 25%) = 360 Std.**

### Schritt 2: Sprint Kapazität berechnen

#### Annahmen:

- 1 MA fällt komplett aus (**-10 Tage**)
- Restliches Team einen Tag auf Schulung (**-5 Tage**)
- 60 (verf. Tage) – 15 (Abwesenheiten) \* 6 (8 Arbeitsstunden – 25%) = 270 Std.**
- 270 (verf. Std.) / 360 (Standard) = 75%**

### Schritt 3: Story Point Prognose berechnen

- 75% (Sprint Kapazität) / 85% (durch. Kapazität) \* 24 (Velocity) = 21**
- Das Entwicklungsteam wird User Stories im Wert von ca. **21 Story Points** umsetzen können!

## Kapazität

Die Kapazität blickt in die **Zukunft** und ermittelt die verfügbare Arbeitszeit für die Umsetzung von User Stories (**Messung in absoluten Stunden**)

Vorsicht!

Die Velocity wird in Story Points berechnet

Die Kapazität wird in Stunden, oder Arbeitstagen berechnet

## Kanban Boards

Kanban Boards dienen der Nachverfolgung innerhalb des Sprints und zur Überwachung des Fertigstellungsgrades einzelner Tasks und User Stories

	To Do	In Arbeit	Fertig
User Story			

Nutze gerne das Excel-Template zur Berechnung der Story Point Prognosen und Team-Kapazität



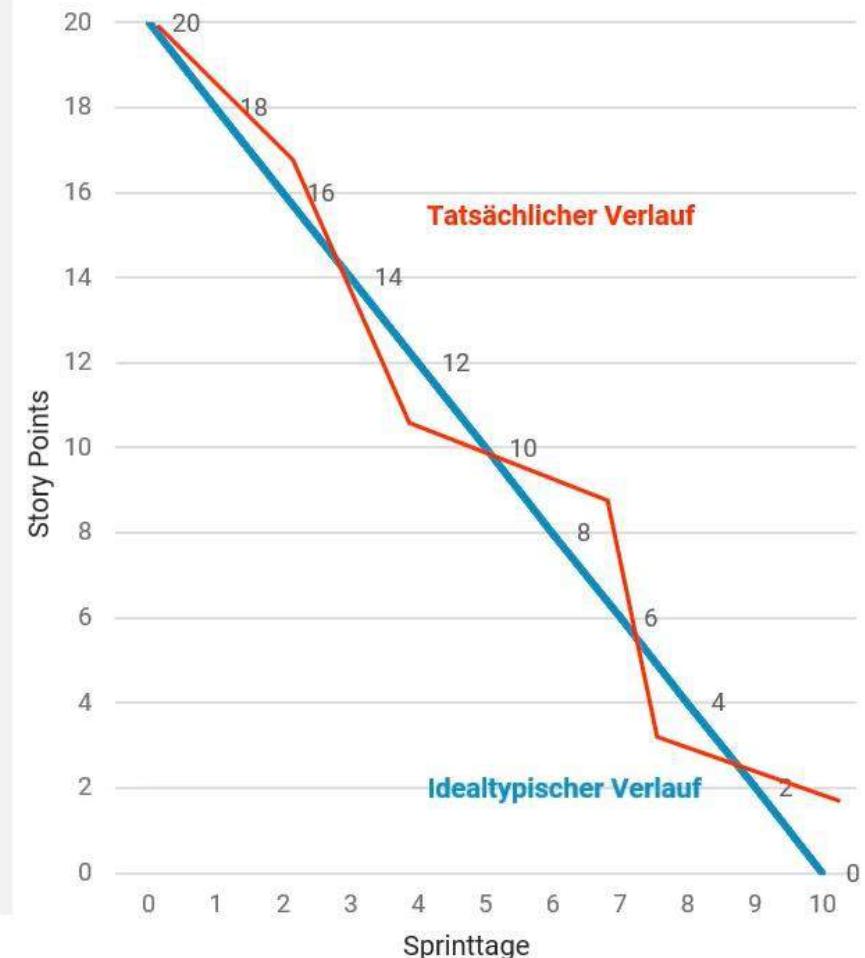
# BURNDOWN CHARTS

## Burndown-Charts

Burndown Charts sind ein gutes Instrument um innerhalb agiler Projekte den Fortschritt zu messen, bzw. den Anteil noch offener Arbeiten bis zum Sprintende, oder einem Release aufzuzeigen

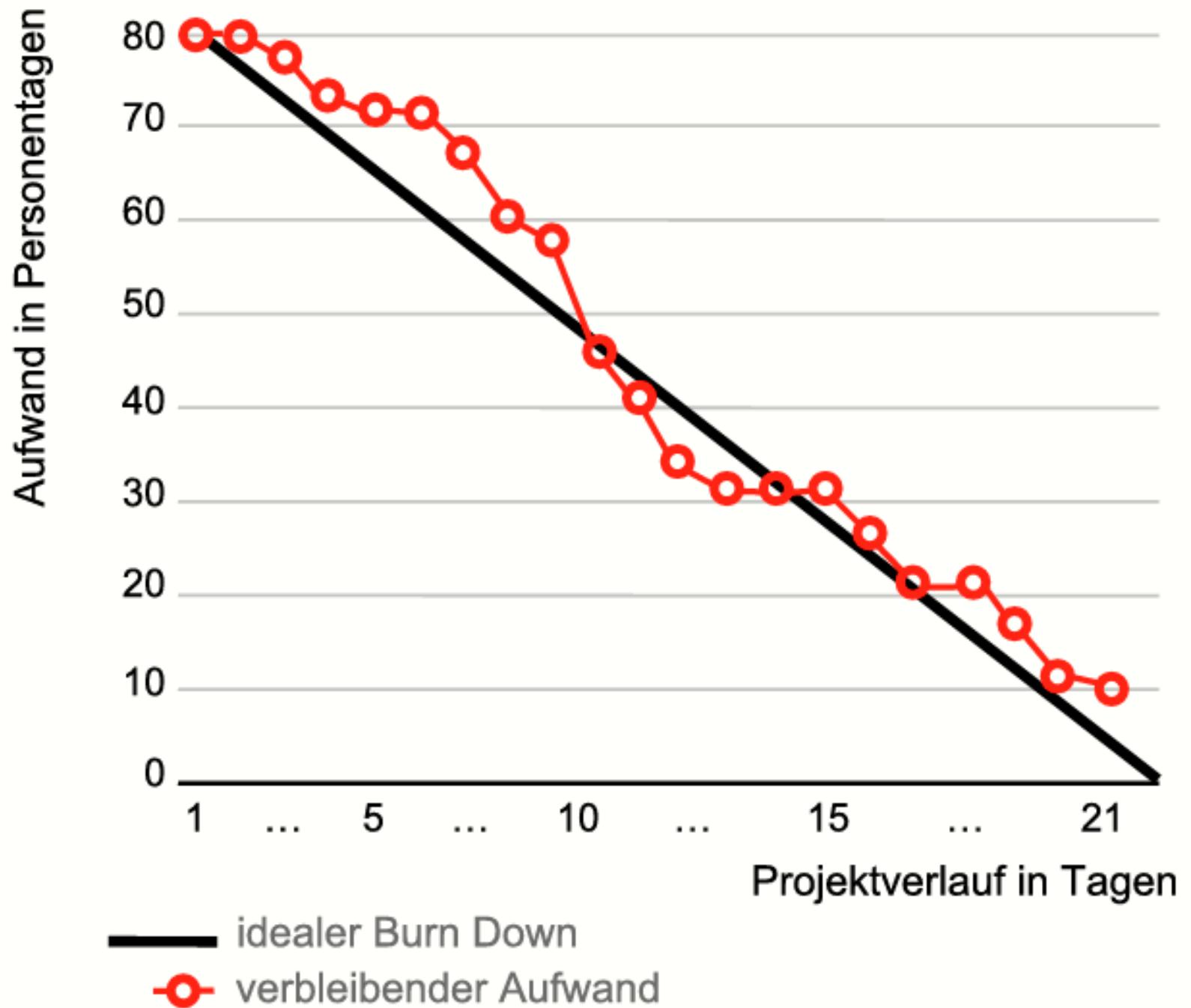
### Beschreibung

- Die Y-Achse zeigt die Anzahl der (verbleibenden) Story Points für den aktuellen Sprint
- Die X-Achse zeigt die Anzahl der Sprinttage an
- Der **idealtypische Verlauf** ist eine lineare Gerade, welche die Gesamtzahl der geplanten Story Points (20) durch die Anzahl der verfügbaren Sprinttage (10) teilt. Zum Sprintende sollte die Zahl der verbleibenden Story Points bei 0 liegen
- Der **tatsächliche Verlauf** ergibt sich aus täglichen Aktualisierungen des Sprint-Backlogs und zeigt den Fortschritt den das Entwicklungsteam bei der Abarbeitung von User Stories macht
- Viele Tools verfügen über eine automatische Berechnung, andernfalls reicht hierfür auch Excel aus
- Burndown-Charts sollten täglich aktualisiert werden und können Probleme sichtbar machen
- Häufig zeigen Burndown Charts in den ersten Tagen wenig Fortschritt, da nur abgeschlossene User Stories berücksichtigt werden sollten

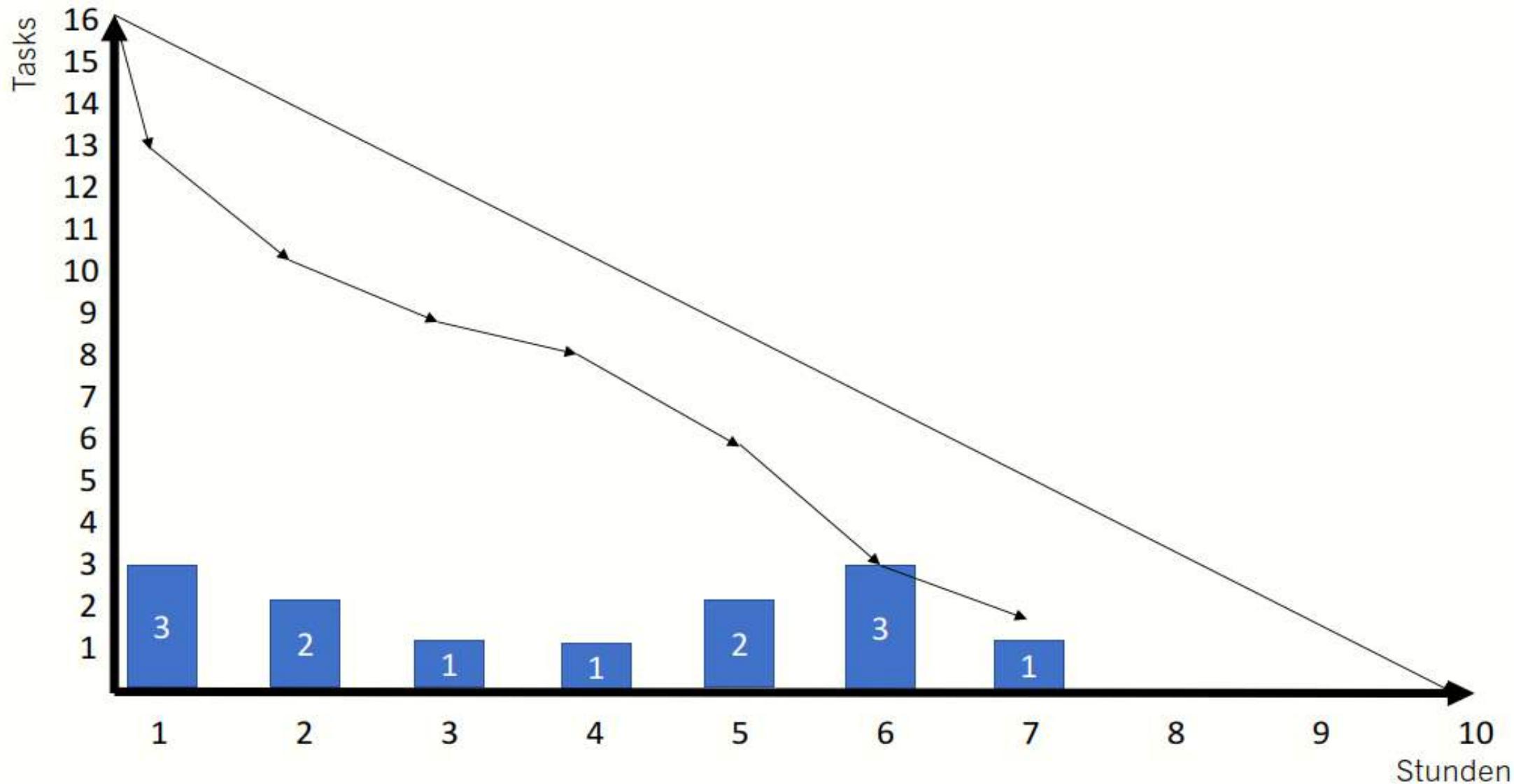


# BURNDOWN CHARTS

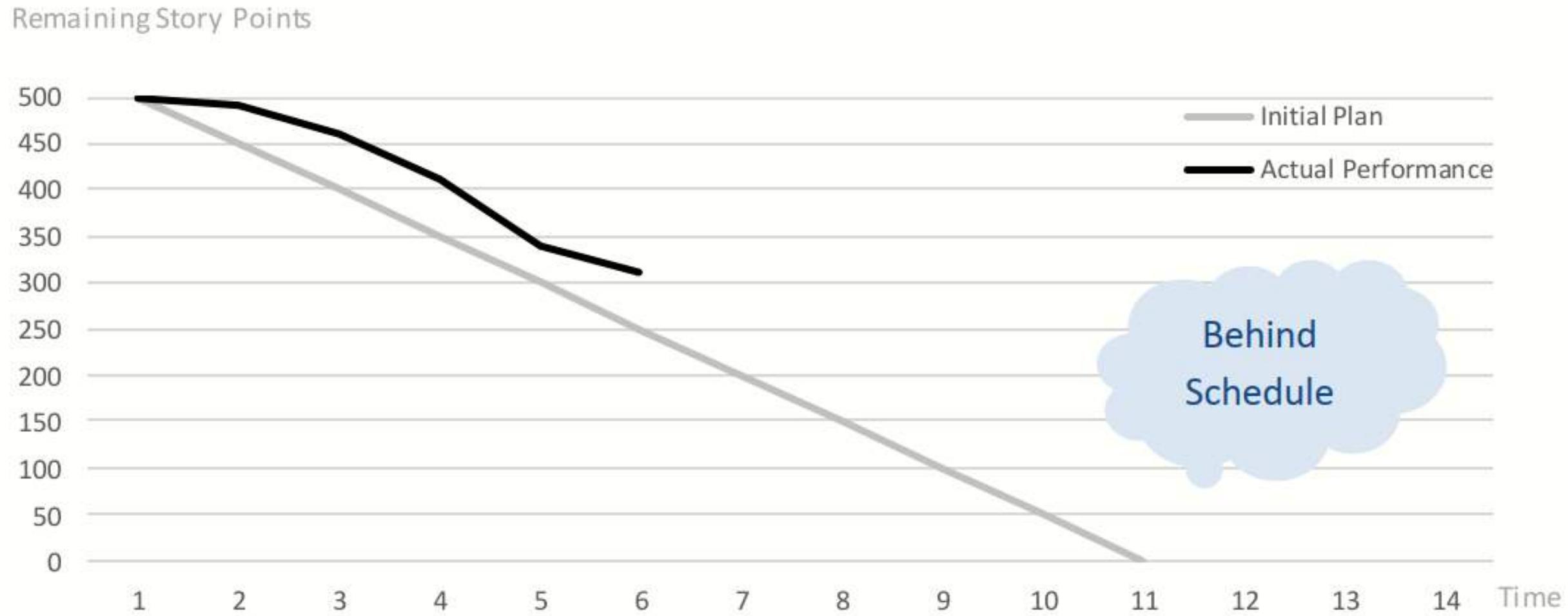
## IDEALER BURNDOWN



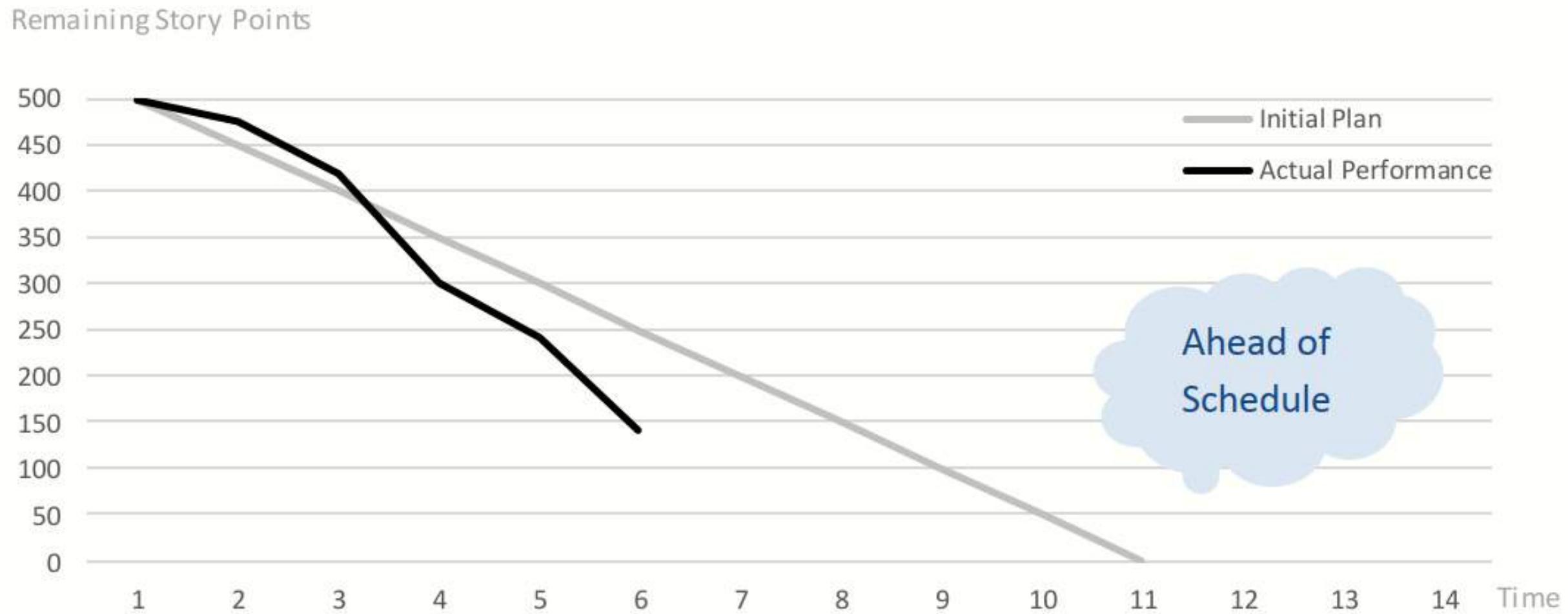
# BURNDOWN CHARTS



# PROJEKT BURNDOWN CHARTS HINTER DEM ZEITPLAN

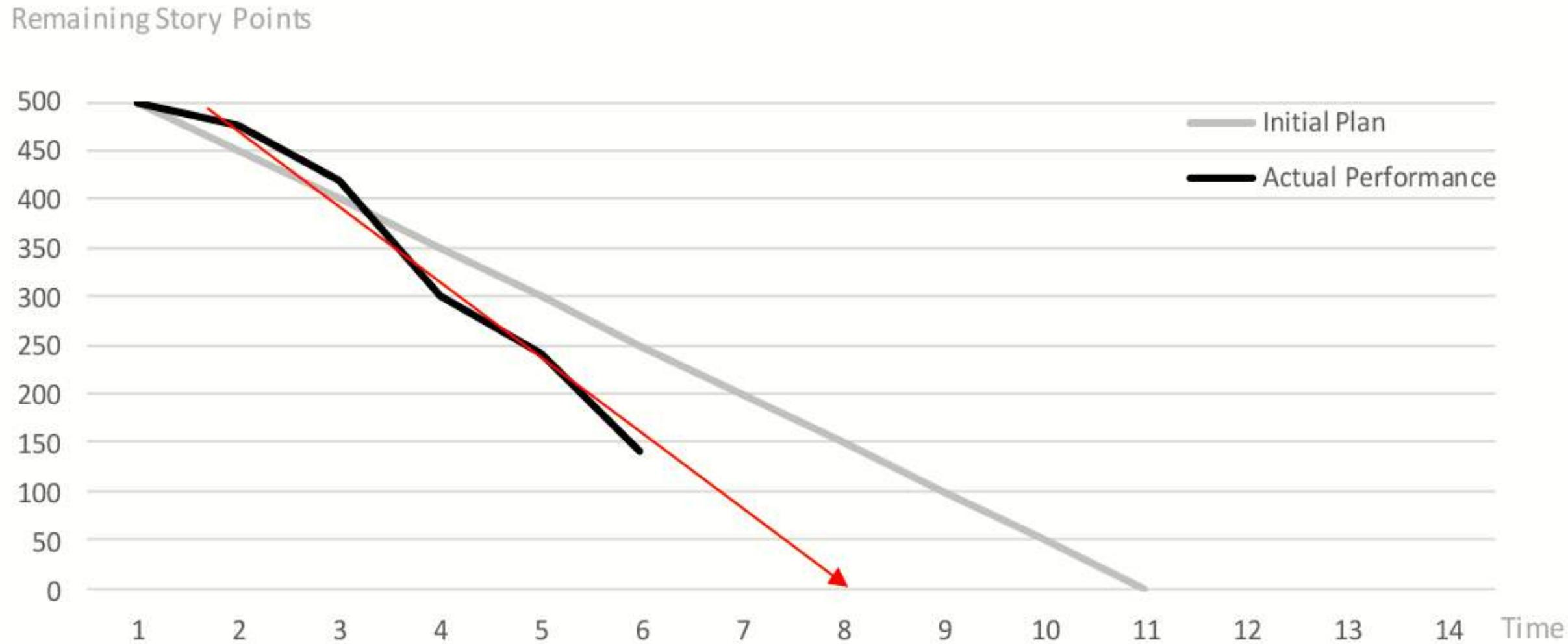


# PROJEKT BURNDOWN CHARTS VOR DEM ZEITPLAN



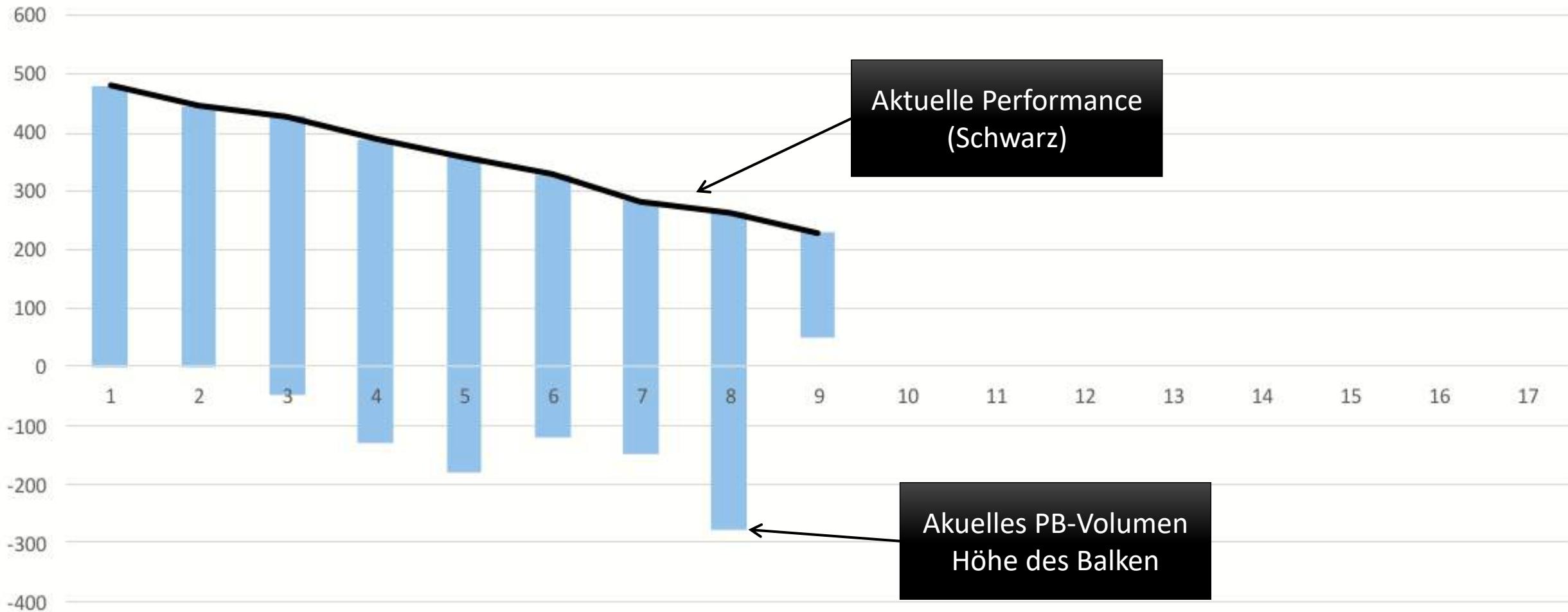
# PROJEKT BURNDOWN CHARTS

## NEUEINSCHÄTZUNG DER PROJEKTDAUER



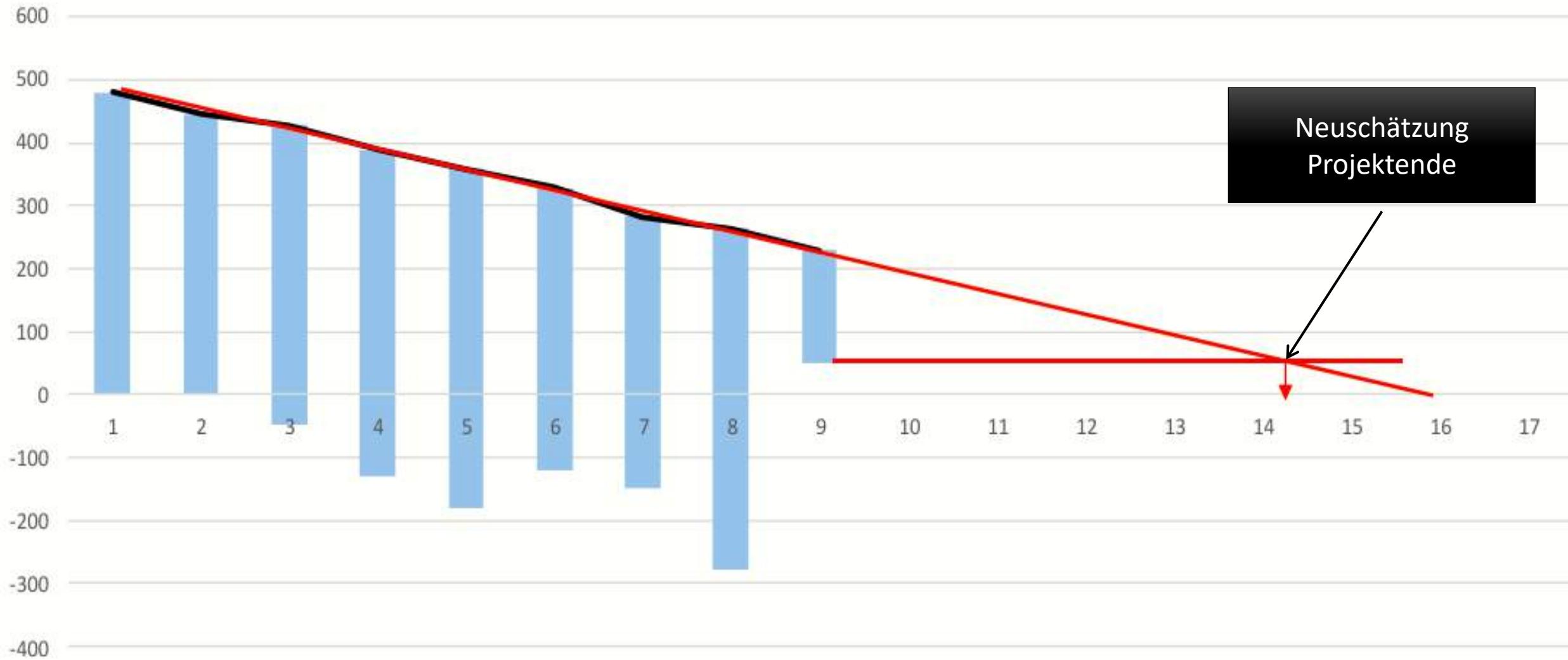
# BURNDOWN BARS

## ÄNDERUNGEN IM PRODUCT BACKLOG MIT BERÜCKSICHTIGEN

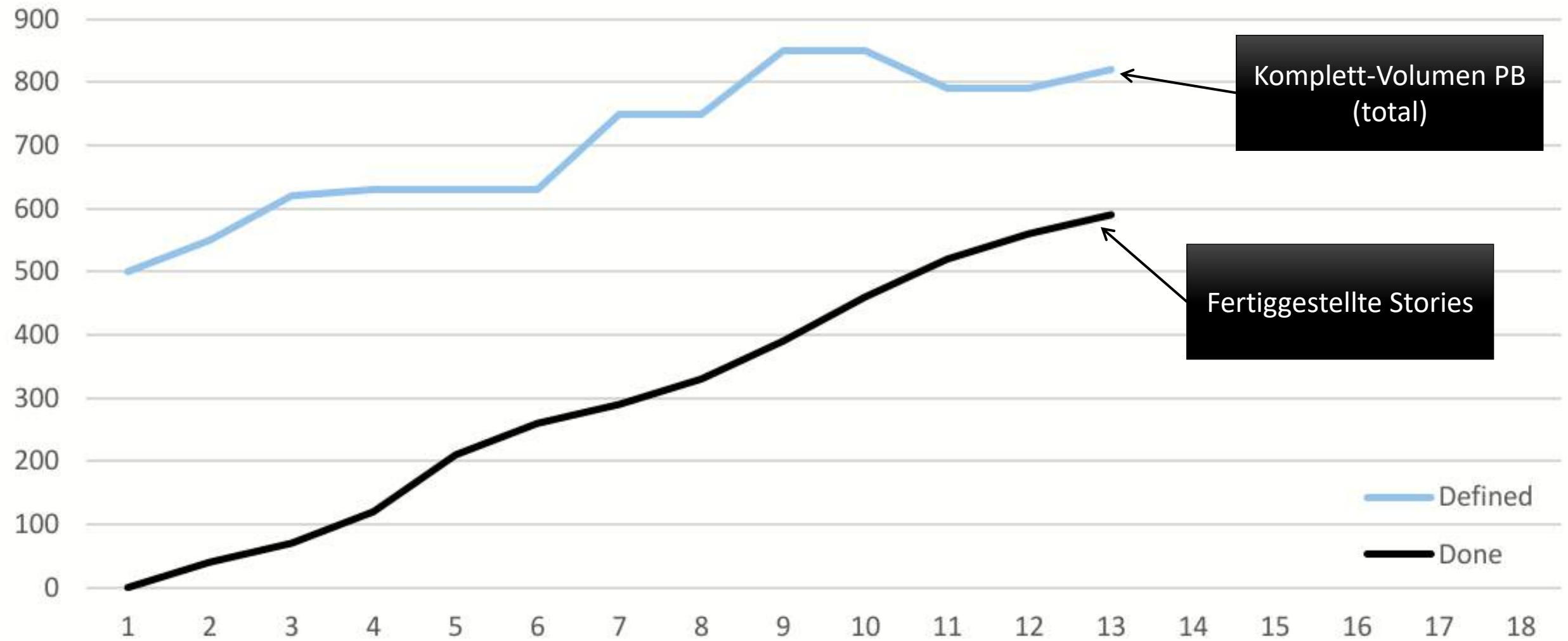


# BURNDOWN BARS

## ÄNDERUNGEN IM PRODUCT BACKLOG MIT BERÜCKSICHTIGEN

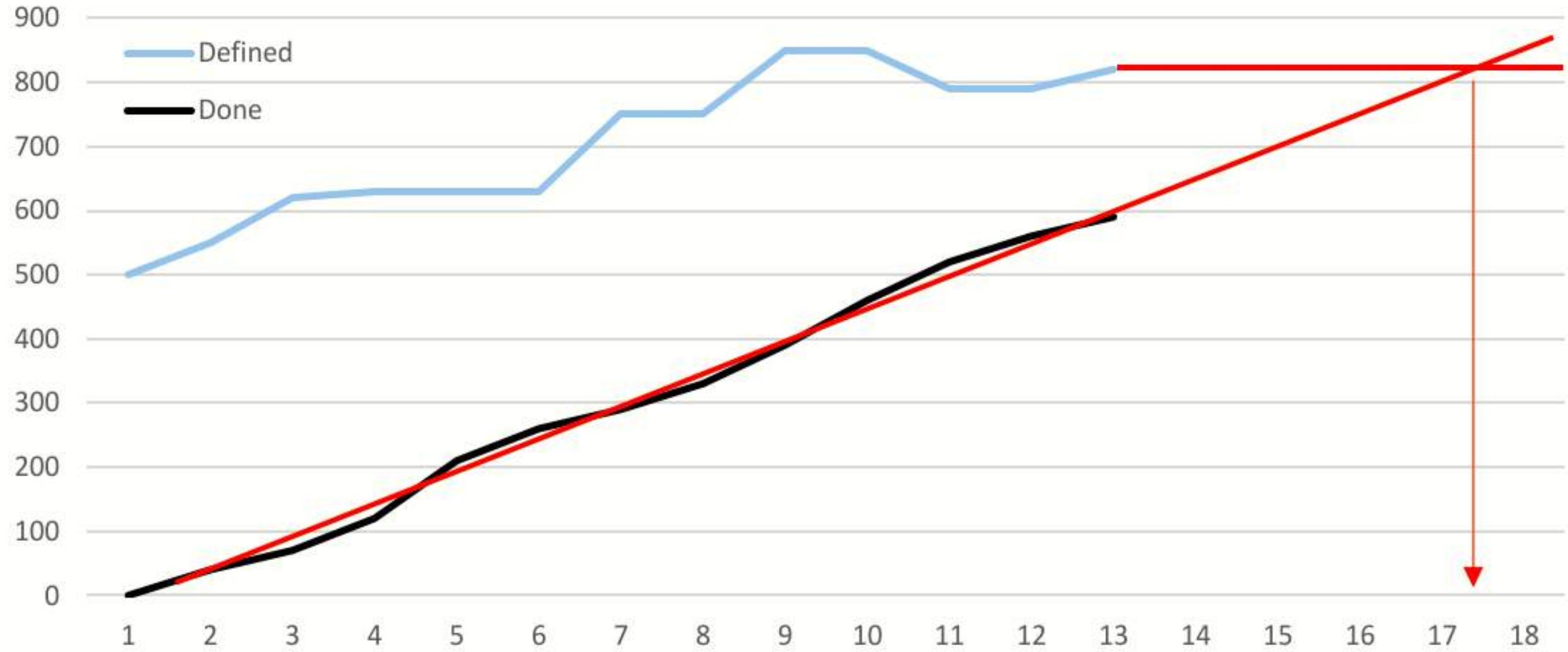


# BURN-UP CHARTS

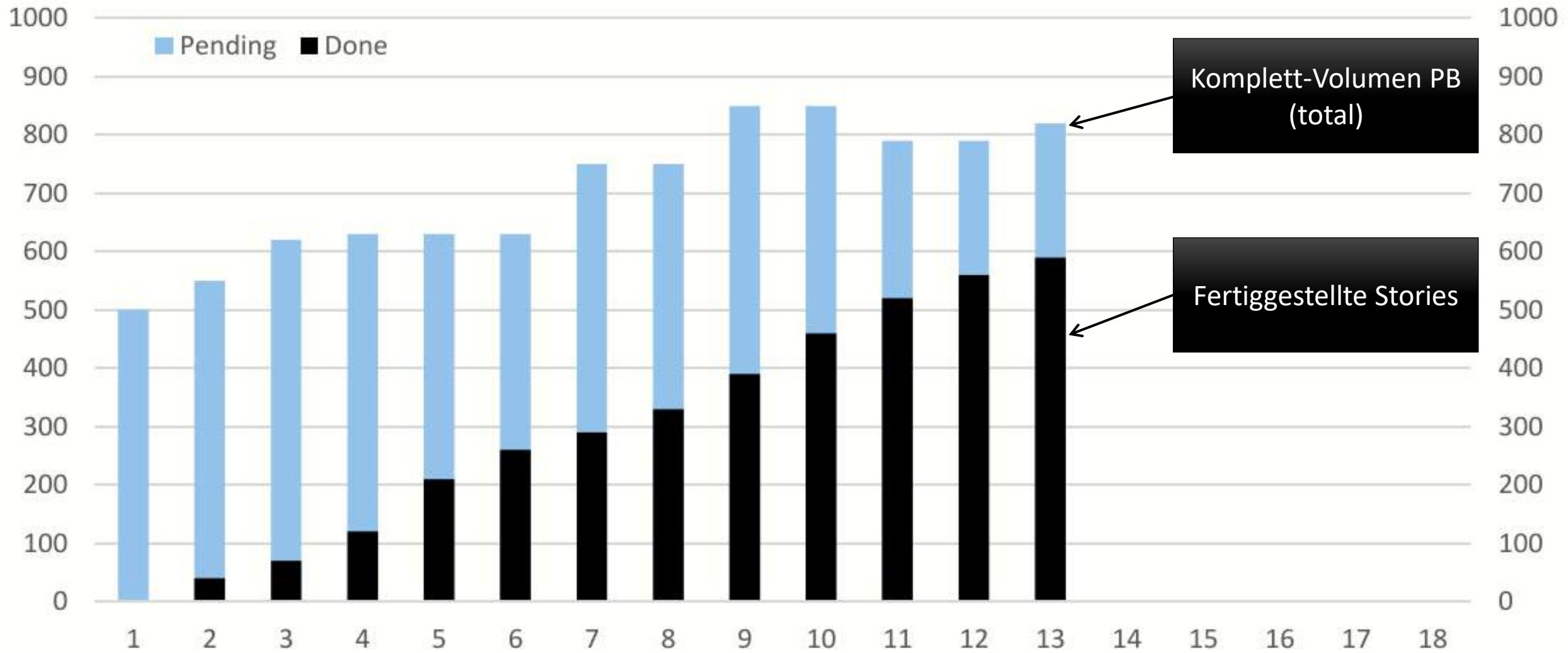


# BURN-UP CHARTS

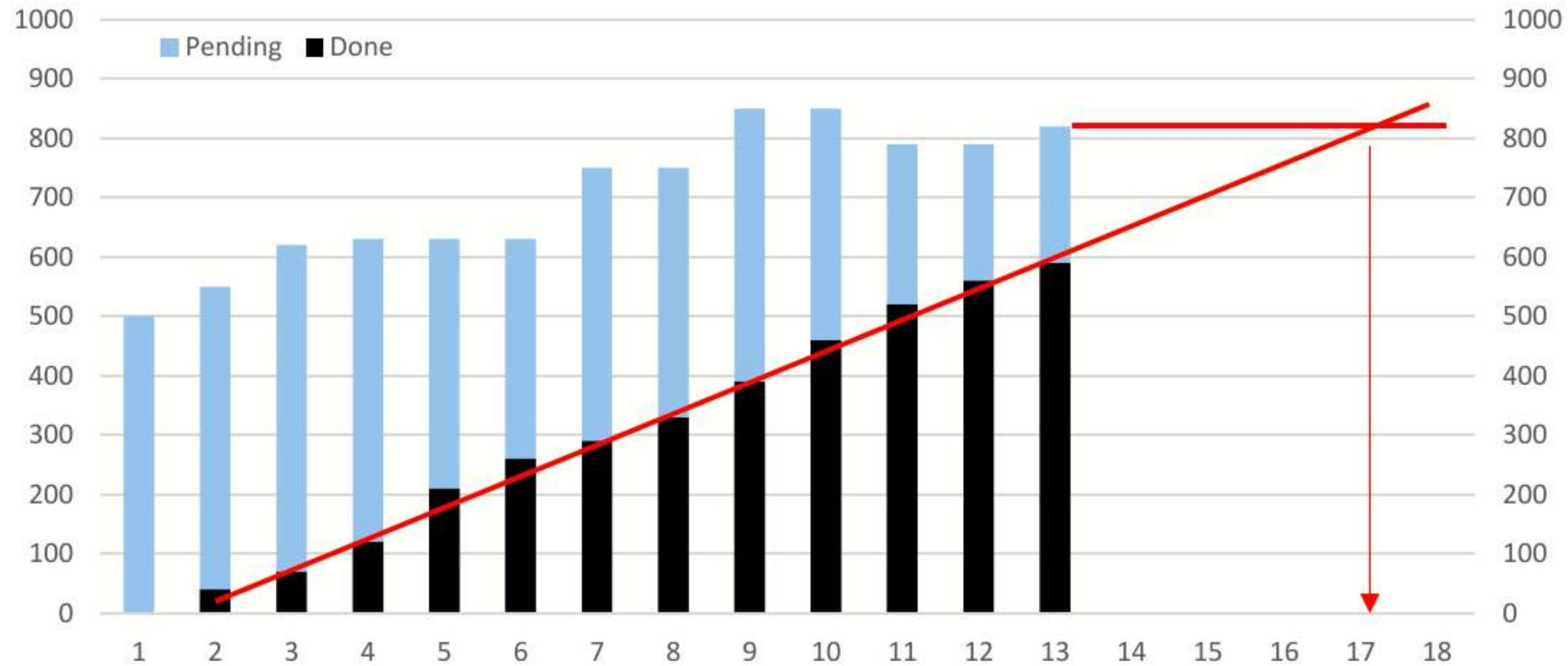
## BERECHNUNG PROJEKTENDE BEI GLEICHBLEIBENDEM PB-VOLUMEN



# CUMULATIVE FLOW DIAGRAMME

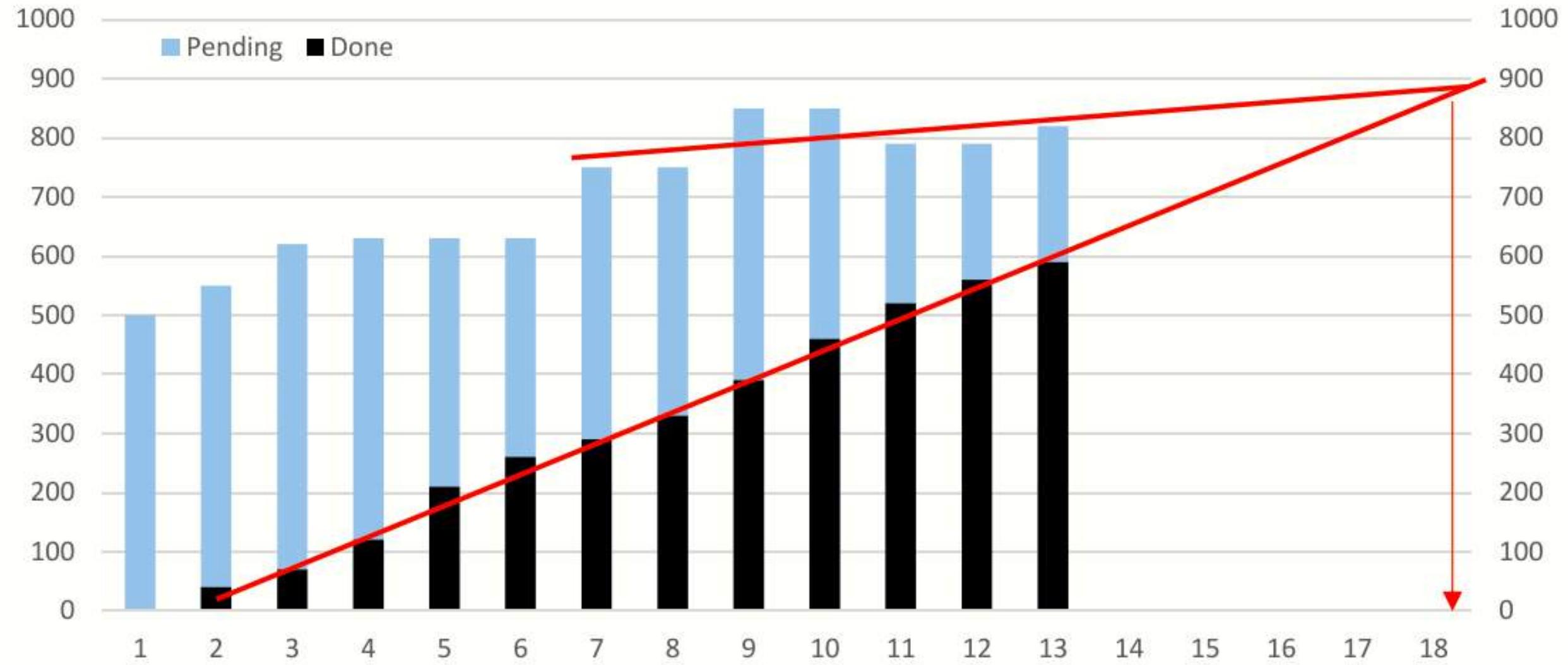


# CUMULATIVE FLOW DIAGRAMME BERECHNUNG PROJEKTENDE



# CUMULATIVE FLOW DIAGRAMME

## BERECHNUNG PROJEKTENDE UNTER BERÜCKSICHTIGUNG ZU ERWARTENDER PB-ÄNDERUNGEN



# BURNDOWN CHARTS

## VORTEILE

- Den Stand der Arbeit für den aktuellen Tag
- Die Historie der vergangenen Tage
- Die verbleibende Arbeit bis zum Sprintende
- Die Aussage, ob das Team die Arbeit pünktlich abschließt

# BURNDOWN CHARTS

## Beispiel:



Im Beispiel oben könnte das wie folgt aussehen: Am ersten Projekttag gehen Teammitglieder zum Task Board und nehmen sich Arbeitspakete (hängen die entsprechenden Karten nach „In Work“). Am dritten Projekttag ist ein Teammitglied mit einem Arbeitspaket fertig (es hängt die entsprechende Karte auf dem Task Board nach „Done“), für das ein Aufwand von 2 Tagen geschätzt war. Dies wird nun im Diagramm eingetragen und die rote Kurve fällt von 80 auf 78 Personentage (PT). Am vierten Tag ist ein anderer mit einem Arbeitspaket fertig, für das 6 PT geschätzt waren. Die Kurve des verbleibenden Aufwands fällt auf 72 PT. Wenn die gesamte Projektdauer 21 Tagen umfasst, müsste die Kurve idealerweise nach dieser Zeit bei 0 PT ankommen.

# MoSCoW-PRIORISIERUNG

## MUST OR SHOULD – COULD OR WON'T

**M**

### MoSCoW Must!

Anforderungen die unbedingt umgesetzt werden müssen!

Anforderungen sind bspw. rechtlich erforderlich, haben einen sehr hohen Nutzen, oder kritische Funktionen

**S**

### MoSCoW Should!

Anforderungen die umgesetzt werden sollten!

Anforderungen haben einen hohen Nutzen und versprechen bspw. Wettbewerbsvorteile

**C**

### MoSCoW Could!

Anforderungen die umgesetzt werden können, wenn "Must" und "Should" implementiert sind

Anforderungen versprechen einen Nutzen, der den zusätzlichen Aufwand noch rechtfertigt

**W**

### MoSCoW Won't

Anforderungen, die vermutlich nie umgesetzt werden

Anforderungen versprechen einen zu geringeren Nutzen um zusätzlichen Aufwand zu rechtfertigen

# KOSTEN – NUTZEN MATRIX

## Kosten-Nutzen Matrix

Eine pragmatische Einschätzung von User Stories auf Kosten-Nutzen Basis hilft bei der Priorisierung enorm

### Kosten-Nutzen Matrix

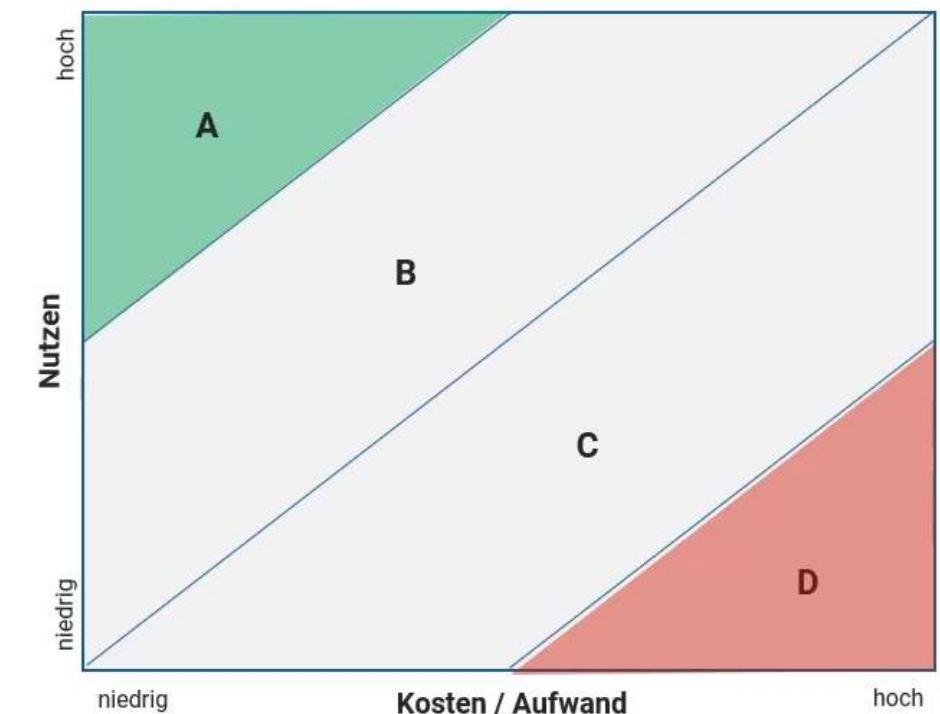
Für die Kosten-Nutzen Matrix müssen die **Kosten** und der **Nutzen** jeder einzelnen Anforderung ermittelt werden

Üblicherweise wird für die **Kosten**, der **Arbeitsaufwand** herangezogen

Hierfür können unterschiedliche Ansätze gewählt werden

- **Monetärer** Ansatz:  
konkrete Ermittlung von **Kosten** und wirtschaftlichem Nutzen in EUR
- **Abstrakter** Ansatz  
Ermittlung von Kosten und Nutzen gemäß individueller Kriterien und Skalen (Story Points)

Bereits eine pragmatische Einordnung der User Stories und eine damit verbundene Diskussion kann enorm hilfreich für das Projekt sein



# RISIKO – NUTZEN MATRIX

## Kosten-Nutzen Matrix

Die **Risiko-Nutzen** Matrix folgt demselben Ansatz und erlaubt eine Einordnung von User Stories anhand klarer Kriterien, ersetzt jedoch die Kosten durch das Risiko

Risiken können hierbei unterschiedlich betrachtet werden:

- Risiken hinsichtlich der **Umsetzung** (bspw. Know-How, **technische Abhängigkeiten, mögliche Komplikationen**)
- Risiken hinsichtlich der **Marktakzeptanz** (**will der Kunde diese Funktionalität wirklich haben?**)

Wie auch bei der Kosten-Nutzen Matrix fördert eine Einordnung auch hier die Diskussion und kritische Auseinandersetzung.

## Kosten-Nutzen Matrix

Die **Risiko-Nutzen** Matrix folgt demselben Ansatz und erlaubt eine Einordnung von User Stories anhand klarer Kriterien, ersetzt jedoch die Kosten durch das Risiko

Risiken können hierbei unterschiedlich betrachtet werden:

- Risiken hinsichtlich der **Umsetzung** (bspw. Know-How, **technische Abhängigkeiten, mögliche Komplikationen**)
- Risiken hinsichtlich der **Marktakzeptanz** (**will der Kunde diese Funktionalität wirklich haben?**)

Wie auch bei der Kosten-Nutzen Matrix fordert eine Einordnung auch hier die Diskussion und kritische Auseinandersetzung.



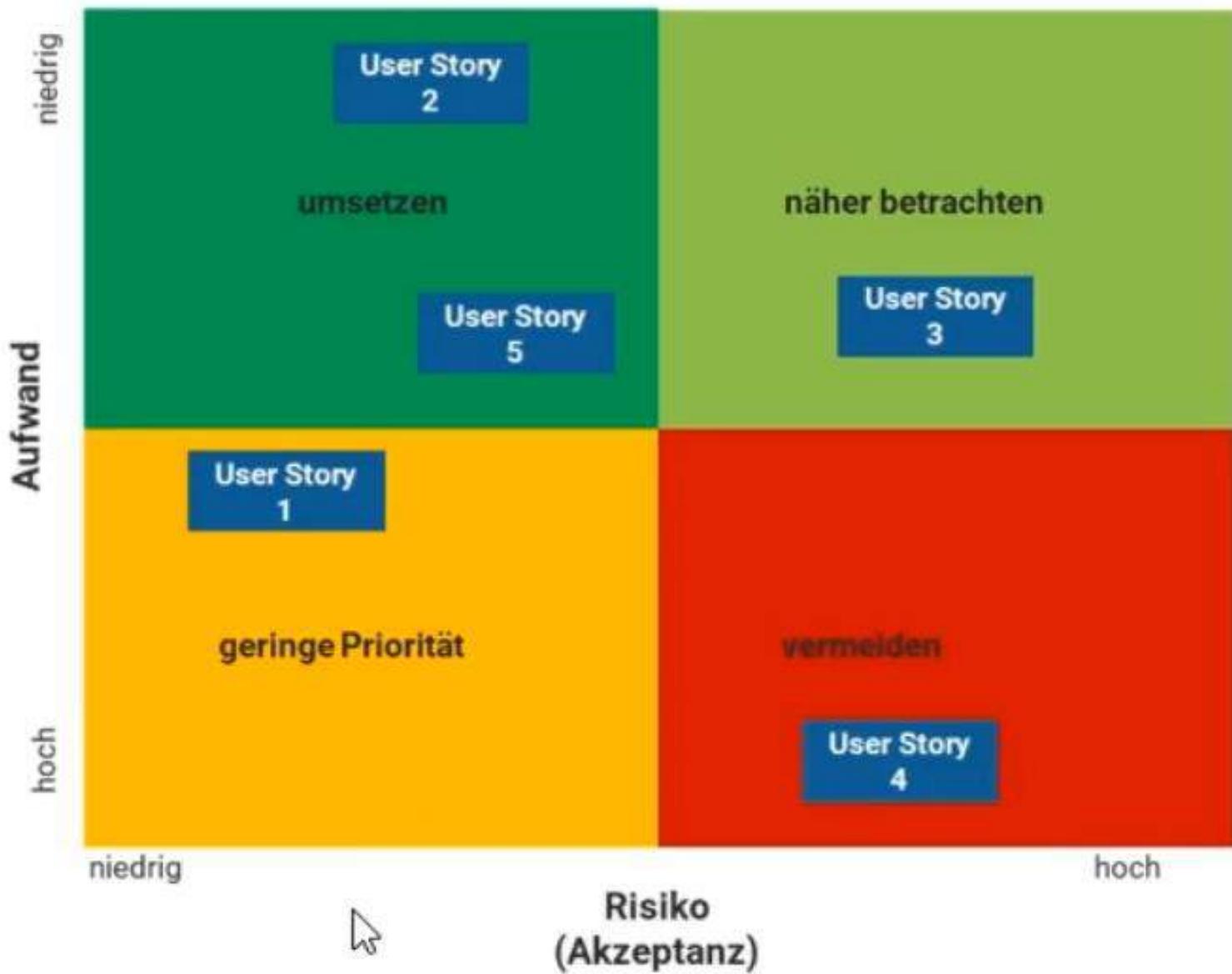
## Kosten-Nutzen Matrix

Die **Risiko-Nutzen** Matrix folgt demselben Ansatz und erlaubt eine Einordnung von User Stories anhand klarer Kriterien, ersetzt jedoch die Kosten durch das Risiko

Risiken können hierbei unterschiedlich betrachtet werden:

- Risiken hinsichtlich der **Umsetzung** (bspw. Know-How, **technische Abhängigkeiten**, mögliche **Komplikationen**)
- Risiken hinsichtlich der **Marktakzeptanz** (**will der Kunde** diese **Funktionalität** wirklich haben?)

Wie auch bei der Kosten-Nutzen Matrix fordert eine Einordnung auch hier die Diskussion und kritische Auseinandersetzung.



# PRIORISIERUNG ZUSAMMENFASSUNG

## Allgemein

- Die Priorisierung von Anforderungen zählt zu den **wichtigsten Erfolgsfaktoren** eines agilen (Scrum) Projekts!
- Die **Schwierigkeit** einer guten Priorisierung mit klarem Fokus und eindeutiger Ablehnung unwichtiger Elemente ist **nicht zu unterschätzen**
- Priorisierungen erfordern eine
  - wirtschaftliche und strategische** Denkweise,
  - sowie **diplomatisches Geschick und Durchsetzungskraft**.

## Backlog

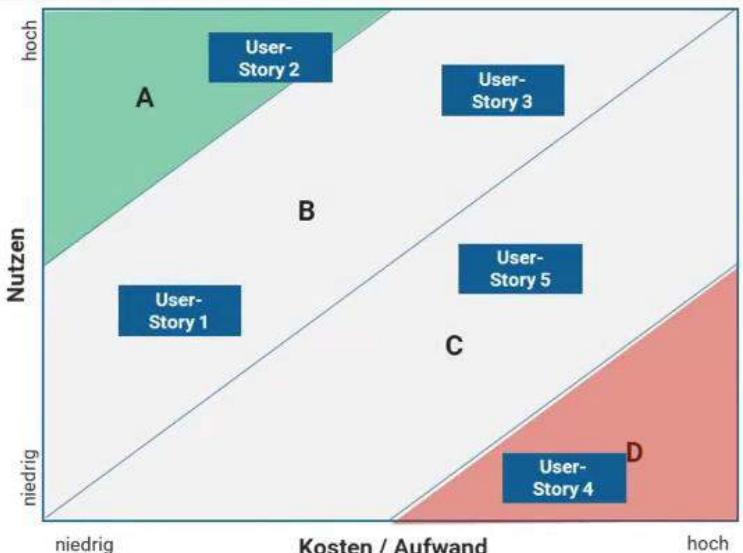
- Die Priorisierung von Anforderungen sollte im Product Backlog stets **ersichtlich** und **aktuell** sein. Dies gilt insbesondere für die **Vorbereitung von Sprint Plannings**!
- Eine **regelmäßige** Priorisierung von Anforderungen sollte entweder in **festgelegten Product Backlog Refinements**, **Sprint Reviews** oder **kontinuierlich** im Sprintverlauf erfolgen.

## MoSCoW



## Kosten-Nutzen

- Für die Kosten-Nutzen Matrix müssen die Kosten und der Nutzen jeder einzelnen Anforderung ermittelt werden
- Monetärer Ansatz:** konkrete Ermittlung von **Kosten** und wirtschaftlichem Nutzen in EUR
- Abstrakter Ansatz**: Ermittlung von Kosten und Nutzen gemäß individueller Kriterien und Skalen



## Risiko-Nutzen

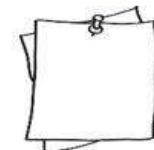
- Die **Risiko-Nutzen Matrix** folgt demselben Ansatz und erlaubt eine Einordnung von User Stories anhand klarer Kriterien, ersetzt jedoch die Kosten durch das Risiko
- Risiken hinsichtlich der **Umsetzung** (bspw. Know-How, **technische Abhängigkeiten**, mögliche Komplikationen)
- Risiken hinsichtlich der **Marktakzeptanz** (will der Kunde diese **Funktionalität** wirklich haben?)



# KANBAN REGELN

## Visualisierung der Arbeit

Durch die Erstellung eines visuellen Modells kann der Arbeitsablauf beobachtet werden, der sich durch das Kanban-System bewegt. Die Arbeit sichtbar zu machen - zusammen mit Blockern, Engpässen und Warteschlangen - führt zu einer erhöhten Kommunikation und Zusammenarbeit.



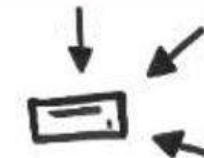
## Fokus auf den Flow

Durch die Nutzung von WIP-Limits und die Entwicklung von teamorientierten Strategien kann der reibungslose Ablauf der Arbeit verbessert, Metriken und der Arbeitsfluss analysiert und sogar Indikatoren für zukünftige Probleme erkannt werden.



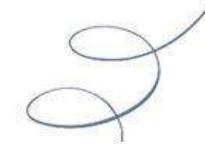
## Limitierung von Work in Process

Durch die Begrenzung, wieviel unfertige Arbeit in Bearbeitung ist, kann die Zeit reduziert werden, die eine Aufgabe braucht um durch das Kanban-System zu reisen. So können auch Probleme vermieden werden, die durch den ständigen Wechsel von Tasks verursacht werden.

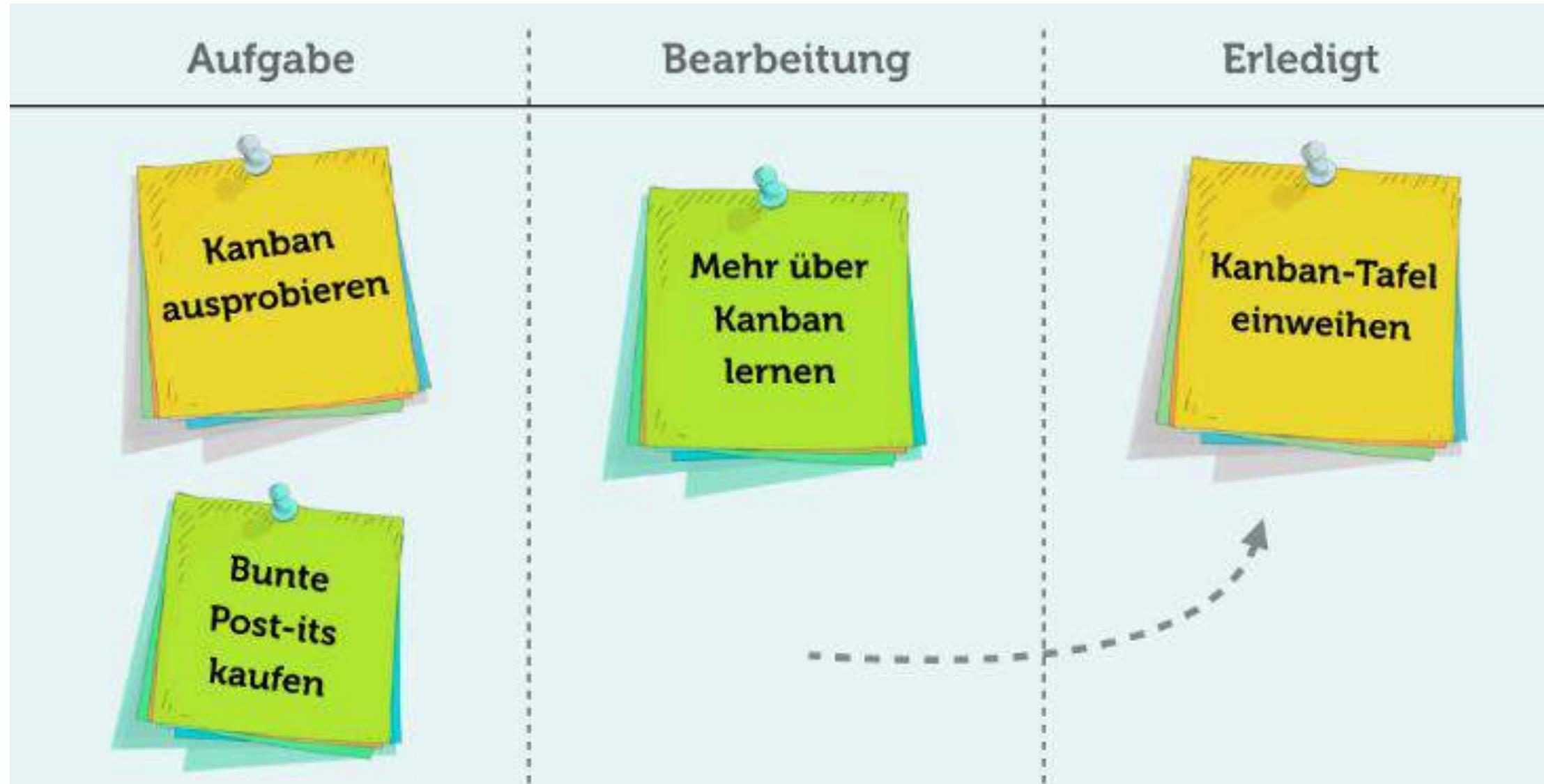


## Kontinuierliche Verbesserung

Teams messen ihre Wirksamkeit durch die Verfolgung von Qualität, Durchsatz, Durchlaufzeiten und mehr. Sie nutzen Retrospektiven und Experimente um das System zu verändern, und die Wirksamkeit des Teams zu verbessern..



# KANBAN BOARD MINIMALVERSION



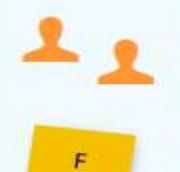
# KANBAN BOARDS



Was bedeutet  
**"fertig"** im Kanban  
Board?

Konkretisierung durch  
Spalten wie "**In Test**", oder  
**"abgenommen"** können  
helfen!

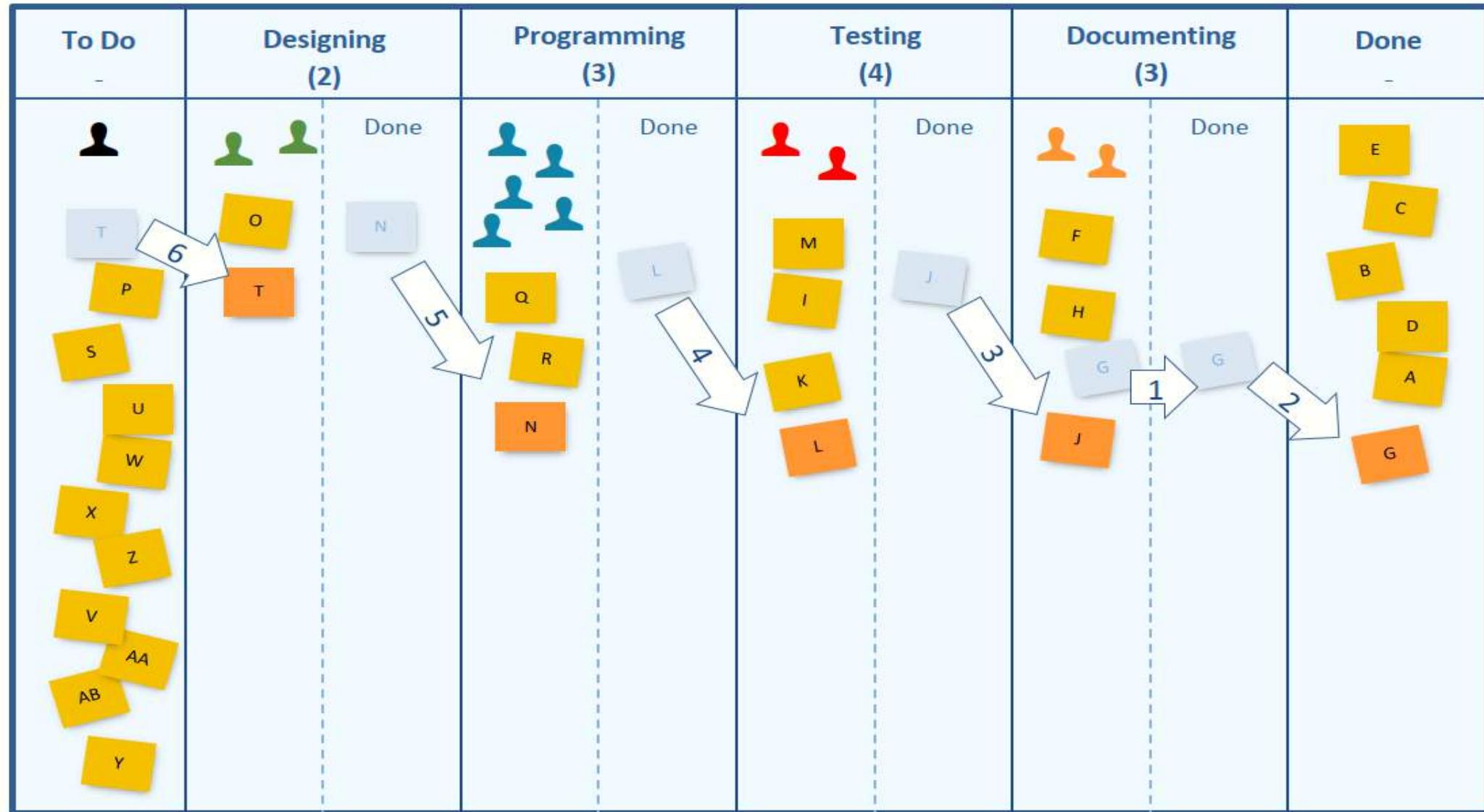
# KANBAN BOARD: BEISPIEL FÜR IT

To Do -	Designing (2)	Programming (3)	Testing (4)	Documenting (3)	Done -	
  T P S U W X Z V AA AB Y	  O N	  Q R	  L	  M I K J	  F H G	  E C B D A

**Work in Progress (WIP)  
hier: 4**  
Beide Spalten werden mitgezählt!

**Limit für Work in Progress (WIP)  
hier: 3**

# KANBAN BOARD: PROZESS FÜR DEN FALL „DOKUMENTATION FÜR G IST FERTIG“



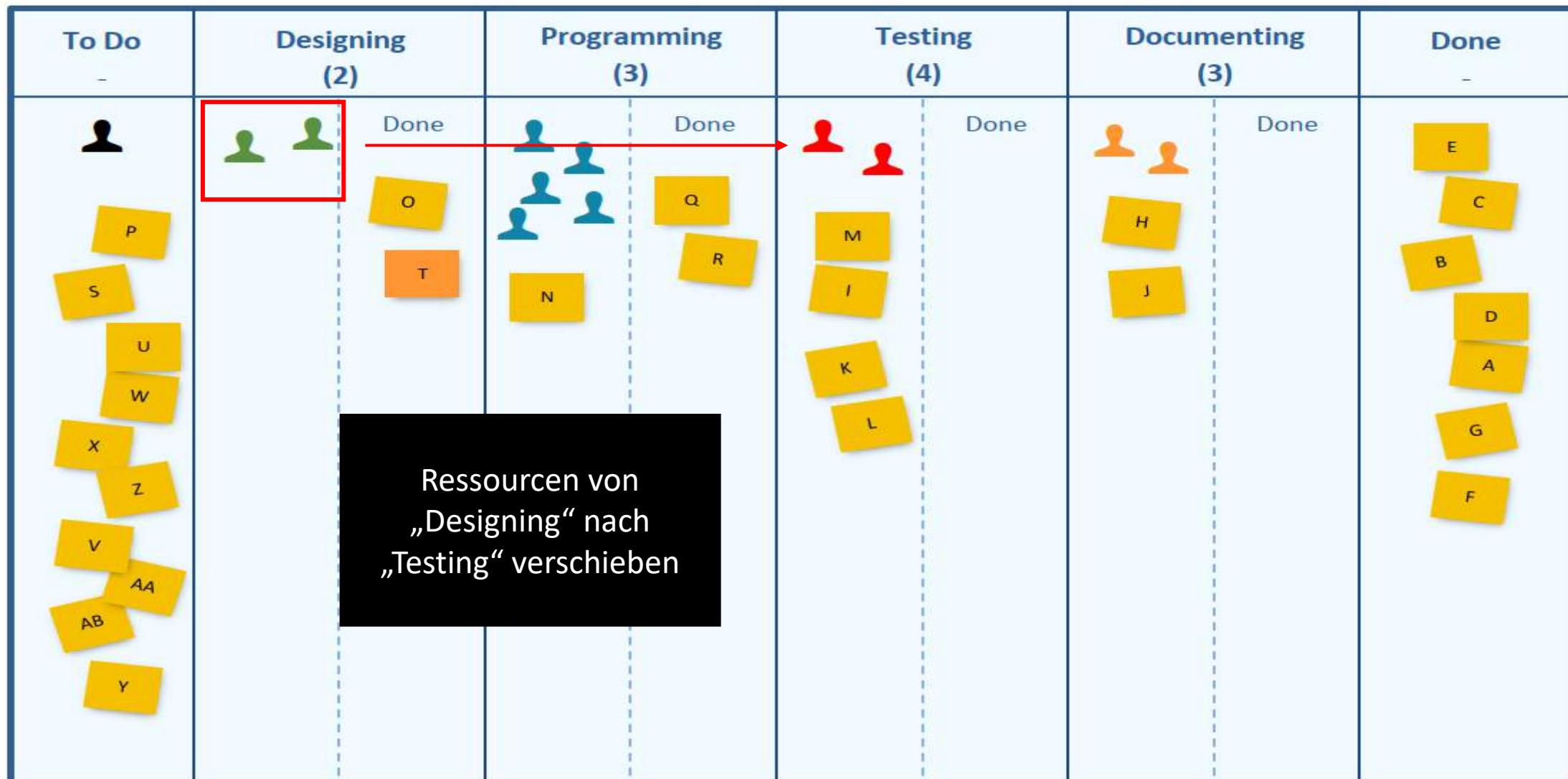
# KANBAN BOARD: NEUES BOARD NACH „DOKUMENTATION FÜR G IST FERTIG“

To Do -	Designing (2)	Programming (3)	Testing (4)	Documenting (3)	Done -

# KANBAN BOARD: ENGPASS SITUATION

To Do -	Designing (2)	Programming (3)	Testing (4)	Documenting (3)	Done -
                                                                                                                                                       <img alt="User icon" data-bbox="185 1625 21					

# KANBAN BOARD: ENGPASS BEHEBEN DURCH RESSOURCENUMVERTEILUNG

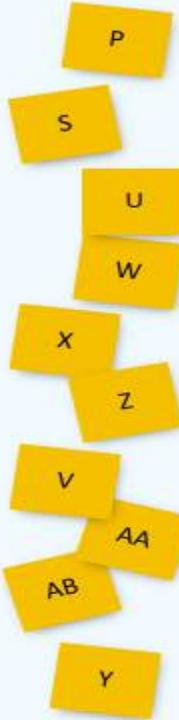
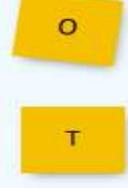
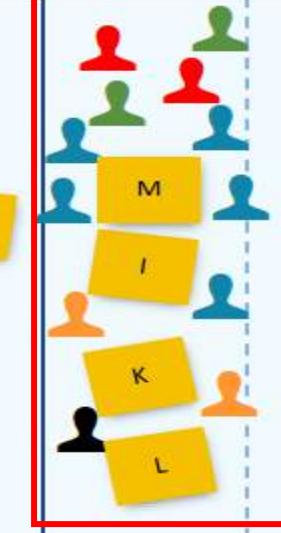
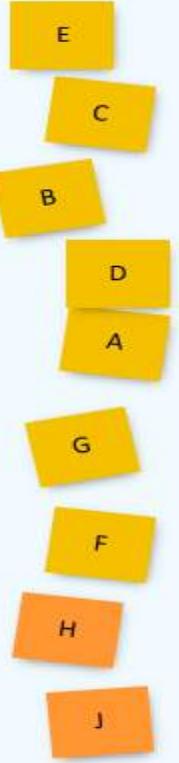


# KANBAN BOARD: ERGEBNIS NACH RESSOURCENUMVERTEILUNG

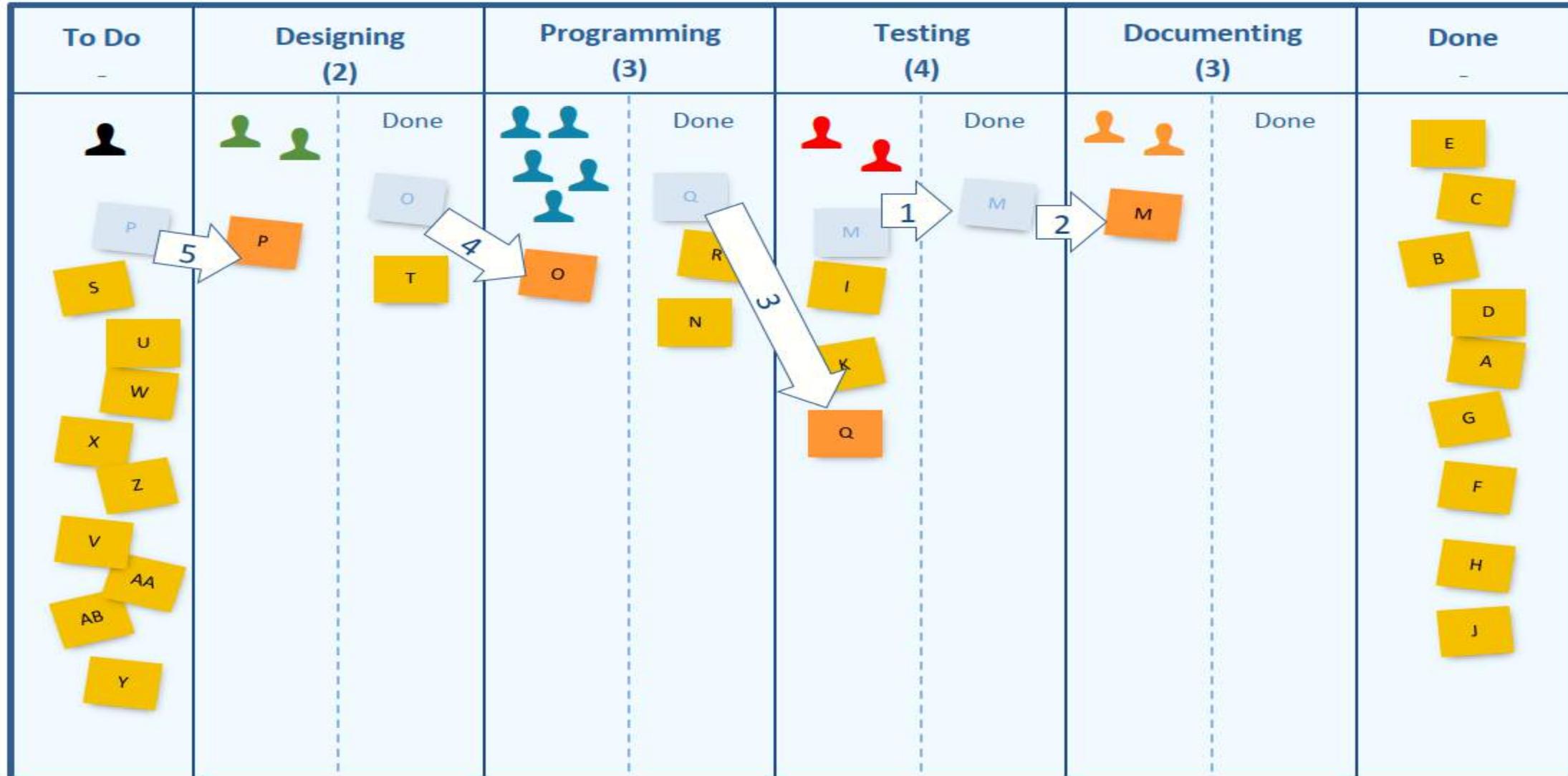
To Do	Designing (2)	Programming (3)	Testing (4)	Documenting (3)	Done
  P S U W X Z V AA AB Y	Done  O T	Done  Q R N	Done  M I K L	Done  H J	Done  E C B D A G F

Bei ausgeprägten Engpassen weiter Ressourcen umverteilen

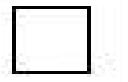
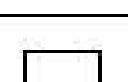
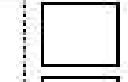
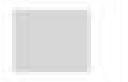
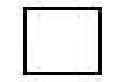
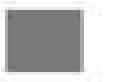
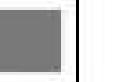
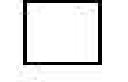
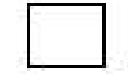
# KANBAN BOARD: ERGEBNIS NACH 2. RESSOURCENUMVERTEILUNG

To Do -	Designing (2)	Programming (3)	Testing (4)	Documenting (3)	Done -
	 Done	 Done	 Done		 Done

# KANBAN BOARD: WORKFLOW IST WIEDERHERGESTELLT



# IMPEDIMENT BOARD: HINDERNISSE VISUALISIEREN

	Ideation	To Do	Design		Dev. & Test		Inte- gration		Pre- prod	Ready!
Dev. Team 1	  	 	 		  	 			 	 
Dev. Team 2		 					 			 
Dev. Team 3					 					
Dev. Team 4		 			 	 				

# IMPEDEMENT BOARD: HINDERNISSE VISUALISIEREN

Beispiel Impediment Backlog:

<b>Impediment</b>	<b>Ersteller</b>	<b>Datum</b>	<b>Status</b>	<b>Priorität</b>	<b>Verantwortlich</b>
Team-Raum zu klein	ALS	17.06.2013	In Arbeit	Hoch	ALS
Klimaanlage reparieren	ALS	15.06.2013	In Arbeit	Hoch	ZUB
Testing- Umgebung fehlt	ALS	18.06.2013	Neu	Normal	ADM
Product- Backlog unvollständig	ALS	18.06.2013	Neu	Hoch	PO

# SCRUM SKALIERUNG



**More than one  
Scrum Team is  
working on the  
same Product.**



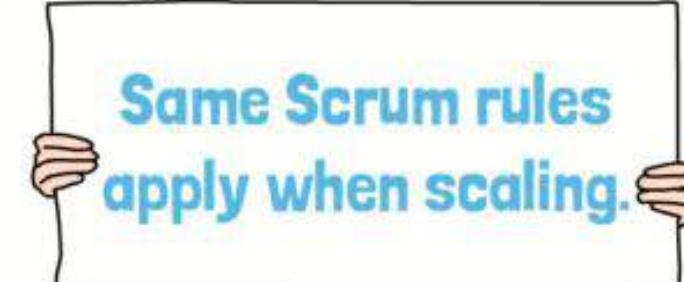
**faster development pace**



**more output**



**more value**



# SCRUM SKALIERUNG



The Nexus™ Guide  
The Definitive Guide to Scaling Scrum with Nexus

January 2021

## The Scrum@Scale® Guide

The Definitive Guide to Scrum@Scale:  
Scaling that Works



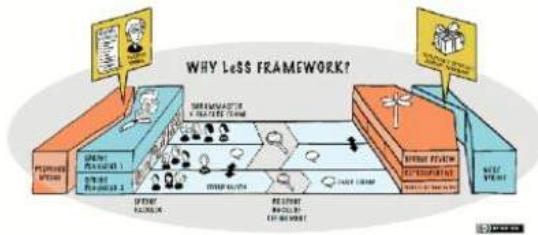
Version 2.1 — January 2021

©2006-2021 Jeff Sutherland and Scrum Inc., All Rights Reserved  
Scrum@Scale is a registered trademark of Scrum Inc.  
Released under Creative Commons 4.0 Attribution-ShareAlike License

Large Scale Scrum (LeSS)

Copyright © 2014 - 2019 The LeSS Company B.V.

## LeSS Framework



Scaling Scrum starts with understanding standard one-team Scrum. From that point, your organization must be able to understand and adopt LeSS, which requires examining the purpose of one-team Scrum elements and figuring out how to reach the same purpose while staying within the constraints of the standard Scrum rules.

Agile development with Scrum requires a deep organizational change to become agile. Therefore, neither Scrum nor LeSS should be considered as merely a practice. Rather, they form an organizational design framework.

## Two Agile Scaling Frameworks

LeSS provides two different large-scale Scrum frameworks. Most of the scaling elements of LeSS are focused on directing the attention of all of the teams onto the whole product instead of "my part." Global and "end-to-end" focus are perhaps the dominant problems to solve in scaling. The two frameworks – which are basically single-team Scrum scaled up – are:

- LeSS: Up to eight teams (of eight people each).
- LeSS Huge: Up to a few thousand people on one product.

Version: November 2019

<https://less.works/>

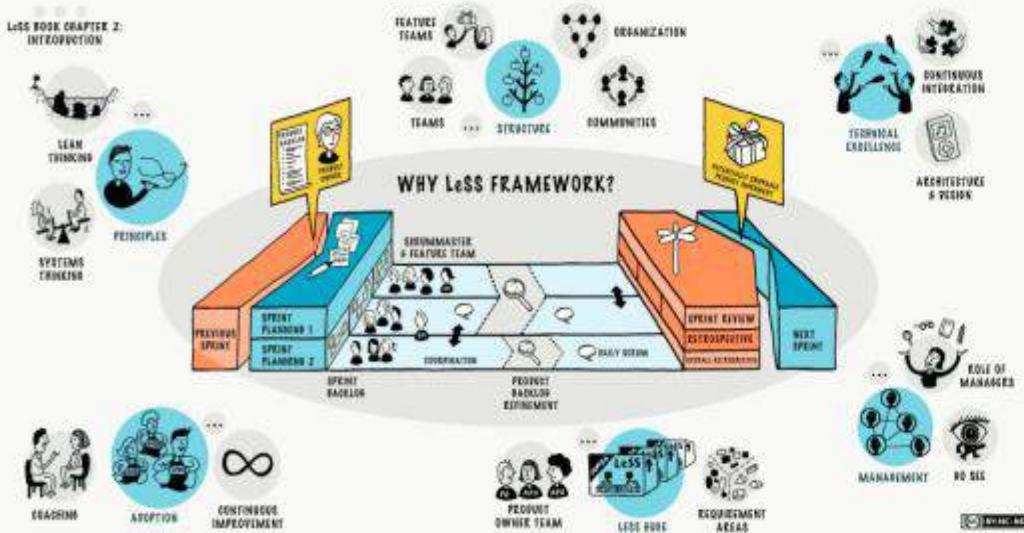
1

# SCRUM SKALIERUNG

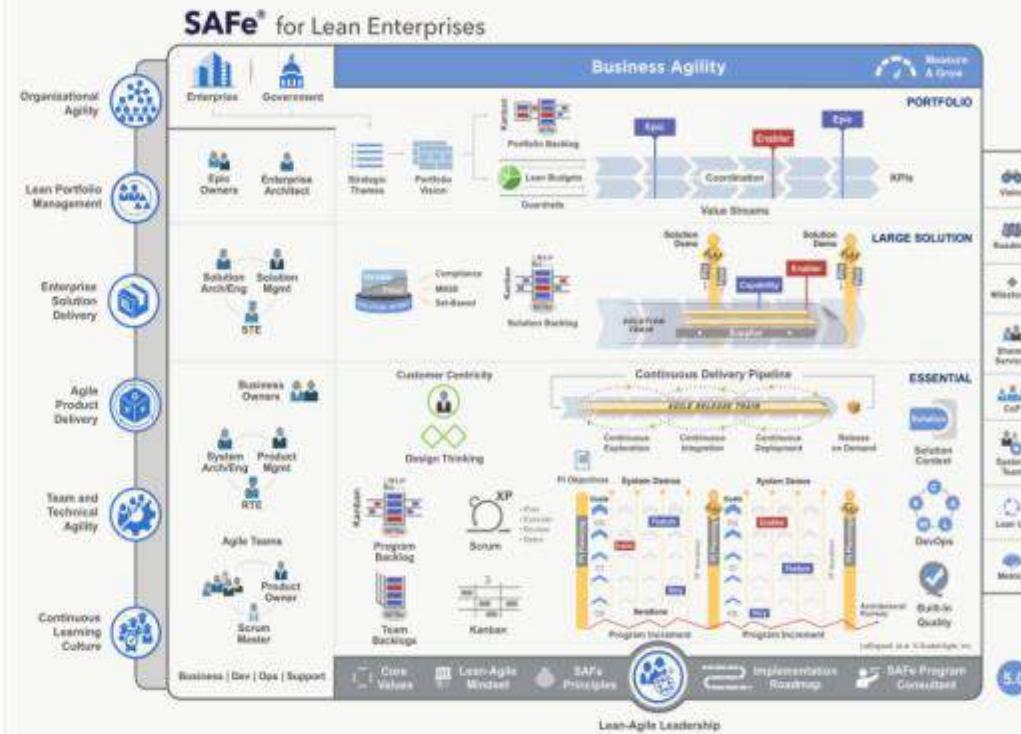
## Weitere Skalierungsansätze

edu bs

- ▶ Large-Scale Scrum – LeSS™  
[less.works](http://less.works)



- ▶ Scaled Agile Framework – SAFe™  
[www.scaledagileframework.com](http://www.scaledagileframework.com)

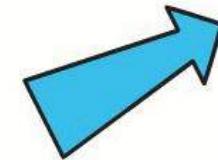


# SCRUM SKALIERUNG



## The Nexus™ Guide

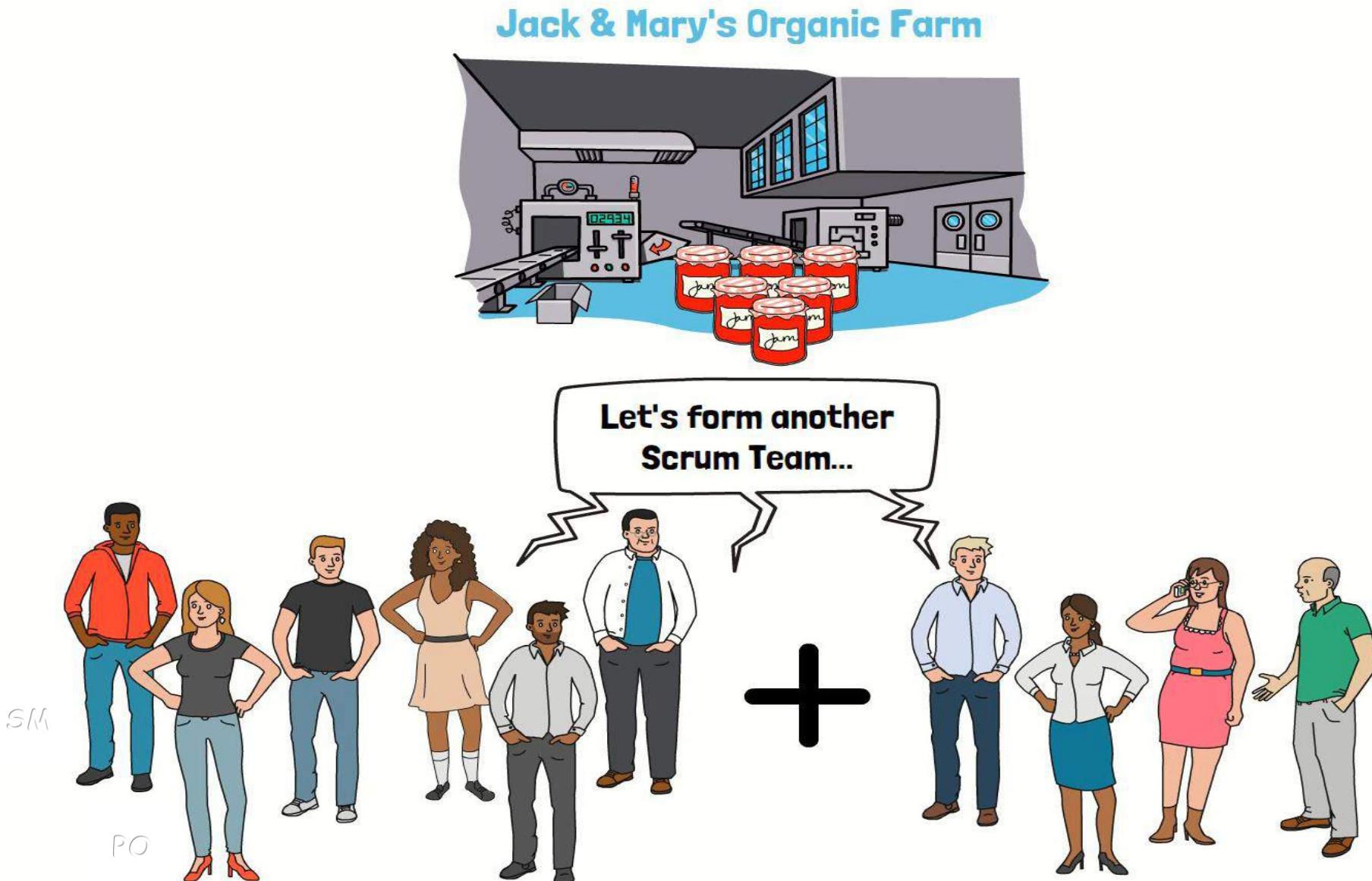
The Definitive Guide to Scaling Scrum with Nexus



January 2021

**Not part of the  
PSM I exam.**

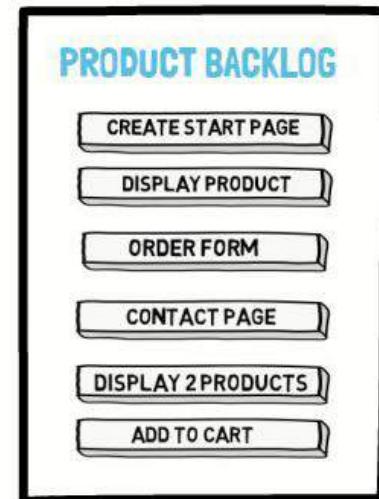
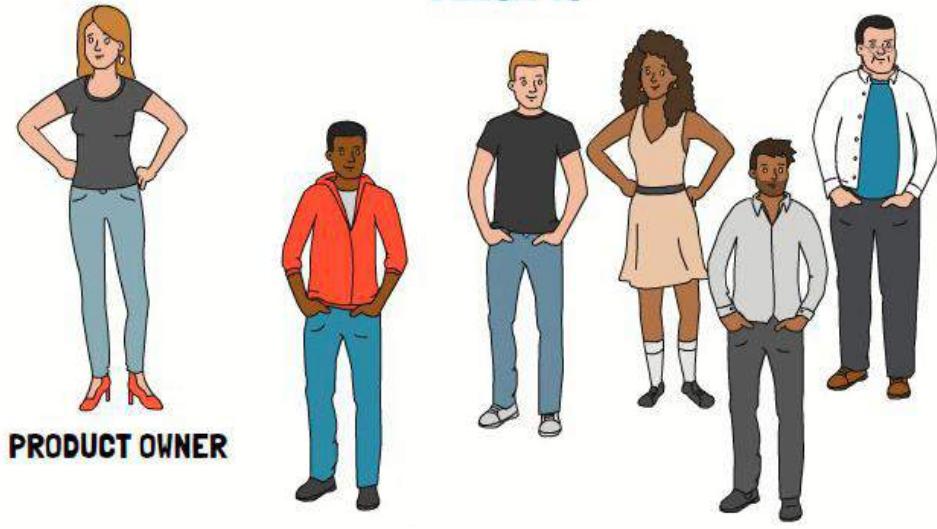
# SCRUM SKALIERUNG



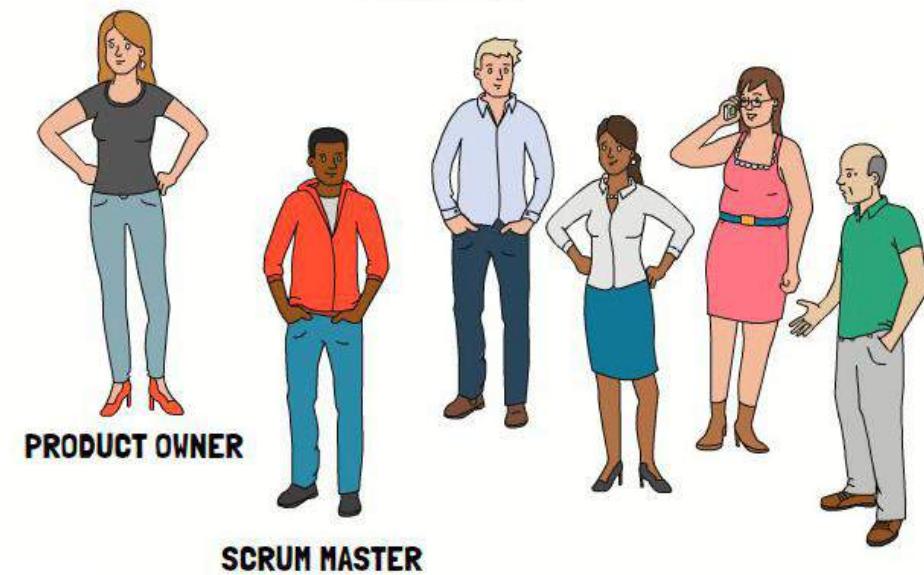
# SCRUM SKALIERUNG

**1 PRODUCT => 1 PRODUCT BACKLOG => 1 PRODUCT OWNER**

TEAM A



TEAM B



**SM kann ein- und dieselbe Person sein...**

# SCRUM SKALIERUNG

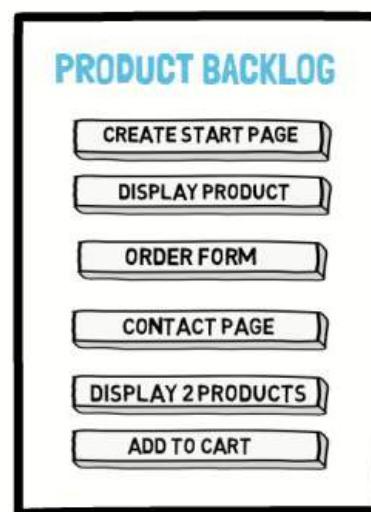
**1 PRODUCT => 1 PRODUCT BACKLOG => 1 PRODUCT OWNER**

TEAM A



PRODUCT OWNER

SCRUM MASTER



TEAM B



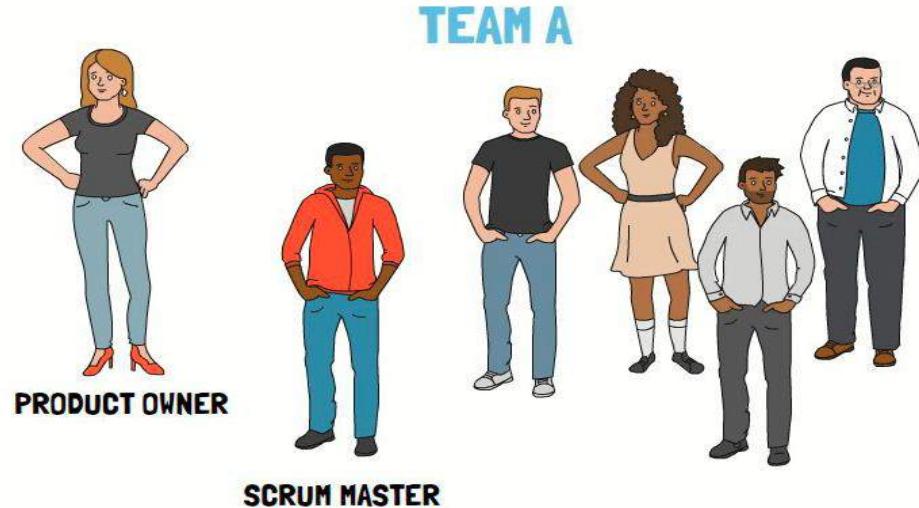
PRODUCT OWNER

SCRUM MASTER

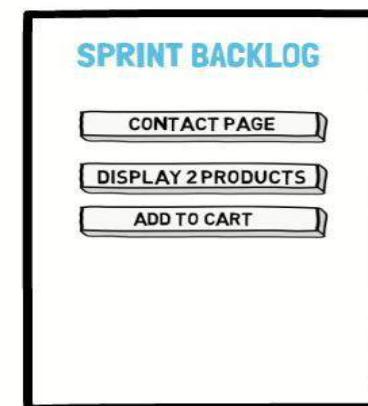
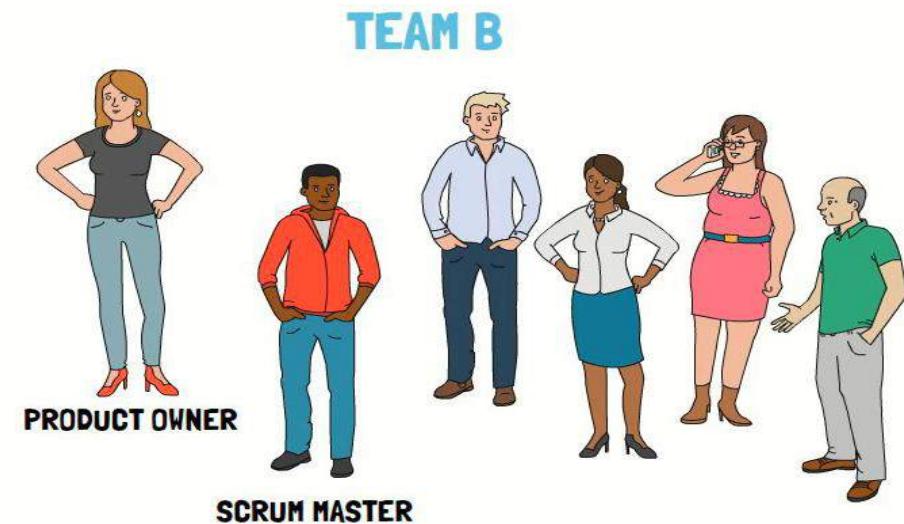
...muss aber nicht ein- und dieselbe Person sein

# SCRUM SKALIERUNG

**1 PRODUCT => 1 PRODUCT BACKLOG => 1 PRODUCT OWNER**



All Scrum Teams share  
ONE Product Backlog!



# SCRUM SKALIERUNG

Impact on velocity  
when scaling Scrum

AVERAGE VELOCITY: 30 SP

TEAM A



PRODUCT BACKLOG

- CREATE START PAGE
- DISPLAY PRODUCT
- ORDER FORM
- CONTACT PAGE
- DISPLAY 2 PRODUCTS
- ADD TO CART

AVERAGE VELOCITY: 15 SP?

TEAM B



# SCRUM SKALIERUNG



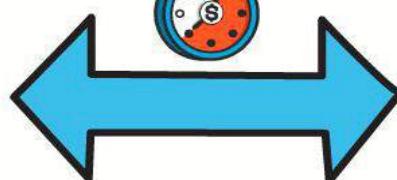
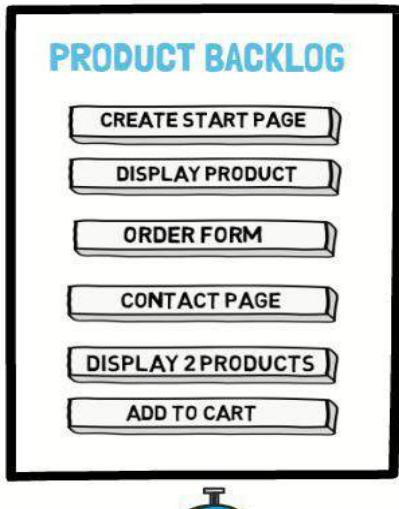
Any changes to a Scrum Team result in a **DECREASE** in velocity.

AVERAGE VELOCITY: 30 SP

TEAM A



Sprint 3



AVERAGE VELOCITY: 15 SP?

TEAM B



Dependencies should be reduced as much as possible.

Any dependencies need to be **TRANSPARENT**.



# SCRUM SKALIERUNG



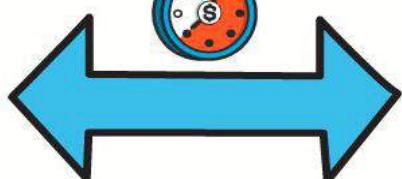
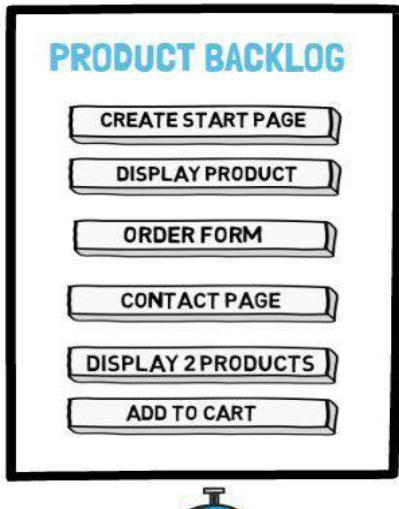
Any changes to a Scrum Team result in a **DECREASE** in velocity.

AVERAGE VELOCITY: 30 SP

TEAM A



Sprint 3



AVERAGE VELOCITY: 15 SP?

TEAM B

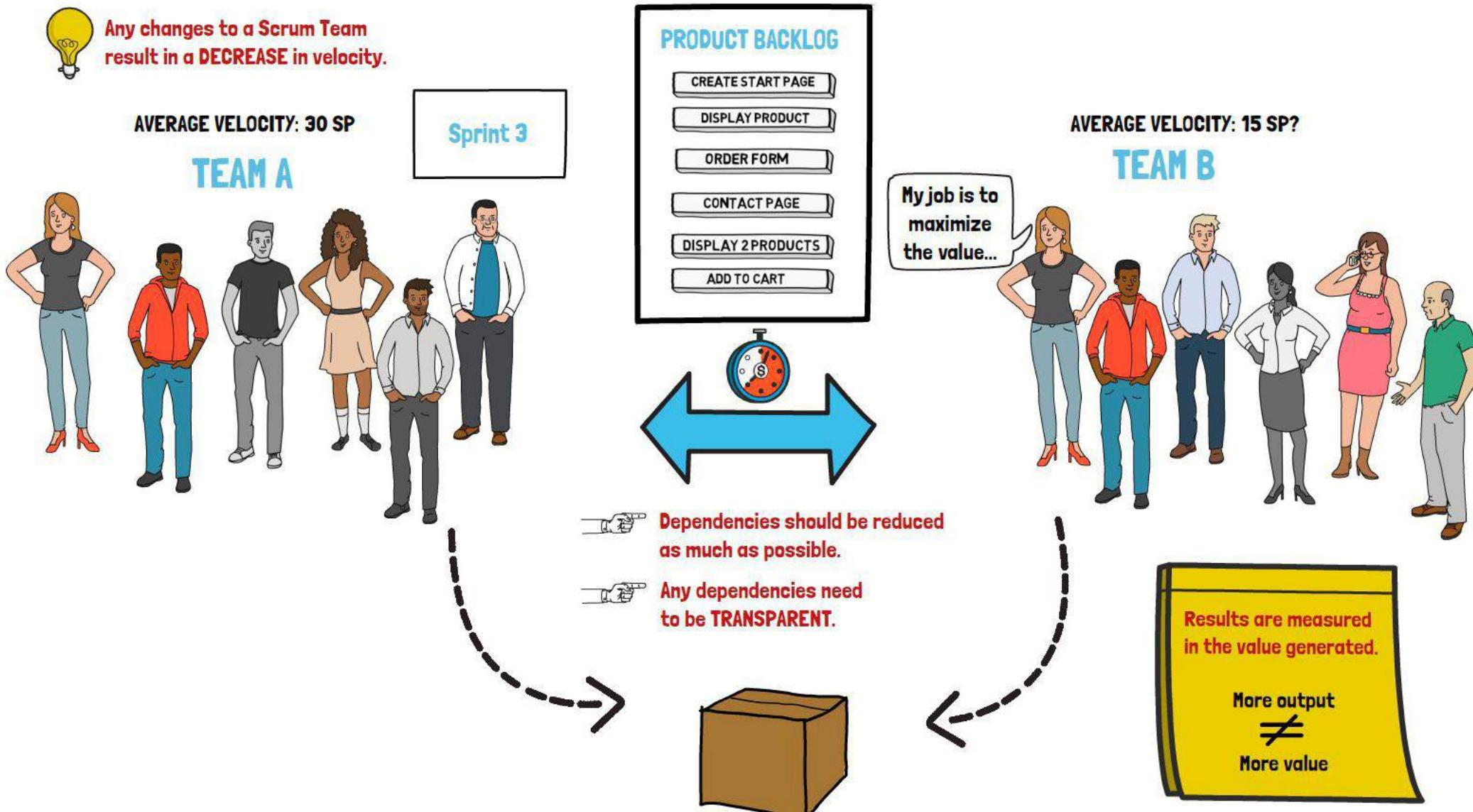


Dependencies should be reduced as much as possible.

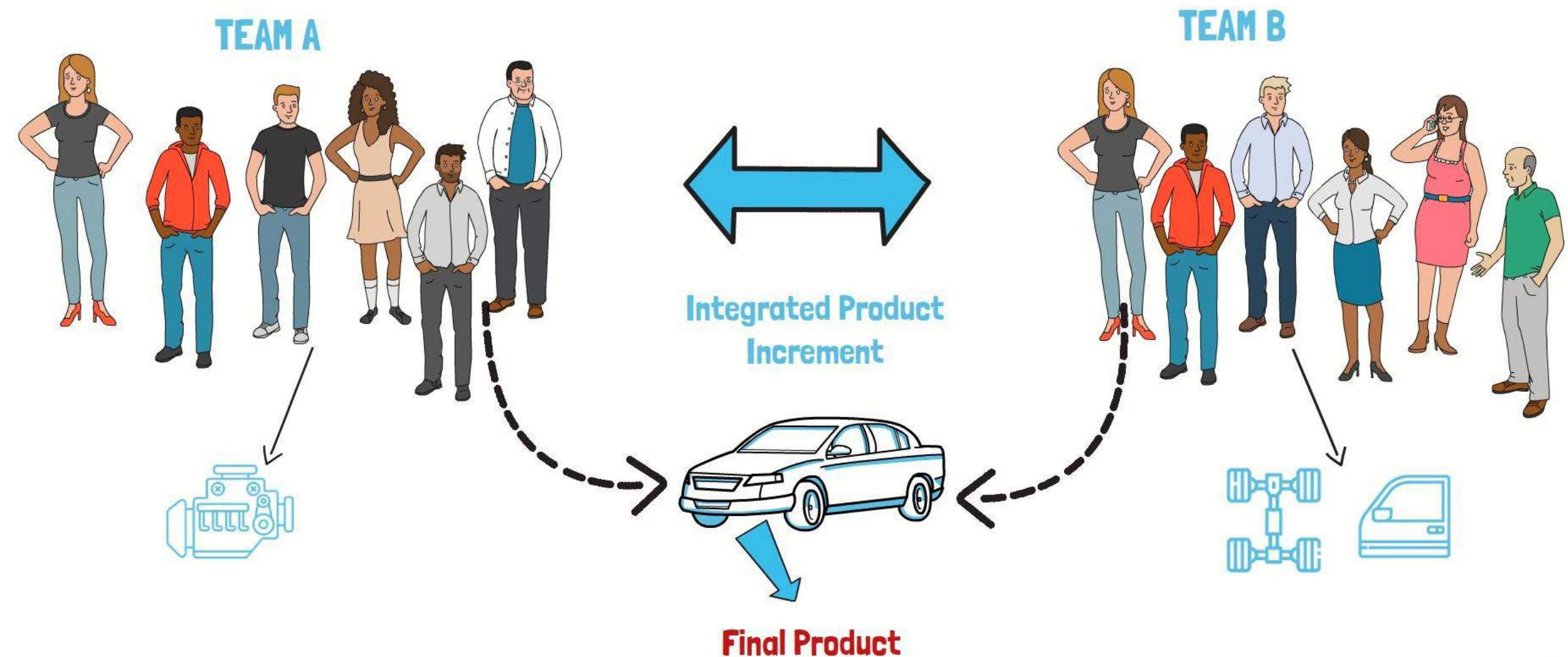
Any dependencies need to be **TRANSPARENT**.



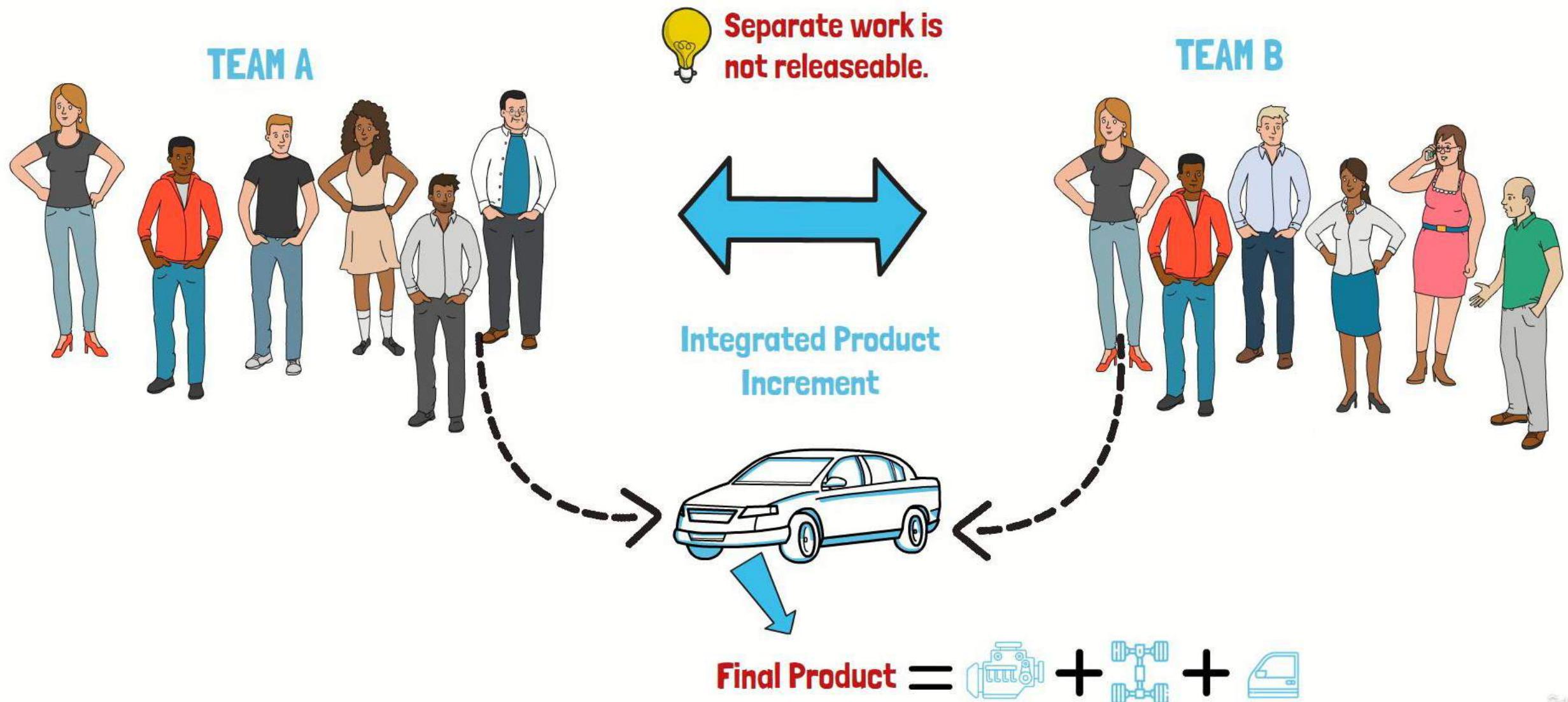
# SCRUM SKALIERUNG



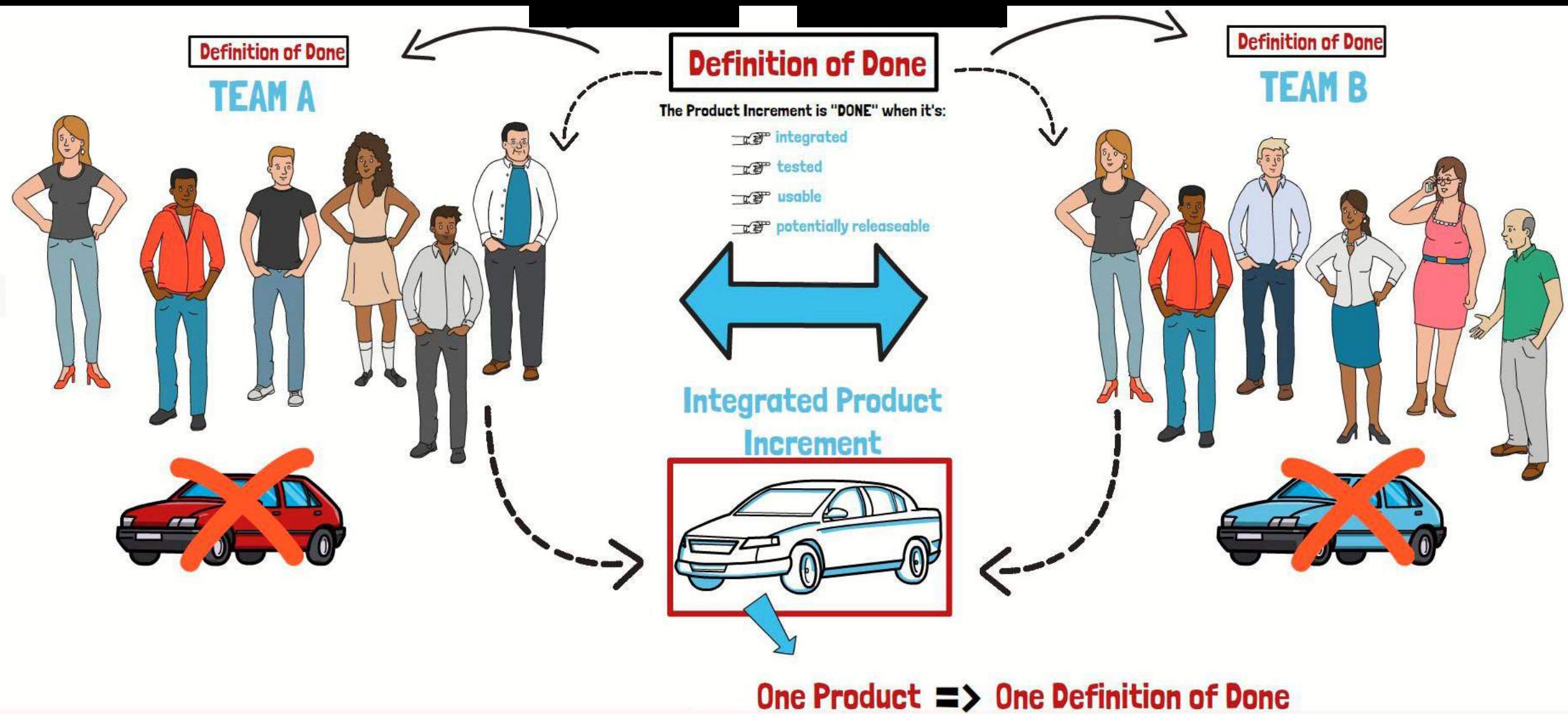
# INTEGRIERTE PRODUKTINKREmente



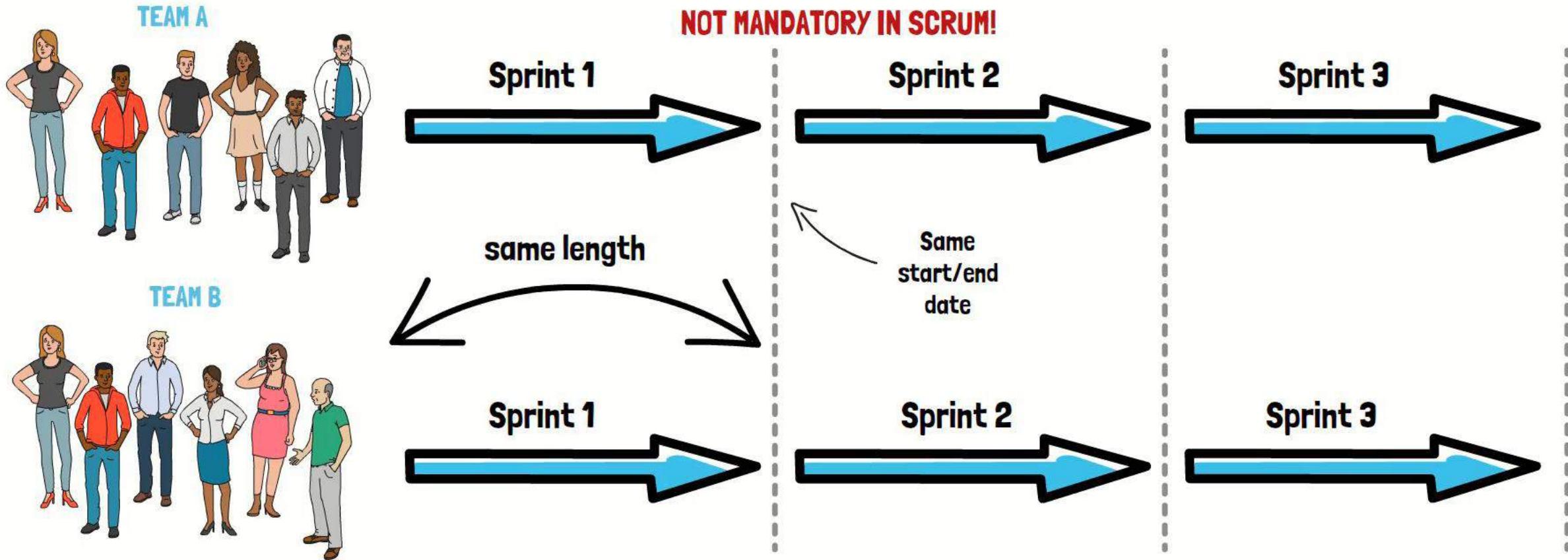
# INTEGRIERTE PRODUKTINKREMENTE



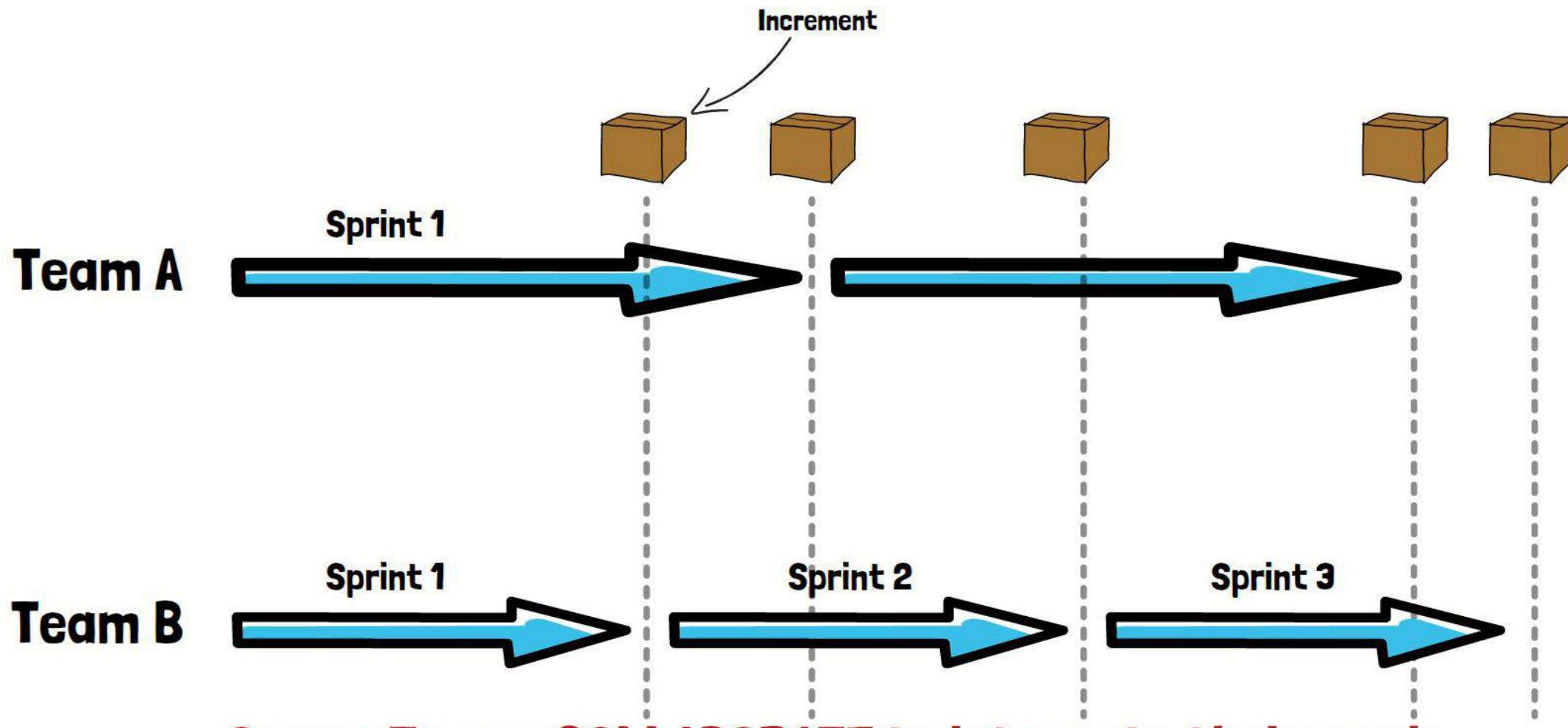
# DEFINITION OF DONE BEI SKALIERUNG



# SPRINTLÄNGE BEI SKALIERUNG

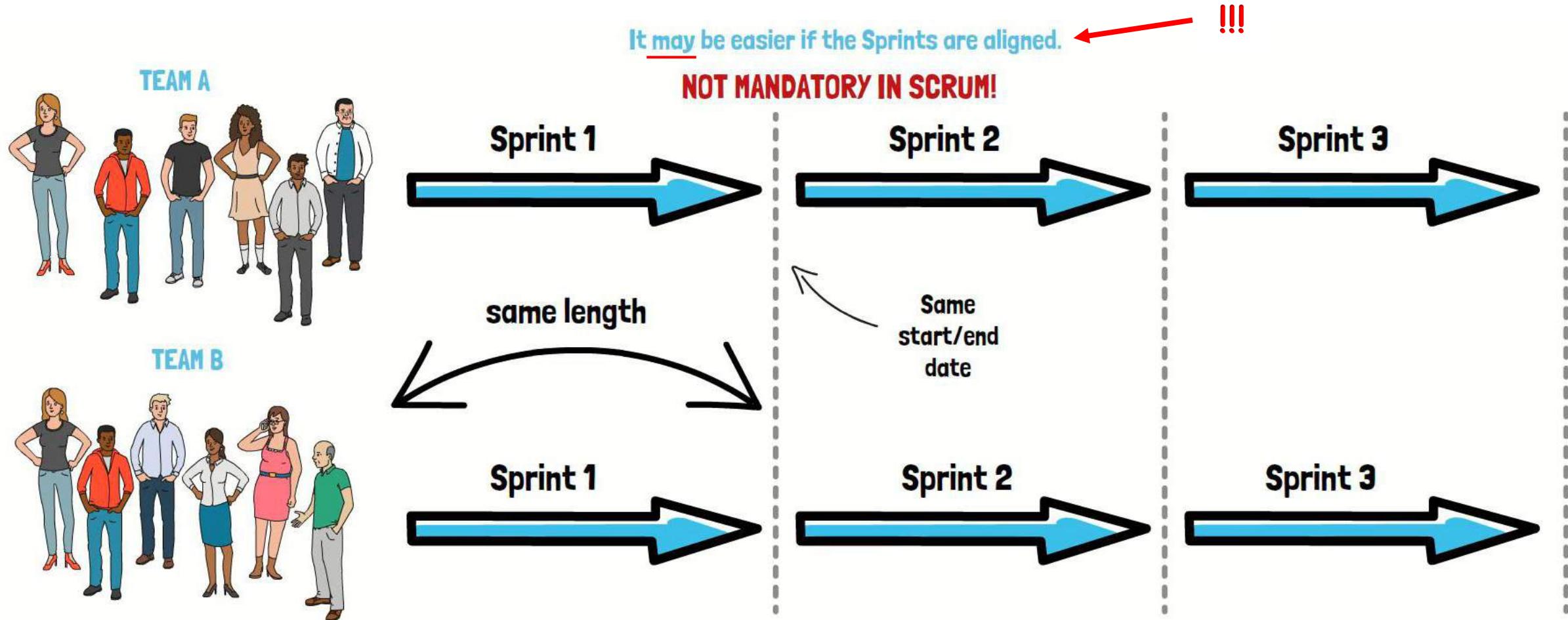


# SPRINTLÄNGE BEI SKALIERUNG

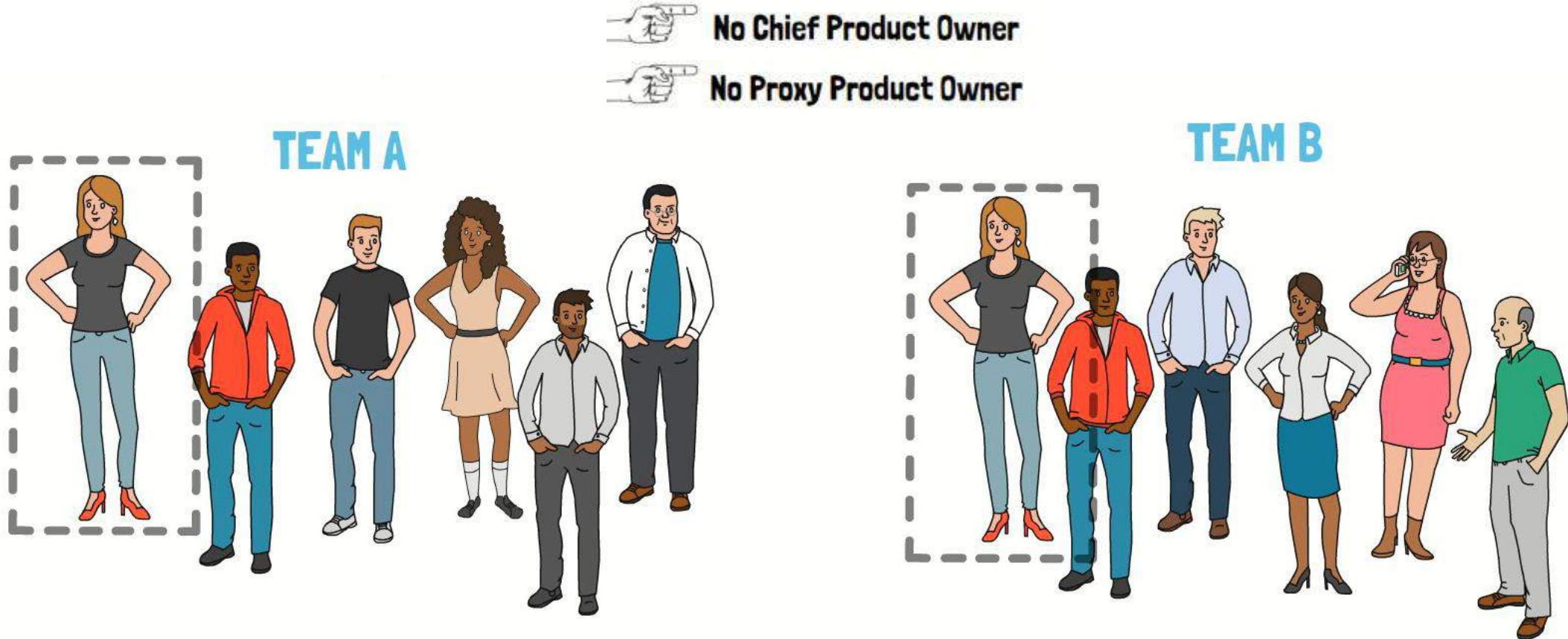


**Scrum Teams COLLABORATE to integrate their work.  
(it is their responsibility!)**

# SPRINTLÄNGE BEI SKALIERUNG

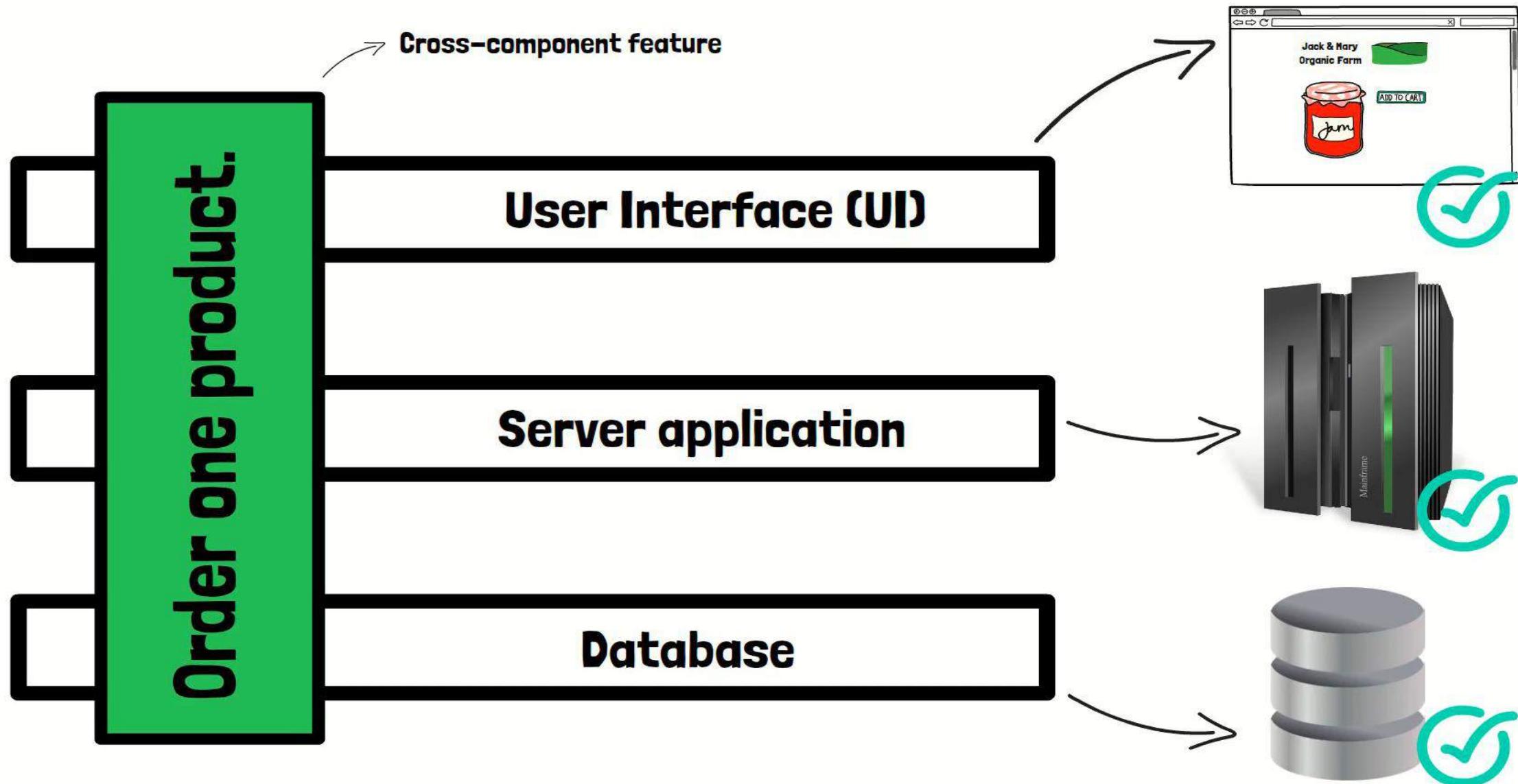


# WIE VIELE PRODUCT OWNER BEI SKALIERUNG?

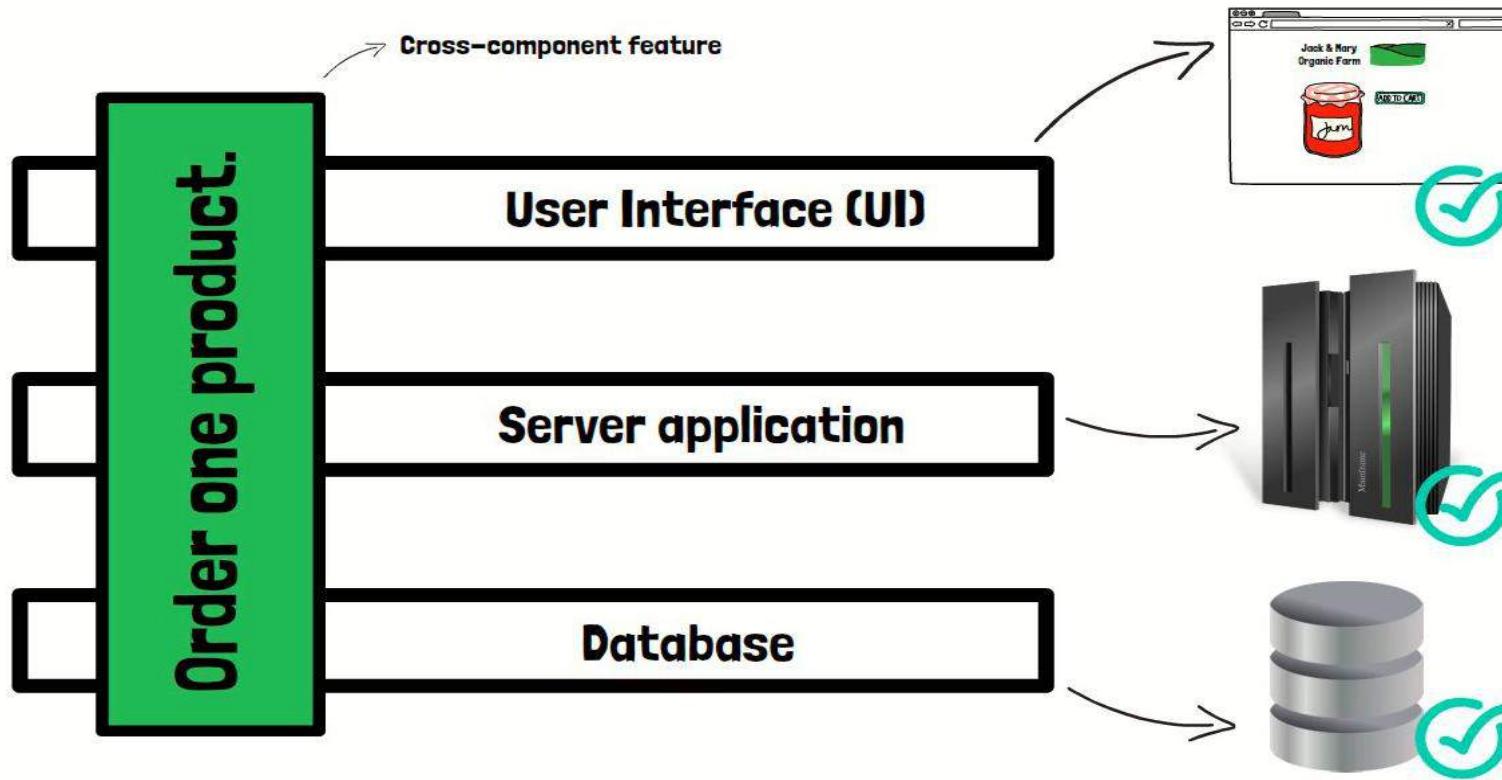


**1 Product => 1 Product Backlog => 1 Product Owner (one person)**

# FEATURE TEAMS VS. COMPONENT TEAMS



# FEATURE TEAMS VS. COMPONENT TEAMS



**FEATURE TEAM -** **works through ALL the layers or components of the application to fulfil a customer need.**

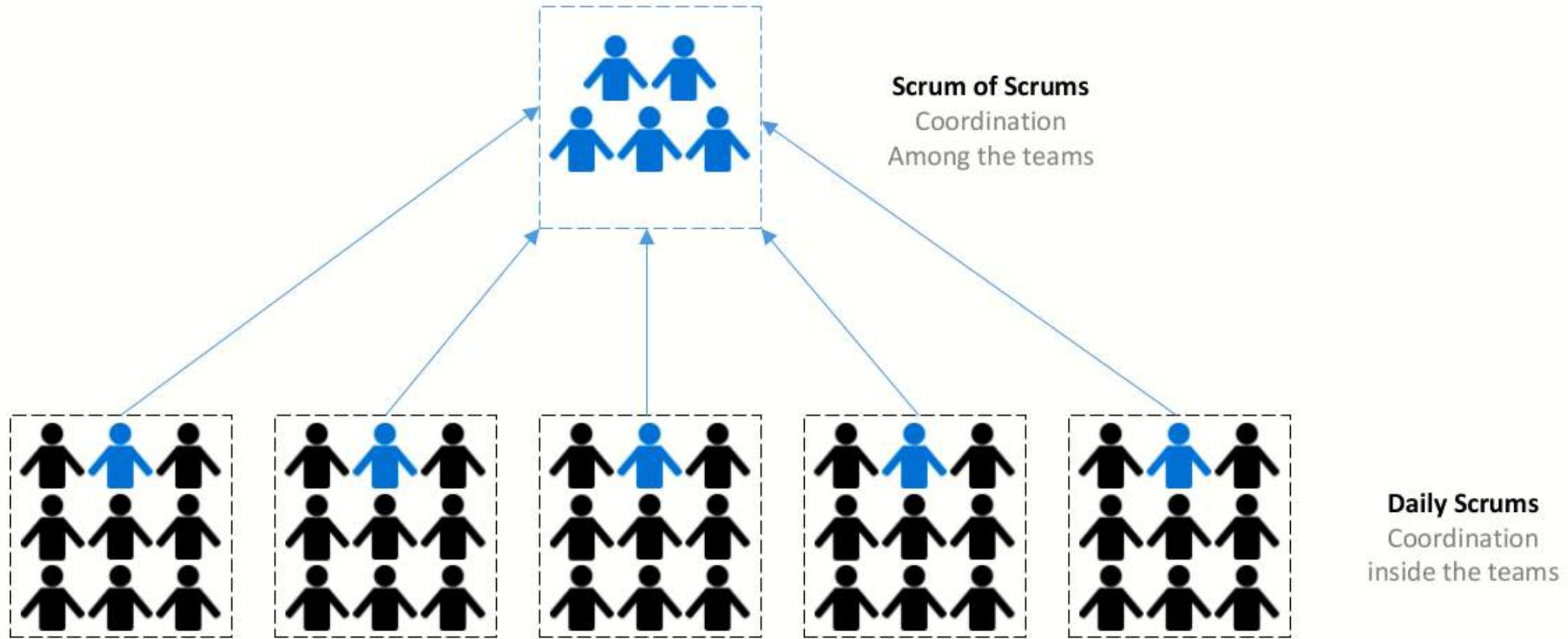
**COMPONENT TEAM -** **focused on a SINGLE layer or component of the application.**

# SCRUM SKALIERUNG: WICHTIGSTE REGELN



- 1 Product => 1 Product Backlog => 1 Product Owner**
- Scrum Teams must have a SHARED Definition of Done.**
- Each Sprint must produce an INTEGRATED Increment.**

# SCRUM OF SCRUMS



# SCRUM OF SCRUMS

Eine kurze Definition von Scrum of Scrums:

Wenn ein Scrum-Team zu groß wird, bilden Sie mehrere Teams.

Die Scrum-Teams arbeiten mit ihren eigenen Team und Sprint Backlogs  
und halten ihre eigenen Daily Scrum-Meetings.

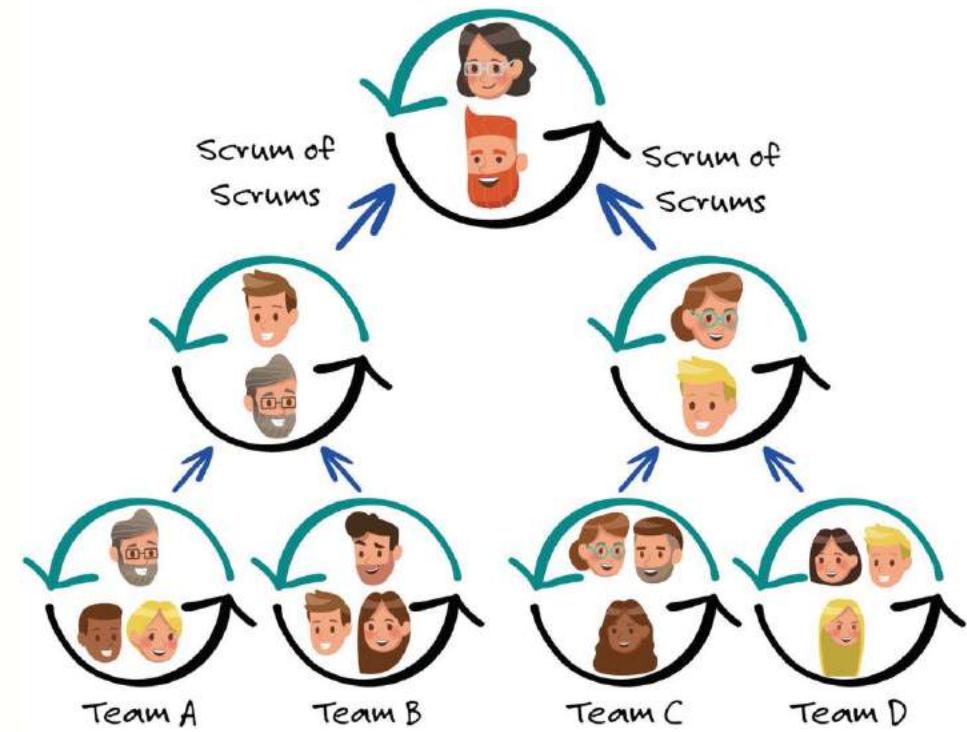
Um sich abzustimmen, findet ein Scrum of Scrums-Meeting zwischen den Teams statt.

# SCRUM OF SCRUMS

## Scrum of Scrums Definition

Das Scrum of Scrums bezeichnet ein regelmäßiges Treffen von Vertretern einzelner Scrum-Teams. Es dient dem Zweck, sich gegenseitig über den Status quo der einzelnen Teams, über anstehende Tätigkeiten und mögliche Hindernisse bei der Entwicklung auszutauschen. Ziel des Scrum of Scrums ist es, die Arbeit der verschiedenen Scrum-Teams zu synchronisieren und Entwicklungen von Teams zu identifizieren, die andere Teams bei der Umsetzung ihrer Anforderungen beeinflussen. Es ist somit eine Technik zur Skalierung von Scrum und für viele größere Unternehmen relevant.

Idealerweise schickt jedes Team einen Vertreter zum Scrum-of-Scrums-Meeting, so dass alle Teams gleichgewichtig repräsentiert werden. Die Arbeit der Scrum-of-Scrums Meetings kann auch auf einer höheren Ebene – mit Vertretern der verschiedenen Scrum-of-Scrums-Runden – fortgeführt werden. Auch wenn Formulierungen wie *Scrum-of-Scrum-of-Scrums* leicht verständlich sind, haben sie sich in der Scrum-Community nicht durchgesetzt.



# SCRUM OF SCRUMS

## Unterschiede zwischen Daily Scrum und Scrum of Scrums

Daily Scrum und Scrum of Scrums sind sehr ähnlich. Beim [Daily Scrum](#) darf jedes Teammitglied meist folgende drei Fragen beantworten:

- Was habe ich seit gestern getan, um das Sprint-Ziel zu erreichen?
- Was mache ich bis morgen, um das Sprint-Ziel zu erreichen?
- Was behindert mich bei meiner Arbeit?

Da beim Scrum of Scrums jeder Teilnehmer als Team-Botschafter auftritt, lassen sich die Fragen entsprechend umformulieren:

- Was hat mein Team geschafft, seit wir uns das letzte Mal getroffen haben?
- Was wird mein Team bis zum nächsten Treffen erledigen?
- Welche Hindernisse behindern mein Team bei der Arbeit?
- Könnte eine Tätigkeit meines Teams ein anderes Team beeinflussen oder behindern?

# SCRUM OF SCRUM OF SCRUMS

