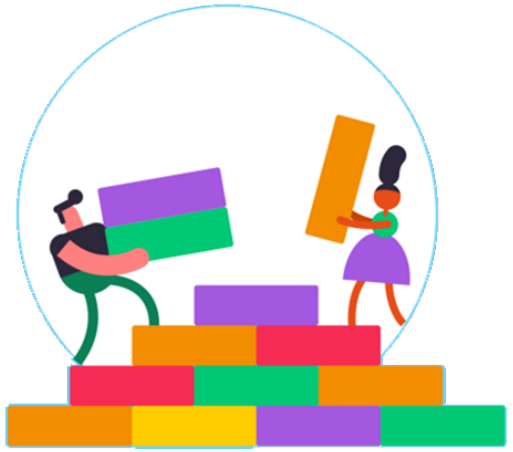
GFN Gemeinsam weiterbilden

EXIN Agile Scrum FOUNDATION Certified by EXIN

Agile Scrum Foundation

Inhaltsübersicht

Ziel der EXIN Agile Scrum Foundation Schulung



- Die Teilnehmer lernen die agilen Prinzipien und das Scrum-Framework kennen. Bei Agile und Scrum geht es um Zusammenarbeit, um ein Ziel erfolgreich zu erreichen.

- Agile Prinzipien sind in der Softwareentwicklung beliebt und werden zunehmend auch in anderen Bereichen eingesetzt. Zu den Scrum-Praktiken gehören die Bildung funktions-übergreifender und selbstverwalteter Teams, die am Ende jeder Iteration oder jedes Sprints ein funktionierendes Inkrement liefern.

- Die agile Denkweise ist vor allem im Bereich der Softwareentwicklung bekannt, doch werden die Grundsätze zunehmend auch in anderen Projektarten angewandt. Scrum ist die am häufigsten verwendete agile Methodik und eignet sich für alle Fachleute, die ihr Wissen über die neuesten Entwicklungen in den Bereichen IT und Projektmanagement auf dem neuesten Stand halten wollen, insbesondere für diejenigen, die Projekte leiten oder an ihnen teilnehmen.

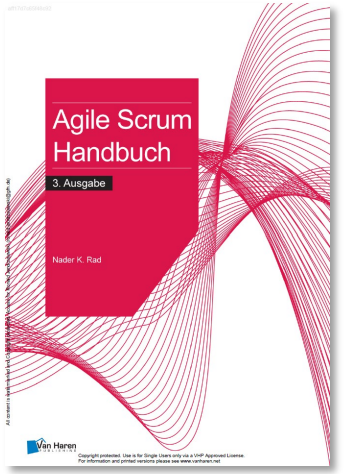
Exam Format

- Art der Prüfung: Multiple-Choice-Fragen
- Anzahl der Fragen: **40**
- Quote zum Bestehen: 65% **(26 Punkte)**
- Unterlagen/Notizen erlaubt? Nein
- Elektronische Geräte/Hilfsmittel erlaubt? Nein
- Dauer der Prüfung: **60 Minuten**
- Sprache: Deutsch

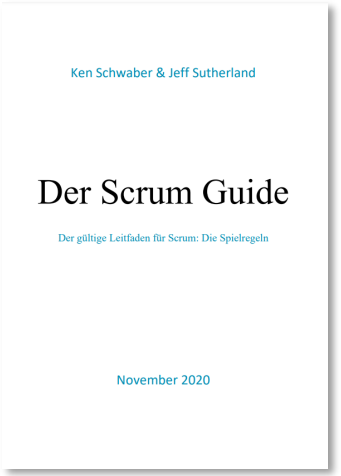
  Die Prüfung kann auf Wunsch auch in Mandarin, Niederländisch, Englisch, Französisch, Japanisch, Portugiesisch und Spanisch gemacht werden.)

Literatur

- Nader K. Rad | Agile Scrum Handbuch
  - Van Haren Publishing (3nd edition, 2021)
  - ISBN: 9789401807593 (hard copy)
  - ISBN: 9789401807609 (eBook)



- Ken Schwaber & Jeff Sutherland | The Scrum Guide 2020
  - z.B.: www.scrumguides.org



1. Das Agile Manifest

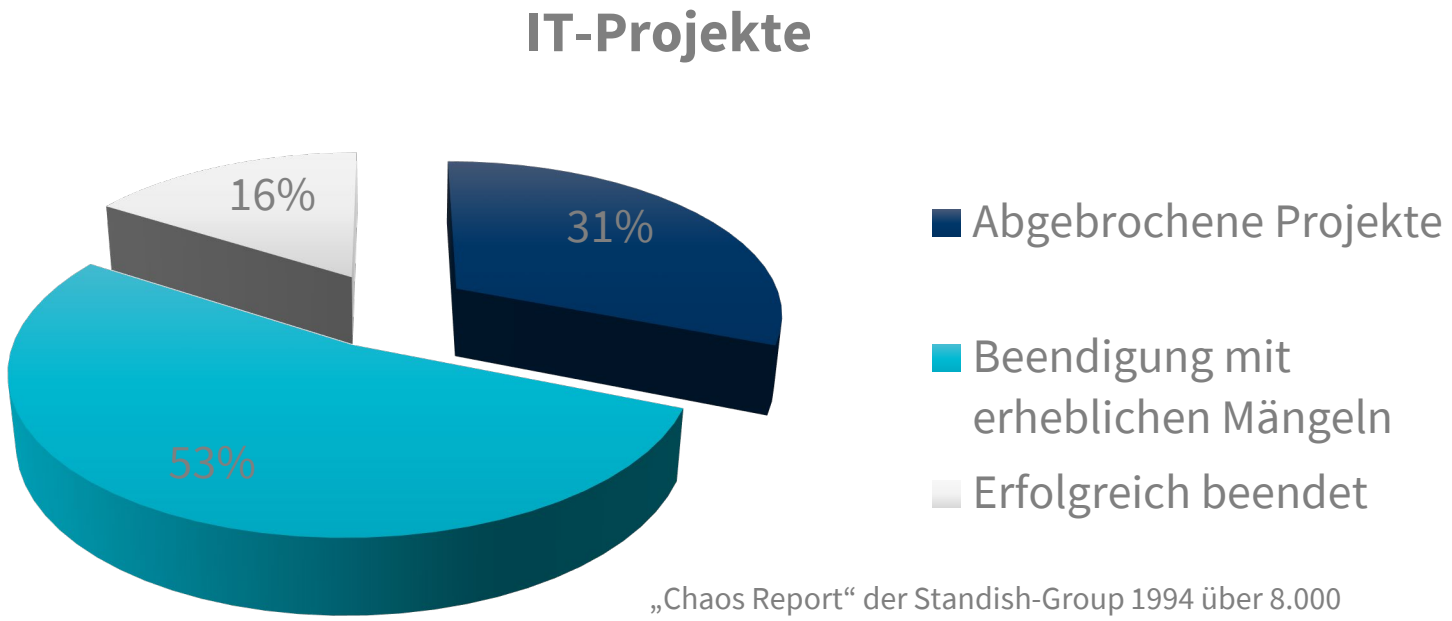
Treffen sich 17 führende Entwickler … (2001)

- Entstehung
- Werte
- Prinzipien

https://agilemanifesto.org/iso/de/manifesto.html

Die Ausgangssituation

- Mehr als zwei Drittel aller Softwareprojekte scheiterten oder gerieten in Schwierigkeiten. Sie wurden, wenn überhaupt, zu spät oder zu teuer fertig und lieferten aus Anwendersicht nicht die gewünschten Ergebnisse.

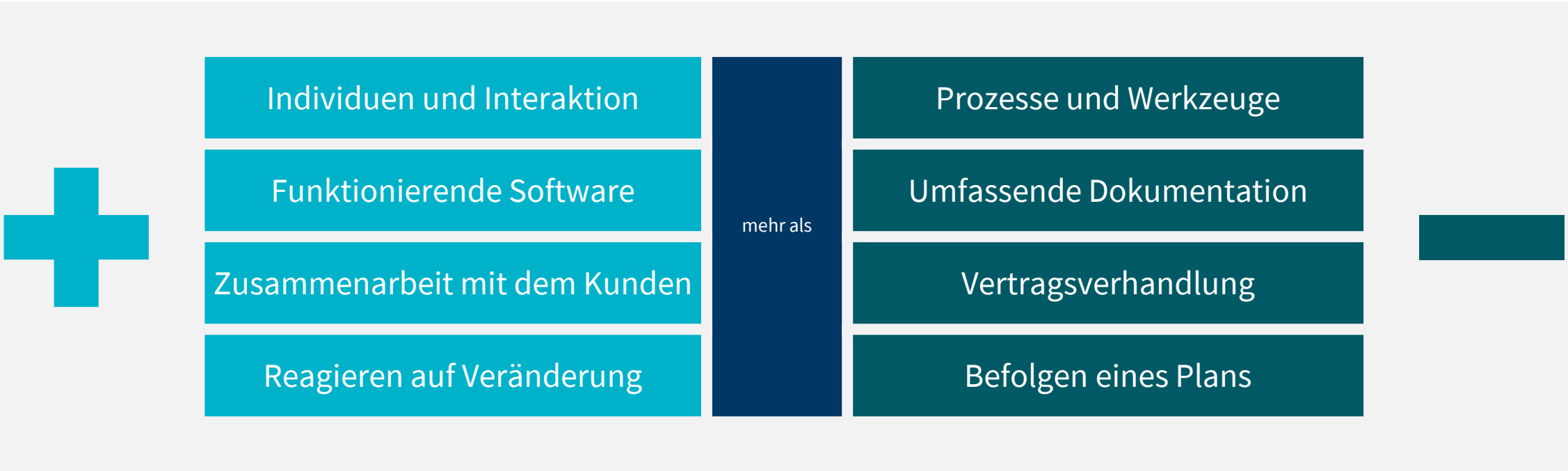


- Welchen Zustand haben wir heute?

„Chaos Report“ der Standish-Group 1994 über 8.000 abgewickelte IT-Projekte in den USA.
Dieser Report wird seitdem jährlich veröffentlicht.

Die vier Werte

- *„Wir erschließen bessere Wege, Software zu entwickeln, indem wir es selbst tun und anderen dabei helfen. Durch diese Tätigkeit haben wir diese Werte zu schätzen gelernt...“*
  (Agile Manifesto)



Die zwölf Prinzipien (1/2)

1. Unsere höchste Priorität ist es, den Kunden durch frühe und kontinuierliche Auslieferung wertvoller Software zufrieden zu stellen.
2. Heiße Anforderungsänderungen selbst spät in der Entwicklung willkommen. Agile Prozesse nutzen Veränderungen zum Wettbewerbsvorteil des Kunden.
3. Liefere funktionierende Software regelmäßig innerhalb weniger Wochen oder Monate und bevorzuge dabei die kürzere Zeitspanne.
4. Fachexperten und Entwickler müssen während des Projektes täglich zusammenarbeiten.
5. Errichte Projekte rund um motivierte Individuen. Gib ihnen das Umfeld und die Unter-stützung, die sie benötigen und vertraue darauf, dass sie die Aufgabe erledigen.
6. Die effizienteste und effektivste Methode, Informationen an und innerhalb eines Entwicklungsteams zu übermitteln, ist im Gespräch von Angesicht zu Angesicht.

Die zwölf Prinzipien (2/2)

7 Funktionierende Software ist das wichtigste Fortschrittsmaß.

8 Agile Prozesse fördern nachhaltige Entwicklung. Die Auftraggeber, Entwickler und Benutzer sollten ein gleichmäßiges Tempo auf unbegrenzte Zeit halten können.

9 Ständiges Augenmerk auf technische Exzellenz und gutes Design fördert Agilität.

10 Einfachheit - die Kunst, die Menge nicht getaner Arbeit zu maximieren - ist essenziell.

11 Die besten Architekturen, Anforderungen und Entwürfe entstehen durch selbstorganisierte Teams.

12 In regelmäßigen Abständen reflektiert das Team, wie es effektiver werden kann und passt sein Verhalten entsprechend an.

2. Agiles Mindset

VUCA-World - Warum Agile wichtig geworden ist

Volatility

- Volatilität, Flüchtigkeit, Unbeständigkeit, Schwankungen

Uncertainty

- Unsicherheit, Ungewissheit, Verunsicherung, Unbestimmtheit

Complexity

- Komplexität, Vielschichtigkeit, Komplexe Strukturen, Abhängigkeiten

Ambiguity

- Ambiguität, Mehrdeutigkeit, Zweideutigkeit, Ambivalenz

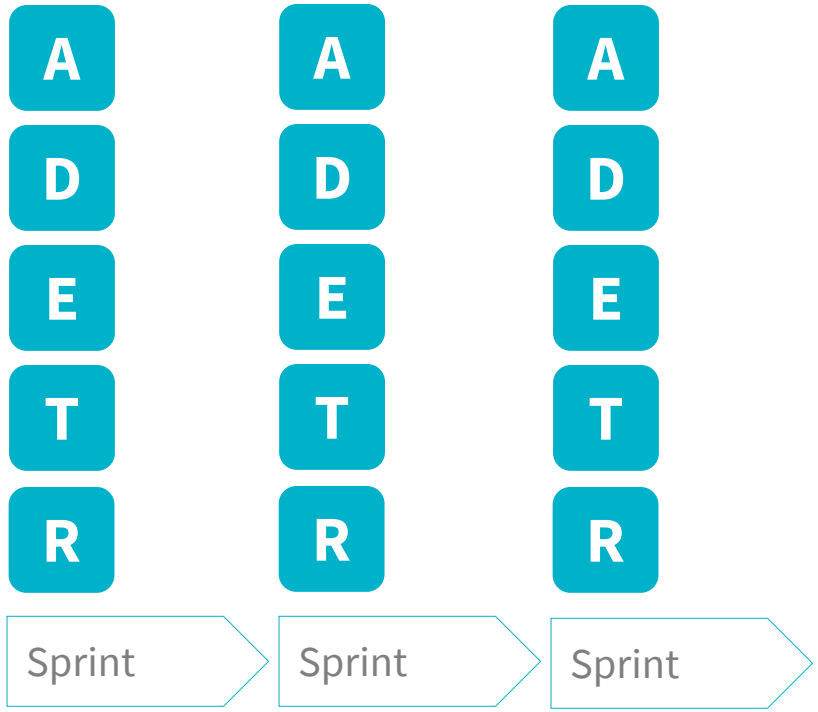
Klassisch vs. agil

Klassisch (Wasserfall) versus …

- **A** Analyse
- **D** Design
- **E** Entwicklung
- **T** Test
- **R** Rollout

- Anforderungen sind klar und definiert
- Lange Planungsphasen
- Änderungen aufwändig formalisiert

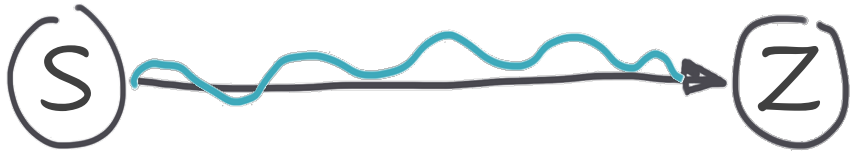
Agil



- Iterativ & inkrementell
- Risikominimierung
- Frühzeitige Integration von Änderungen
- Detailplanung nur für nächsten Sprint/Zyklus

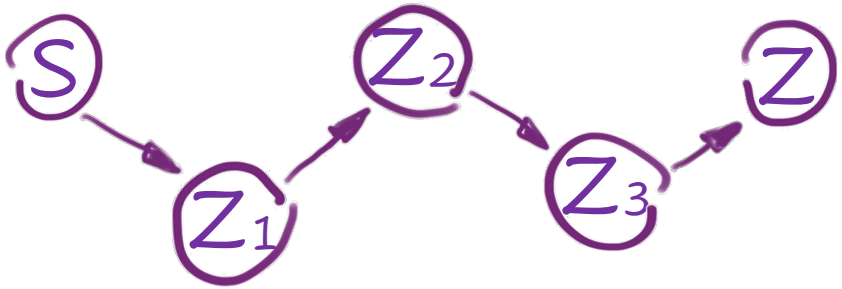
Prozessteuerung

Definierte Prozesssteuerung



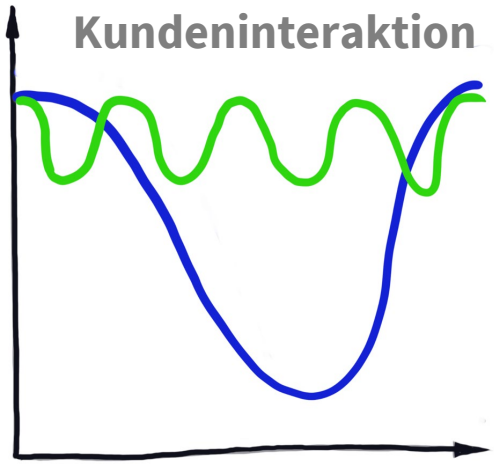
- Prozess mit allen Zwischenschritten im Vorfeld definierbar
- Kontrolle durch Soll-Ist-Vergleich
- gleiche Ergebnisse reproduzierbar

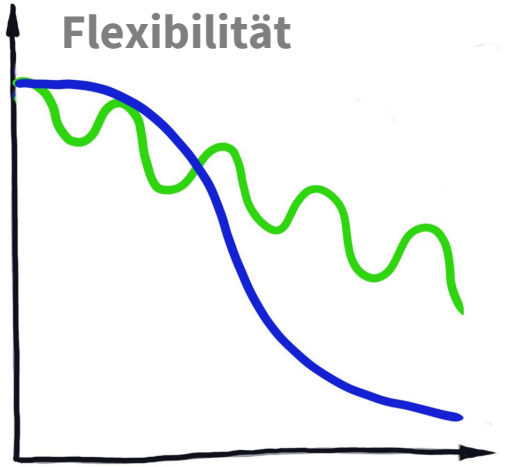
Empirische Prozesssteuerung

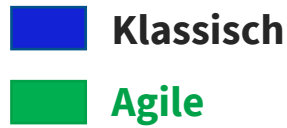


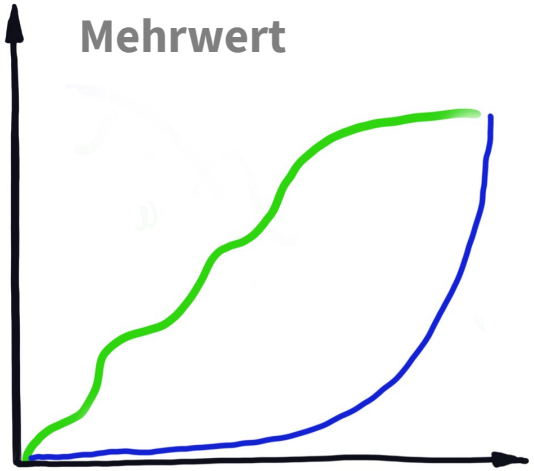
- Erwartet das Unerwartete
- Ergebnisse und Prozess nicht vollständig definiert
- Kontrolle durch häufige Inspektion des ISTs

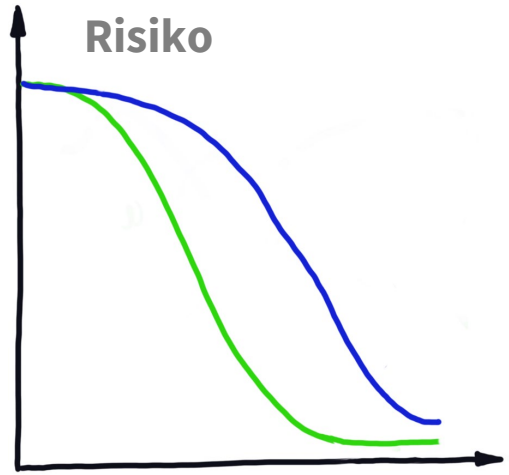
Kundeninteraktion, Flexibilität, Mehrwert und Risiko











Agilometer

KLASSISCH

- Flexibilität bzgl. des Liefergegenstandes
- Niveau der Zusammenarbeit, kulturelle Umgebung
- Einfachheit der Kommunikation
- Fähigkeit der iterativen Vorgehensweise und stufenweiser Lieferung
- Vorteilhafte Umgebungsbedingungen

AGILE

Charakteristik – klassisch und agil

Charakteristik	Klassisch	Agil
Frameworks/Methoden (Beispiele)	PRINCE2® (AXELOS), PMBOK® (PMI), ICB® (IPMA Competence Baseline), V-Modell XT® (Behörden in D), HERMES® (Behörden in CH)	Scrum, Kanban, Extreme Programming® (XP), Crystal Family®
Anwendbarkeit	Komplexe Projekte mit klar definierbarem Ziel	Komplexe Projekte mit visionärem Ziel
Bezeichnung	Methodik	Framework
Zielsetzung	Produkt(e)	Produkt(e), Release(s)
Voraussetzungen	Business Case, Steuerungsgrößen	Definition of Done
Fortschrittsmaßstab	Phase, Meilenstein, Arbeitspaket	Iteration (Timebox), (Produkt)Reviews
Änderungsverfahren	Change Management Prozess (Change Authority, ggf. mit Eskalation)	Product Backlog-Refinement, Daily Standup, Sprint Planning
Nutzenerbringung	Nach Projektübergabe/Projektabnahme (selten schon Teilnutzen während des Projekts erzielbar)	Mit jedem fertigen Inkrement

Worin besteht der Unterschied bzw. Zusammenhang?

Rahmenwerk

- Übergeordnete Struktur
- Prinzipien
- Leitlinien/-planken
- Regelwerk
- beinhaltet Methodik
- beinhaltet Technik
- „Was?“

Landkarte

Methodik

- Vorgehensweise
- Verfahren
- Prozesse
- „Werkzeugkoffer“
- Plan
- „Wie?“

Weg(e)

Technik

- Werkzeuge, Tools
- Praktische „Kunst“
- „Womit?“

Fortbewegungs-mittel

Definition „agile“ und „Agile“

- **„agile“ mit kleinem „a“ (Adjektiv)**
  - Fähigkeit, sich schnell und flexibel zu bewegen, schnell zu denken, zu verstehen und zu reagieren. Das Management einer Organisation muss einen Kulturwandel einleiten und Führungsstrukturen anpassen.
- **„Agile“ mit großem „A“ (Substantiv)**
  - Einführung einer Methode bzw. Vorgehensweise für das Entwickeln von Produkten durch die Unterteilung von Aufgaben in kurze Arbeitszyklen (Timeboxes) sowie häufige Neubewertungen und Anpassungen auf Produkt- und auf Prozessebene (Empirie).

  - Agile Projektmanagementansätze
    - eignen sich für „empirische“ Projekte in komplexen Umgebungen (hohe Anzahl von Variablen und unbekannten Faktoren).
    - betonen das Konzept des „fortlaufenden Lernens“ (Discovery) als Folge von Experimenten und stellen diese in den Mittelpunkt der Projektdurchführung.

3. Scrum Grundlagen

Definition

- Die Erfinder über Scrum:

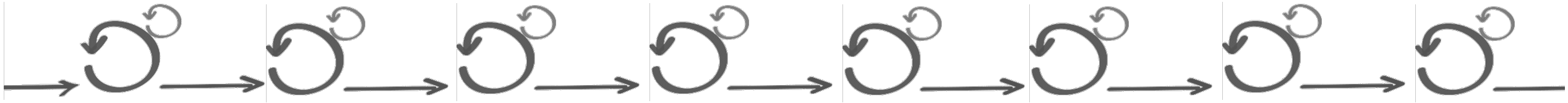
„It is lightweight, simple to understand but so difficult to master.“

„Scrum ist ein leichtgewichtiges Rahmenwerk, welches Menschen, Teams und Organisationen hilft, Wert durch adaptive Lösungen für komplexe Probleme zu generieren.“

Aktivitäten

Der Scrum Master fördert ein Umfeld, in dem folgende Aktivitäten stattfinden:

- Ein **Product Owner** ordnet die Arbeit für ein komplexes Produkt/Problem in ein Product Backlog ein
- Das **Scrum Team** erzeugt aus einer Auswahl dieser Arbeit innerhalb eines Sprints ein wertvolles Inkrement
- Das Scrum Team und dessen **Stakeholder** überprüfen die Ergebnisse am Ende des Sprints und passen sie für den nächsten Sprint an
- Diese Schritte werden wiederholt



Basis der Scrum-Theorie

Scrum-Theorie basiert auf

- auf **Empirie**: Wissen wird aus Erfahrung und Beobachtungen gewonnen und liefert somit die Grundlage für Entscheidungen
- auf **Lean Thinking**:
  reduziert Verschwendung und fokussiert auf das Wesentliche
- Scrum verwendet einen **iterativen, inkrementellen** Ansatz zur Optimierung der Vorhersagbarkeit und zur Risikokontrolle.
- Scrum kombiniert…
  - …**vier formale Events** zur Überprüfung und Anpassung innerhalb eines weiteren, umspannenden Events – dem Sprint.
  - Durch diese Events werden die drei empirischen Scrum-Säulen **Transparenz, Überprüfung und Anpassung** ermöglicht und implementiert.

Die drei Säulen von Scrum (1/4)

Die drei Säulen von Scrum (2/4)

Transparenz

- Der zu entwickelnde Prozess,
  die Aufgaben und
  die Ergebnisse müssen
  für die Entwickler als auch
  für die Kunden sichtbar sein.

Die drei Säulen von Scrum (3/4)

Überprüfung

- …der Scrum-Artefakte und des Fortschritts (um Abweichungen oder Probleme rechtzeitig zu erkennen).
- Fünf Events, um Anpassungen einzuleiten.

Die drei Säulen von Scrum (3/4)

Anpassung

- …als Folge der Überprüfung.
- Ist sehr anspruchsvoll, weil aus „kontinuierlicher Veränderung“ folgt: schnellstmöglich umsetzen.

Werte

Commitment

Verpflichtung, Ziele zu erreichen und sich gegenseitig zu unterstützen.

Fokus

Fokus auf den bestmöglichen Fortschritt der Arbeit in einem Sprint, gemeinsames Verständnis für das Ziel.

Offenheit

Offenheit bezüglich der zu erledigenden Aufgaben und der damit verbundenen Herausforderungen, grundsätzlicher kultureller Aspekte der Zusammenarbeit.

Respekt

Sich gegenseitig als fähige, unabhängige Personen respektieren (cross-functional), Respekt gegenüber dem Kunden.

Mut

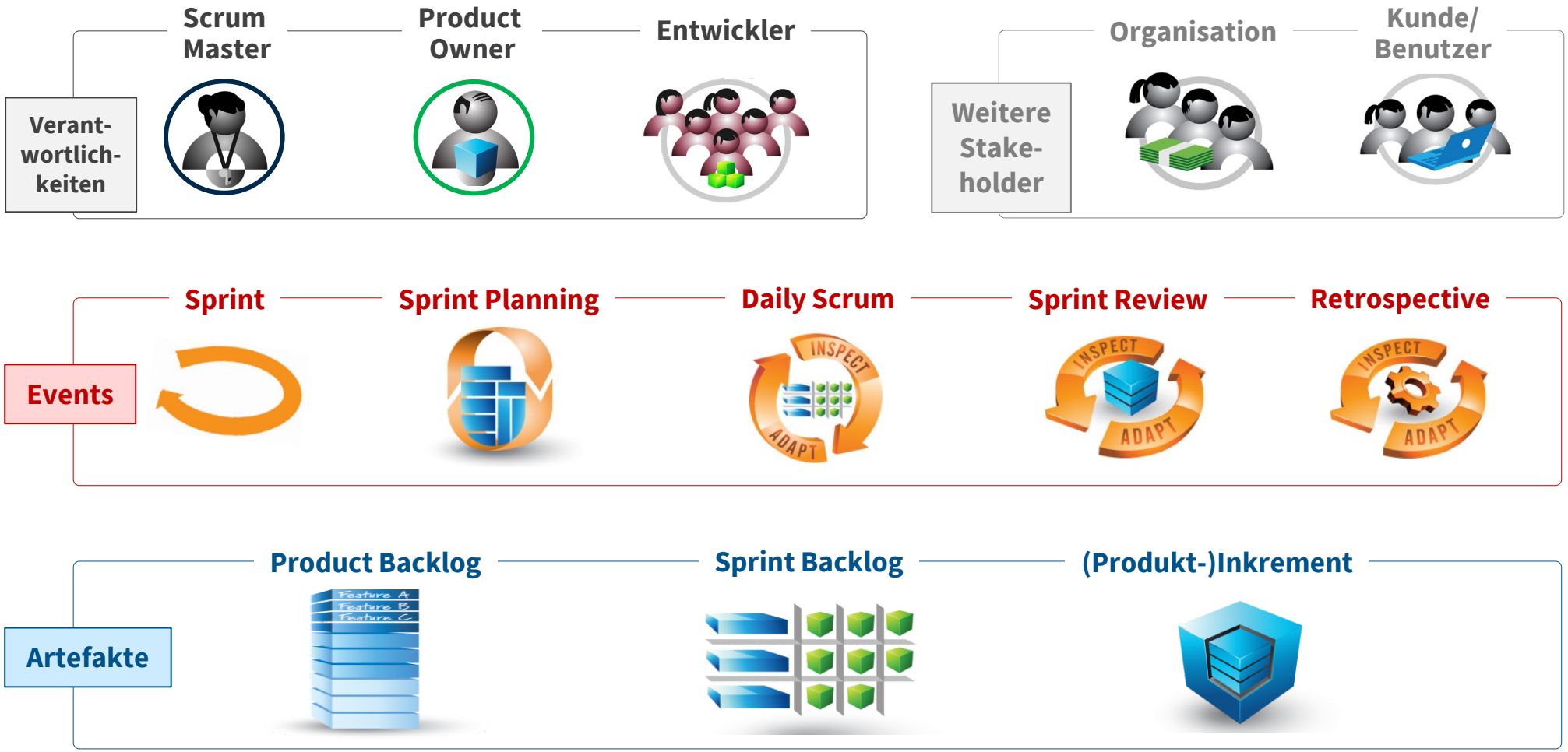
Mut an schwierigen Aufgaben zu arbeiten. Mut, Fehler zu machen, Risiken wegen Veränderungen einzugehen, Dinge in Frage zu stellen, über den Tellerrand zu schauen.

Säulen und Werte im Kontext

Wenn diese Werte durch das Scrum Team und die Menschen, mit denen es arbeitet, verkörpert werden,
werden die Scrum-Säulen
Transparenz, Überprüfung und Anpassung
lebendig und bauen ***Vertrauen*** *auf.*

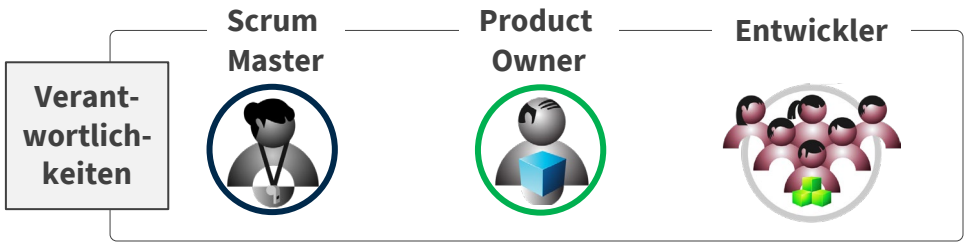
4. Scrum Framework

Überblick


Verant-
wortlich-
keiten
Scrum
Master
Product
Owner
Entwickler
Weitere
Stake-
holder
Organisation
Kunde/
Benutzer
Events
Sprint
Sprint Planning
Daily Scrum
Sprint Review
Retrospective
INSPECT
ADAPT
Artefakte
Product Backlog
Sprint Backlog
(Produkt-)Inkrement
Feature A
Feature B
Feature C

5. Scrum Team

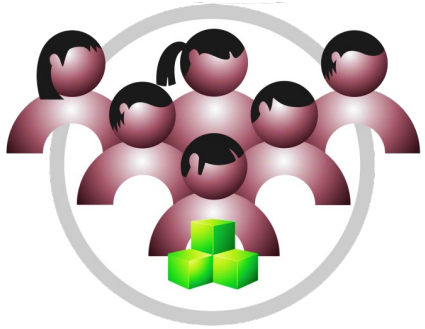
Das Scrum Team



- Zentraler Bestandteil von Scrum
- Keine Sub-Teams oder Hierarchien
- Konzentration auf das Produkt-Ziel
- Interdisziplinär: die Mitglieder verfügen über alle Fähigkeiten, die erforderlich sind, um in jedem Sprint Wert zu schaffen
- Organisiert sich selbst → entscheidet z.B. intern, wer was wann und wie macht
- Klein genug, um flink zu bleiben und groß genug, um Wert zu schaffen. Daumenregel: 10 oder weniger Mitglieder
- Umsetzungsverantwortlich (responsible)
- Ergebnisverantwortlich (accountable), in jedem Sprint ein wertvolles, nützliches Inkrement zu schaffen

Drei spezifische Ergebnisverantwortlichkeiten (1/3)

Entwickler



- Erstellen den Plan für den Sprint: Sprint Backlog
- Sichern die Qualität durch die Einhaltung einer Definition of Done (DoD)
- Passen täglich den Plan zur Erreichung des Sprint-Ziels an
- Handeln eigenverantwortlich und nehmen sich gegenseitig als Experten in die Verantwortung
- Sind als Entwicklerteam interdisziplinär (crossfunctional) aufgestellt und verfügen über alle notwendigen Fähigkeiten, um ein fertiges (done) Inkrement herzustellen

Was macht ein gutes Team aus?

Drei spezifische Ergebnisverantwortlichkeiten (2/3)

Product Owner

- Maximierung des Wertes des Produkts
- Verantwortlich für ein effektives Product-Backlog-Management (delegierbar, bleibt aber ergebnisverantwortlich):
  - Produkt-Ziel entwickeln und explizit kommunizieren
  - Product-Backlog-Einträge erstellen und eindeutig kommunizieren
  - Reihenfolge der Product-Backlog-Einträge festlegen
- Stellt sicher, dass das Product Backlog transparent, sichtbar und verstanden ist
- Nur eine Person, kein Gremium
- Änderungen am Product Backlog nimmt nur Product Owner vor

Was versteht man eigentlich als „Wert“?

Drei spezifische Ergebnisverantwortlichkeiten (3/3)

Scrum Master (1/3)

- Ergebnisverantwortlich für die Einführung und das Verständnis von Scrum (innerhalb des Scrum Teams als auch in der Organisation)
- Servant Leadership:
  - Teammitglieder in Selbstmanagement und interdisziplinärer Zusammenarbeit coachen
  - Scrum Team bei der Fokussierung auf hochwertige Inkremente unterstützen (DoD!)
  - Hindernisse beseitigen
  - Sicherstellen, dass alle Events positiv und produktiv innerhalb der Timebox stattfinden

Drei spezifische Ergebnisverantwortlichkeiten (3/3)

Scrum Master (2/3)

- Unterstützung des Product Owner:
  - Vermittelt Techniken zur effektiven Definition des Produkt-Ziels und zum Product-Backlog-Management
  - Hilft dem Scrum-Team, die Notwendigkeit klarer und präziser Product-Backlog-Einträge zu verstehen
  - Unterstützt bei einer empirischen Produktplanung
  - Fördert die Zusammenarbeit mit Stakeholdern

Drei spezifische Ergebnisverantwortlichkeiten (3/3)

Scrum Master (3/3)

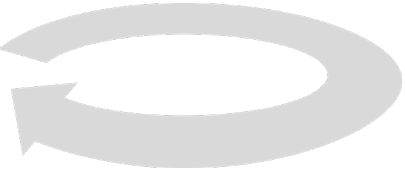
- Unterstützung der Organisation:
  - Empfiehlt der Organisation die Einführung von Scrum und unterstützt sie in der Planung
  - Führt, schult und coacht bei der Einführung von Scrum
  - Erzeugt bei Mitarbeitenden ein Verständnis für einen empirischen Ansatz in komplexen Aufgaben
  - Beseitigt Barrieren zwischen Stakeholdern und Scrum Teams

6. Scrum Events

Scrum Events

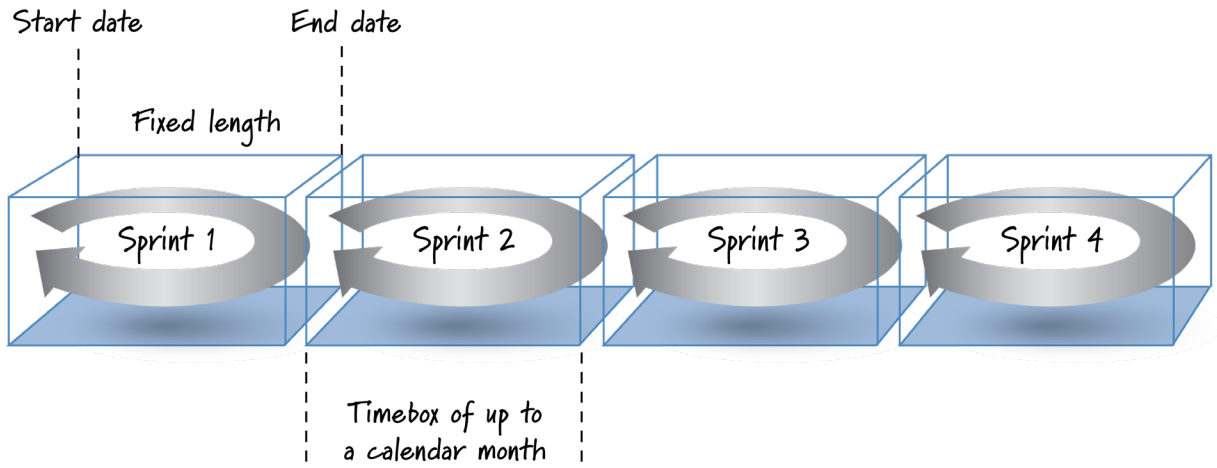
- Sind formale Gelegenheiten, Scrum-Artefakte zu überprüfen und anzupassen.
- Ermöglichen die notwendige Transparenz.
- Schaffen Regelmäßigkeit (Rituale).
- Reduzieren Meetings, die in Scrum nicht definiert sind.
- Finden zu festen Zeiten am selben Ort statt (Komplexität vermeiden).

Sprint (1/2)

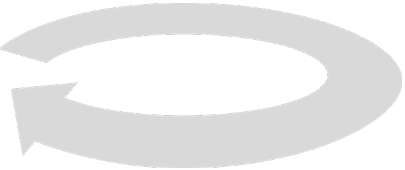


- Container für alle anderen Events, d.h. alle Arbeiten, die notwendig sind, um das Produkt-Ziel zu erreichen, einschließlich Sprint Planning, Daily Scrum, Sprint Review und Sprint Retrospective
- Herzschlag von Scrum (Ideen werden in Wert umgewandelt)
- Feste Länge von einem Monat oder weniger (ein neuer Sprint beginnt unmittelbar nach dem Abschluss des vorherigen Sprints)
- Die Festlegung der Sprintdauer kann verändert werden, wenn es dafür triftige Gründe gibt.
  Keine willkürliche Veränderung!

Zu welchem Zeitpunkt in einer Kalenderwoche könnte ein neuer Sprint beginnen?



Sprint (2/2)



- Während des Sprints:
  - Werden keine Änderungen vorgenommen, die das Sprint-Ziel gefährden
  - Nimmt die Qualität nicht ab
  - Wird das Product Backlog bei Bedarf verfeinert (Refinement)
- Ein Sprint kann abgebrochen werden, wenn das Sprint-Ziel obsolet wird (nur durch Product Owner!).

- Ist der Horizont eines Sprints zu weit entfernt, kann das Sprint-Ziel hinfällig werden, die Komplexität kann steigen und das Risiko kann zunehmen. Kürzere Sprints werden eingesetzt, um mehr Lernzyklen zu generieren und das Risiko von Kosten und Aufwand auf einen kleineren Zeitrahmen zu begrenzen.

Sprint Planning

- Der Product Owner stellt eine gute Vorbereitung sicher (Verständnis der Product-Backlog-Einträge).
- Regelmäßiges Refinement durch das ganze Team schafft die Voraussetzung für ein effektives Sprint Planning.
- Das gesamte Scrum Team initiiert den Sprint durch die Auswahl der zu erledigenden Aufgaben (auch Stakeholder können beratend einbezogen werden).

max. 8 Stunden (2 Std./Sprintwoche)

3 Themen des Sprint Planning

1. **Warum** ist dieser Sprint wertvoll?

   Wie können wir den Wert des Produkts steigern und daraus das Sprint-Ziel definieren. Diese Definition muss vor dem Ende des Sprint Planning finalisiert sein.

2. **Was** kann in diesem Sprint abgeschlossen (Done) werden?

   Im Dialog mit dem Product Owner wählen die Entwickler aus dem Product Backlog für den aktuellen Sprint Einträge aus. Diese können vom Scrum Team verfeinert werden.

3. **Wie** wird die ausgewählte Arbeit erledigt?

   Planung der Arbeit der ausgewählten Product-Backlog-Einträge (oft durch Zerlegung von Product-Backlog-Einträgen in kleinere Arbeitseinheiten von einem Tag oder weniger)

   Das Sprint-Ziel, die ausgewählten Product-Backlog-Einträge und der Plan für deren Lieferung werden zusammenfassend als Sprint Backlog bezeichnet.

Daily Scrum

- Überprüfung des Fortschritts bezüglich des Sprint-Ziels
- Sprint Backlog bei Bedarf anpassen
- 15-minütiges Event für die Entwickler des Scrum Teams
- An jedem Arbeitstag des Sprints zur gleichen Zeit und am gleichen Ort
- Verbessert die Kommunikation kontinuierlich
- Identifizieren von Hindernissen
- Fördert die schnelle Entscheidungsfindung
- Verringert die Notwendigkeit anderer Meetings

max. 15 Minuten

Sprint Review

- Überprüfung des Ergebnisses des Sprints und Anpassungen festlegen:
  - Scrum Team stellt die Ergebnisse den wichtigsten Stakeholdern vor
  - Vorgestellt wird nur, was zu 100% der Definition of Done entspricht (99% ist nicht „Done“)
  - Fortschritt bezüglich des Produkt-Ziels wird besprochen
  - Festlegen, was als Nächstes zu tun ist
  - Modifizierung des Product Backlogs (wenn sinnvoll)

  (Häufig läd der Product Owner die Stakeholder ein)

max. 4 Stunden (1 Std./Sprintwoche)

Sprint Retrospective

- Planung der Steigerung von Qualität und Effektivität:
  - Scrum Team überprüft, wie der letzte Sprint in Bezug auf Individuen, Interaktionen, Prozesse, Werkzeuge und seine Definition of Done verlief
  - Was ist im Sprint gut gelaufen?
  - Welche Probleme gab es und wie wurden diese Probleme gelöst (oder auch nicht)?
- Verbesserungen werden priorisiert und deren Umsetzung zeitnah geplant (möglicherweise als Sprint Backlog Eintrag)
- Abschluss des Sprints

Die K.A.L.M. Methode (Keep, Add, Less, More) wird für die visuelle Darstellung der Verbesserungen genutzt

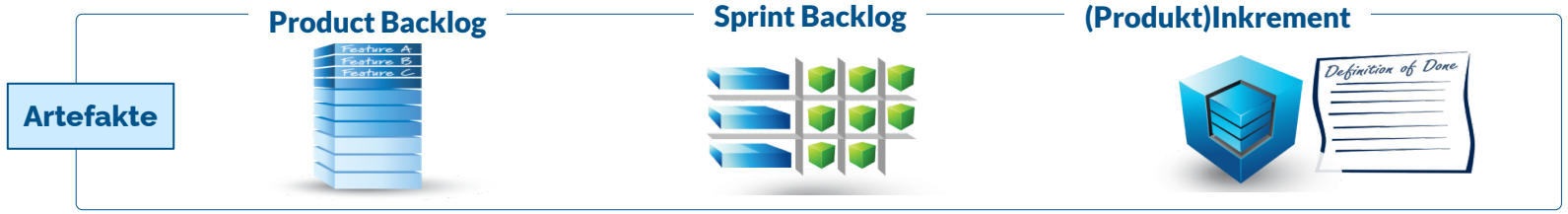
max. 3 Stunden (45 Min./Sprintwoche)

Events im zeitlichen Ablauf

- Sprint — Ereigniscontainer (max. 1 Monat)
- Sprint Planning — Drei Fragen … Warum? Was? Wie? (max. 8 Std.)
- Daily Scrum — Tägliche Überprüfung des Fortschritts (max. 15 Min.)
- Sprint Review — Überprüfung der Sprintergebnisse (max. 4 Std.)
- Sprint Retrospective — Überprüfung der Zusammenarbeit und Planung von Verbesserungen (max. 3 Std.)

7. Scrum Artefakte

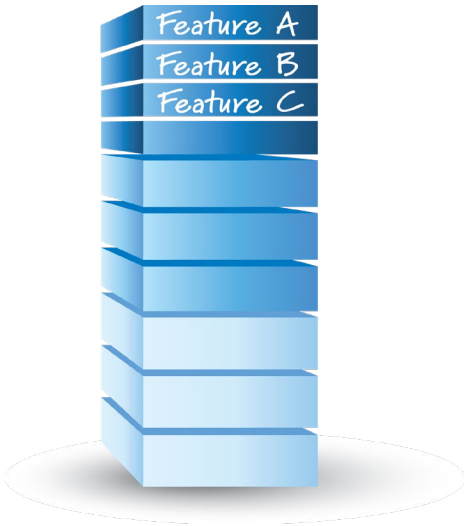
Jedes Artefakt beinhaltet ein Commitment



- Product Backlog → Produkt-Ziel
- Sprint Backlog → Sprint-Ziel
- Inkrement → Definition of Done

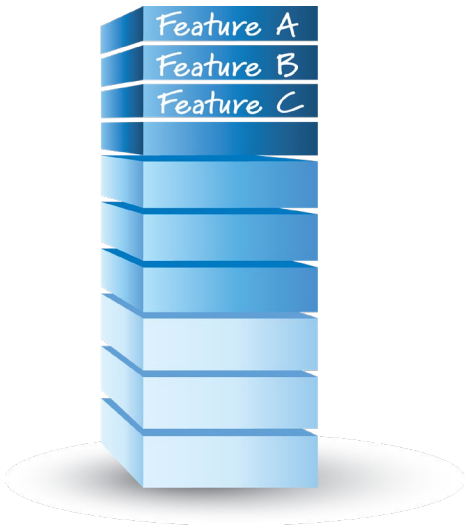
- Die Commitments dienen dazu, Empirie und die Scrum-Werte für das Scrum Team und seine Stakeholder zu verstärken.

Product Backlog



- Beschreibt in Listenform, was das Produkt aus Sicht des Kunden/Product Owner leisten muss.
  - Geordnete Liste als einzige Quelle der Arbeit für das Scrum Team.
  - Product-Backlog-Einträge (meist User Stories) werden durch das Refinement in kleinere, präzisere Elemente zerlegt.
    Sie enthalten mindestens
    - Eine Beschreibung,
    - Eine Angabe zur Reihenfolge,
    - Eine Größe (Komplexitätsschätzung).
  - Für die Größenbestimmungen sind die Entwickler umsetzungsverantwortlich, der Product Owner unterstützt.
  - Die Reihenfolge entsteht durch den Wert der Einträge.
    Dabei werden Nutzen, Kosten, Größe, Abhängigkeiten und Risiko etc. betrachtet.

Product Backlog | Was gehört in die Liste?



- Alle Einträge, die nötig sind, um die Anforderungen an das Produkt zu verstehen und es herzustellen.
- Einträge werden auch Product Backlog Items (PBIs) genannt.
- Unterscheidung zwischen funktionalen und nicht-funktionalen Anforderungen.
  - Funktional: Führen direkt zu einer Funktion und erfüllen eine Anforderung an das Produkt.
  - Nicht-Funktional: Werden benötigt, um die funktionalen Aufgaben erfüllen zu können, z.B. Einrichten einer Entwicklungs-Umgebung.
- Fehler, die zu beheben sind.

Backlog Items werden häufig als User Story formuliert, also aus Sicht des Nutzers.

Product Backlog | Die User Story

- Die User Story ist eine sehr kurze und in Alltagssprache formulierte Beschreibung einer Anforderung. Die Form ist festgelegt: WER – WAS – WARUM.

  Als ***[Rolle]*** möchte ich ***[Eigenschaft/Funktion]*** , damit ***[Nutzen]*** .

- User Stories werden mit weiteren Details auf einer Story Card festgehalten.
- Sie können ergänzt werden durch Akzeptanzkriterien, um die Story zu präzisieren. Diese Kriterien müssen erfüllt sein, damit die Story als realisiert gilt.
- Zwei Regeln: ***Nicht-technisch*** und ***Unabhängig***.

Nr.	Prio	Thema	User Story / Anforderung	Akzeptanzkriterien	Story Points
1	M	Login	Als Kunde möchte ich mich anmelden können, um eine Bestellung aufzugeben.	- Eingabe der Zugangsdaten möglich - Überprüfung der Zugangsdaten - Zugangsdaten richtig -> Zugang erlaubt - Zugangsdaten falsch -> Zugang verweigert	4
2	M	Login	Als Interessent möchte ich Zugang zum Katalog haben, damit ich Angebote recherchieren kann.	- Keine Eingabe von personenbezogenen Daten nötig - Keine Bestellung möglich	2

Product Backlog | INVEST

- Wann ist ein PBI (Product Backlog Item) bereit für die Übernahme in den Sprint? Anforderungen werden nach INVEST-Kriterien erfasst:

I
- **Independent** – voneinander unabhängig

N
- **Negotiable** – verhandelbar

V
- **Valuable** – wertvoll

E
- **Estimable** – schätzbar

S
- **Small** oder **Sized appropriately** – angemessen klein/groß

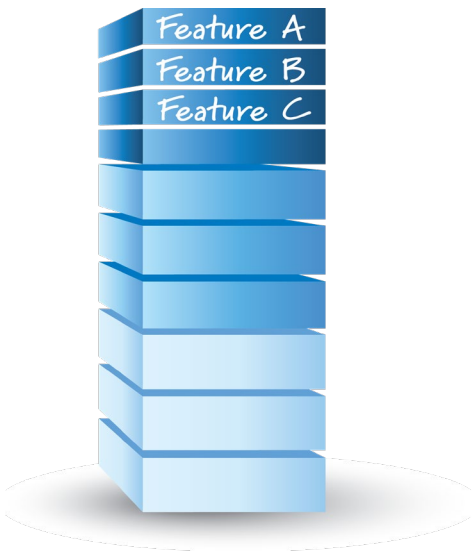
T
- **Testable** – überprüfbar

MoSCoW-Priorisierung

- **M – Must have**
  - Feature muss entwickelt werden, da das Produkt sonst nicht nutzbar sein wird.
- **S – Should have**
  - Feature sollte entwickelt werden, da sonst Probleme in der Produktnutzung auftreten werden.
- **C – Could have (Nice to have)**
  - Feature kann entwickelt werden, wenn „Must have“- und „Should have“-Features erledigt sind.
- **W - Won‘t have**
  - Feature ist nützlich, aber nicht dringlich. Seine Entwicklung wird deshalb ggf. auf einen späteren Zeitpunkt verschoben.

MoSCoW ist in der DSDM-Methode von zentraler Bedeutung!
Höchstens 60% sollen Must Have-Features, mindestens 20% Could Have-Features sein.

Product Backlog | Commitment: Produkt-Ziel



Ein Produkt ist ein Instrument, um Wert zu liefern. Es hat klare Grenzen, bekannte Stakeholder, eindeutig definierte Benutzer oder Kunden. Ein Produkt kann eine Dienstleistung, ein physisches Produkt oder etwas Abstrakteres sein.

- Zukünftiger Zustand des Produkts ist das Planungsziel für das Scrum Team
- Das Produkt-Ziel wird im Produkt Backlog abgebildet
- Der Inhalt des Product Backlogs definiert, was das Produkt-Ziel erfüllen muss

Das Produkt-Ziel ist das langfristige Ziel für das Scrum Team. Das Scrum Team muss eine Zielvorgabe erfüllen (oder aufgeben), bevor es die nächste angeht.

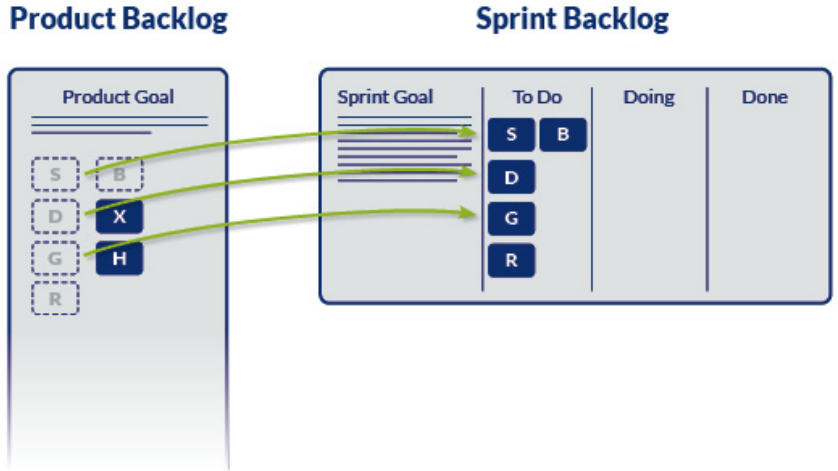
Sprint Backlog I Sprint-Ziel

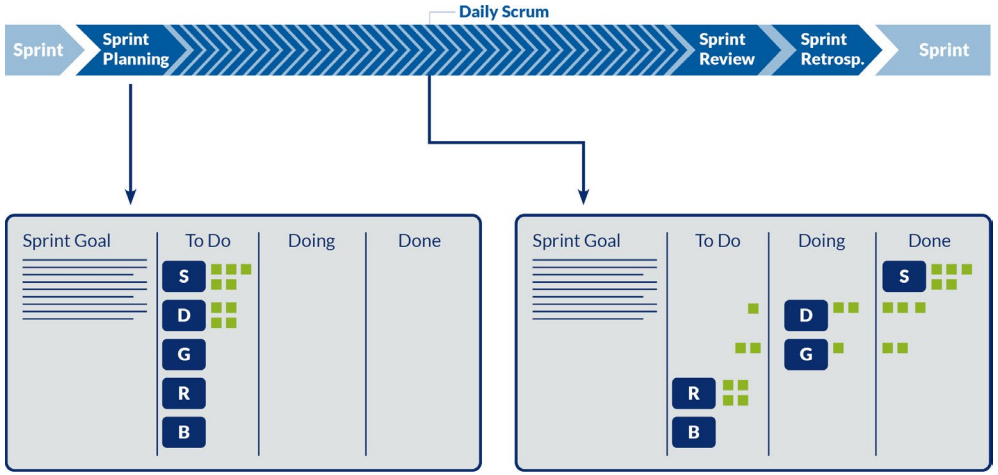
- **Sprint Backlog**
  - Enthält die für den Sprint ausgewählten Product-Backlog-Einträge (**was**).
  - Enthält einen umsetzbaren Plan für die Lieferung des Inkrements (**wie**).
  - Bildet das Sprint-Ziel ab (**wofür**).
  - Wird permanent aktualisiert durch empirisches Lernen.
  - Der Detailgrad muss der Fortschrittsüberwachung im Daily Scrum genügen.

- **Commitment: Sprint-Ziel**
  - Einzige Zielsetzung für den Sprint als ein Commitment der Entwickler
  - Wird während des Sprint-Planning-Events erstellt und im Sprint Backlog abgebildet.

Sprint Backlog | Aktivitäten

- Enthält die ausgewählten User Stories mit geschätzter Komplexität.
- Die Stories werden in Aufgaben (Tasks) zerlegt.
  Ideale Dauer eines Tasks: maximal ein Tag. *(Abb. Links)*
- Es müssen nicht gleich alle Stories in Tasks zerlegt werden,
  aber mindestens so viele, dass man einige Tage arbeiten kann,
  der Rest kann später erfolgen. Auch hier ist Empirie wichtig. *(Abb. Rechts)*





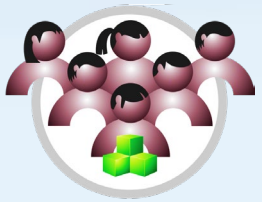
Inkrement I Definition of Done

- **Inkrement**
  - Konkreter, additiver Schritt in Richtung des Produkt-Ziels
  - Um einen (Mehr)Wert zu erzielen, muss das Inkrement verwendbar sein
- **Commitment: Definition of Done (DoD)**
  - Formale Beschreibung des Zustands des Inkrements, wenn es die für das Produkt erforderlichen Qualitätsmaßnahmen erfüllt.
  - DoD schafft Transparenz und ein gemeinsames Verständnis darüber, wann eine Arbeit fertiggestellt ist.
  - Entspricht ein Product-Backlog-Eintrag nicht der DoD, kann er weder „released“ noch beim Sprint Review präsentiert werden, er wandert zurück in das Product Backlog.
  - Arbeiten mehrere Scrum Teams an einem Produkt zusammen, müssen sie eine gemeinsame DoD definieren und sich daran halten.

Scrum Prozess

Input von Endnutzern, Kunden, Scrum-Team und anderen Stakeholdern

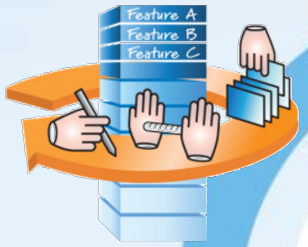
Product Owner



Entwickler

Scrum Master

Verfeinerung



Daily Scrum

Entwicklung Inkrement(e) und Verfeinerung Backlog(s)

Daily Scrum 15 Minuten/Tag

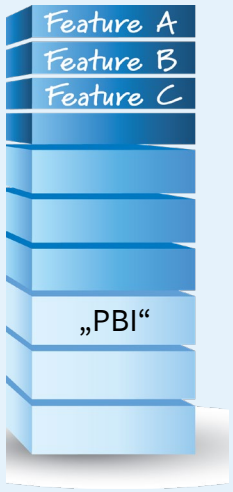
Sprint Review Max. 4 Std. bei 1 Monat

Sprint Planning
(Why, What, How)

Entwickler ziehen PBIs vom Product Backlog zum Sprint Backlog

Sprint Backlog

Nutzbares Inkrement Def. of Done



Product Backlog

Max. 8 Std. bei 1 Monat

SPRINT
(max. 1 Monat)

keine Änderungen der Dauer oder des Ziels, keine Minderung der Qualität

Sprint Retrospective

Max. 3 Std. bei 1 Monat

Nachtrag zu den fünf Werten

Es gibt keine definierten Prinzipien und Methoden in Scrum.
Wichtig als Ableitung aus dem Agilen Manifest sind jedoch…

Empirie

- Wird realisiert durch den iterativ-inkrementellen Entwicklungsprozess – „Trial and Error“ / „Inspect and Adapt“

Selbstorganisation

- Ist Ausdruck der eigenen Verantwortung und stärkt die Motivation des Teams.

Priorisierung

- Dadurch wird früh maximaler Wert erzeugt und die Empirie unterstützt.

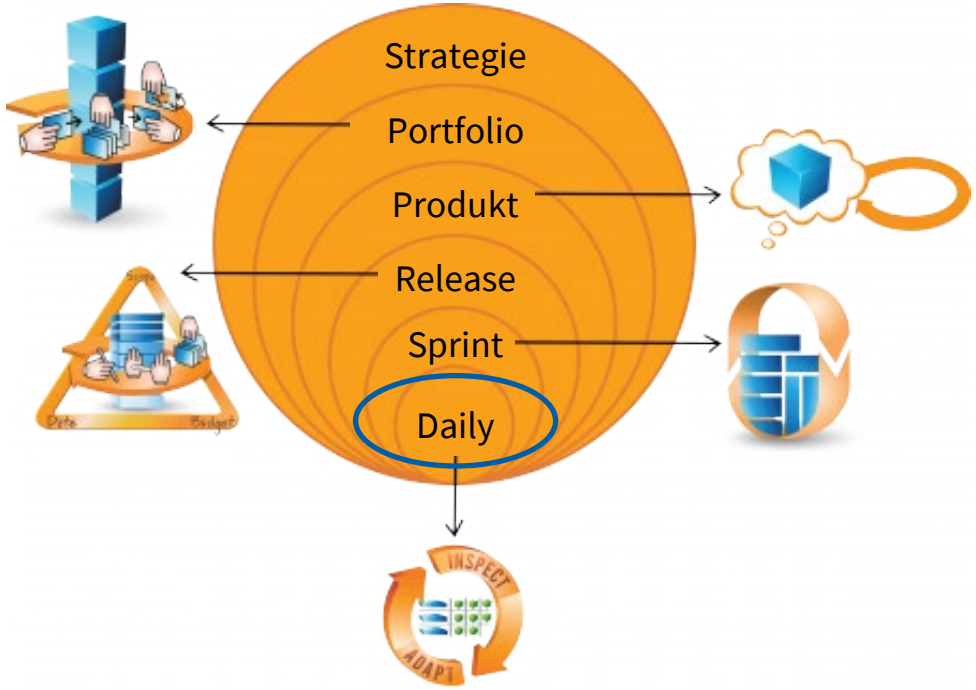
Timeboxing

- Kleine Zeiteinheiten erleichtern die Arbeitsorganisation. Der entstehende Rhythmus unterstützt effiziente Entwicklung.

8. Planen
Vom Groben bis zum Detail

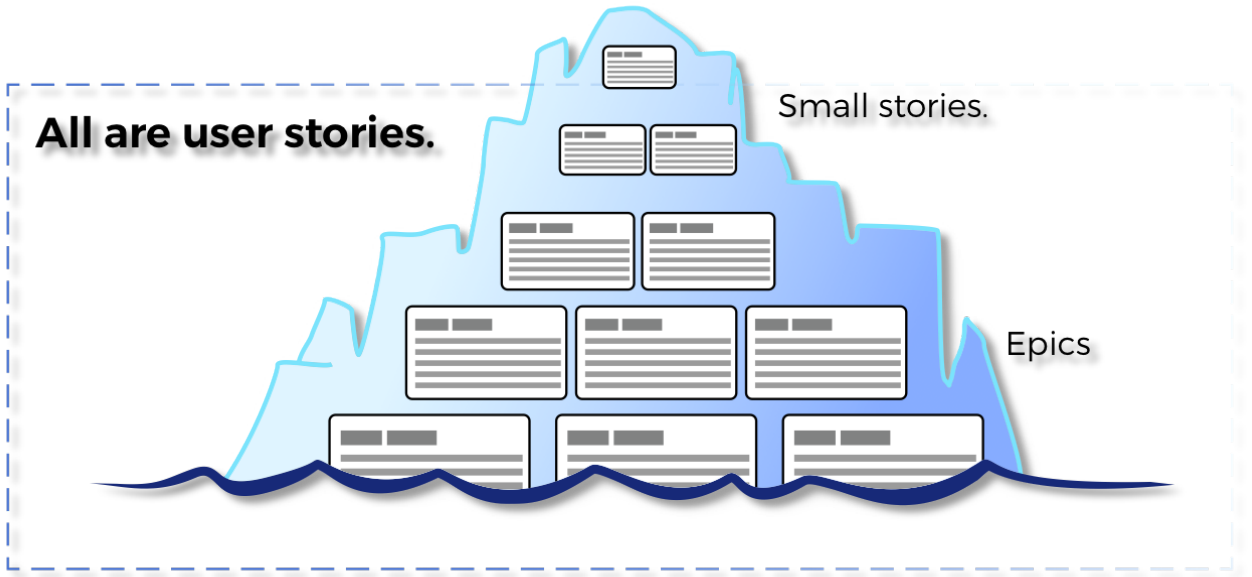
Planning Onion

- Strategie und Portfolio-Planung laufen außerhalb der Projektebene.
- Strategie-Ebene definiert die Ziele des Unternehmens.
- Portfolio legt die Projekte und Produkt-bereiche fest, mit denen die Ziele erreicht werden sollen.
- Die Produktplanung kombiniert die High-Level Produktvision mit einem Set von Stories für das PB.
- Für Scrum ist kein „Release-Planning“ vorgesehen. Denn klassisches Release-Planning würde die agilen Effekte der Entwicklung schmälern.
  Je nach Methode für die Skalierung eigene Lösungen.



Das Backlog enthält unterschiedlich fein detaillierte User Stories

- Kleine User Stories sind detailliert und gut verstanden.
- Größere Stories beinhalten noch Spielraum für tieferes Verständnis.
- Epics umreißen Ideen, sind aber nicht gut im Detail verstanden.
- Vorteile:
  - Während des Projekts kann sich vieles ändern. Eine detaillierte Planung für ***alles*** wäre verschwendete Energie.
  - Die Erfahrung aus realisierten Stories kann mit einfließen.
  - Das Backlog bleibt übersichtlich.



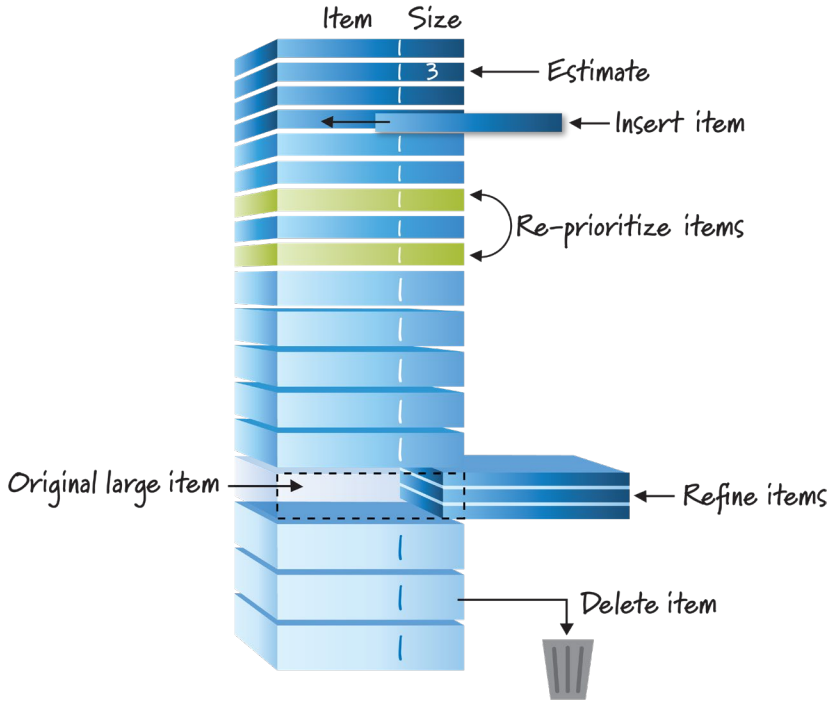
Gibt es einen Unterschied zwischen Epic, User Story, Thema und Feature?

- Scrum kennt nur **Product Backlog Items (PBIs)**
- PBIs können sein: Anforderungen, Fehler (Bugs), technische Aufgaben
- **Epic** ist eine große User Story, die einen oder mehrere Monate dauert.
  Epics werden in kleinere User Stories zerlegt, die in einen Sprint passen.
- Ein Thema ist eine Sammlung verwandter User Stories.
  JIRA: Epics werden genutzt, um **Stories** zu gruppieren.
- Der Begriff **Feature** ist unscharf und wird in der Praxis unterschiedlich verstanden.
  IceScrum: Features werden genutzt, um Stories zu gruppieren.
- Häufiges Verständnis von Feature: Beschreibung einer Produkteigenschaft, die eine oder mehrere Anforderungen erfüllt und vorteilhaft oder nützlich ist.

Gewissheit: Wie auch immer Einträge im Product Backlog heißen, sie müssen INVEST sein!

Das Meeting für die permanente Pflege des Backlogs

- Das Meeting hat keinen festen Zeitpunkt innerhalb des Sprints. Das Scrum Team legt Regeln und Zeitpunkte fest. Es hat keine Timebox.
- Faustregel: ca. 10% der Projektdauer.
- Anlässe:
  - Neue Ideen des Kunden/PO.
  - Detaillierung von Epics/Stories.
  - Änderung der Reihenfolge.
  - Entfernen von Einträgen.
  - Neuschätzung.
- Product Owner und Entwickler tragen die Durchführungsverantwortung.



In Echtzeit schätzen ist bei agilen Projekten kaum möglich

- Wir wissen nicht, wer an welcher Aufgabe arbeitet.
- Die Entwickler in einem interdisziplinären Team sind unterschiedlich schnell.
- Wir wissen zu Beginn zu wenig über die Anforderungen
  und wie sie sich entwickeln werden.
- Wir wissen nicht, wie viele Fehler gemacht werden, die Zeit kosten,
  aber auch wertvolle Erfahrung erzeugen.
- Eine Vorhersage in Echtzeit impliziert die Erwartung,
  dass es in dieser Zeit auch fertig wird.
- Wir müssen also ***abstrahieren***.

Abstrakte Schätzung von Aufwand und Zeit

- Erster Abstraktionsversuch: **Ideal Hours/Days**.
  - Geht davon aus, dass keine Störungen vorliegen und der Entwickler ununterbrochen an der Aufgabe arbeiten kann.
  - Ein Arbeitstag entspricht im Durchschnitt 6 Ideal Hours (bei 8 Std./Arbeitstag).
- Zweiter Abstraktionsversuch: **Story Points**
  - 1 – 1 – 2 – 3 – 5 – 8 – 13 – …
  - Geschätzt wird die Komplexität, nicht der zeitliche Aufwand.
  - Bezugsgrundlage ist die Erfahrung des Entwicklungsteams.
  - Komplexität und Story Points stehen im selben Verhältnis zueinander:
    Aufgabe mit 40 Story Points ist fünfmal so groß wie eine mit 8 Story Points.
- Dritter Abstraktionsversuch: **T-Shirt Größen**
  - XXS – XS – S – M – L – XL - XXL
  - Nur für Einsteiger in die Agilität.

Story Points folgen der Systematik der Fibonacci-Zahlen

- Die Fibonacci-Zahlen sind ganze Zahlen größer Null und beschreiben näherungsweise Wachstumsvorgänge in der Natur.
- Die nächste Zahl der Reihe entsteht aus der Summe der beiden vorherigen Zahlen.

1 1 2 3 5 8 13 21 34 55 89 144 233 377 …

- Als Formel: $f_n = f_{n-1} + f_{n-2}$ für $n \geq 3$ und $f_1 = f_2 = 1$
- Für das Schätzen in Storypoints wird häufig diese vereinfachte Folge verwendet.

1 **2** **3** **5** **8** **13** **20** **40** **100**

Geschwindigkeit

- Wird angegeben in Story Points pro Sprint.
- Mit der Kenntnis der Geschwindigkeit kann man abschätzen:
  - Wie viele Storys pro Sprint fertig gestellt werden können.
  - Wann ein Projekt abgeschlossen sein wird.
- Die Komplexität wird von Teams auf Basis ihrer Erfahrung geschätzt. Also unterschiedlich…
- Jedes Team hat seine individuelle Geschwindigkeit.
- Die Geschwindigkeit eines erfahrenen Teams bleibt nahezu konstant.

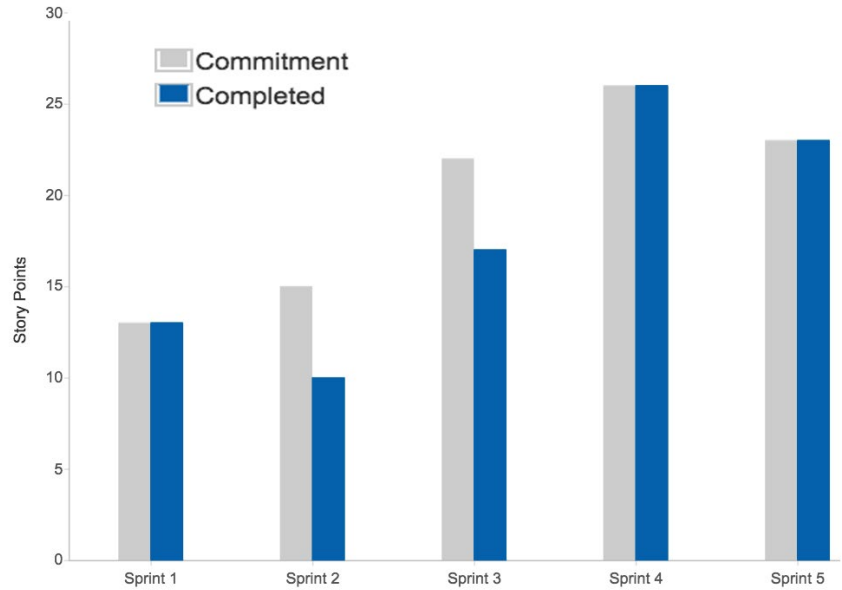
Beispiel

- Im letzten Projekt entwickelte ein Team 210 Story Points, verteilt über 10 Sprints.
- Die durchschnittliche Geschwindigkeit des Teams ist 210 / 10 = 21.
- Die Sprints dauerten jeweils 3 Wochen.
- Pro Sprintwoche wurden (durchschnittlich) 7 Story Points entwickelt.

Geschwindigkeit

Geschwindigkeit – Das „Velocity“ Diagramm

- „Velocity“ wird visualisiert
- Nach einigen Sprints werden Prognosen genauer und die Geschwindigkeit beständiger



Methoden, um Aufwand/Komplexität zu schätzen

- Planning Poker
- Triangulation
- Affinity Estimation / Estimation Game
- Magic Estimation

Planning Poker

- Jeder Entwickler erhält einen Kartensatz eines Planning Poker-Spiels.
- PO stellt den Entwicklern die zu schätzende Story vor.
- Der PO beantwortet Fragen der Entwickler zur Story.
- Jeder Entwickler wählt verdeckt eine Karte aus, die aus seiner Sicht zur Komplexität der Story passt.
- Alle Karten werden gleichzeitig aufgedeckt.
- Die Entwickler mit der niedrigsten und höchsten Schätzung begründen ihre Entscheidung.
- Der Vorgang wird wiederholt bis ein Konsens gefunden ist.

- Vor dem Schätzen einigt man sich, wie man vorgeht, falls nach mehreren Runden kein Konsens gefunden wurde.

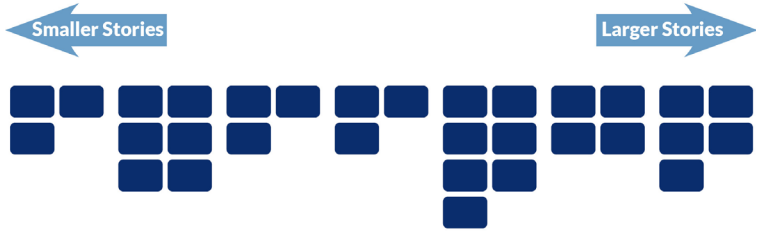
Triangulation

- Es werden zwei unterschiedlich große Stories als Referenz ausgewählt, bei denen sich das Team einig ist über die Größe. Z.B. eine 2 und eine 20.

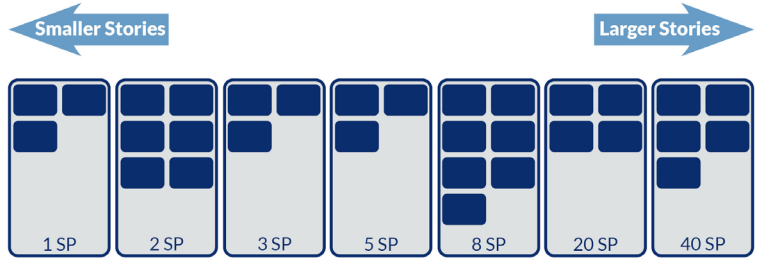
- Neue Stories werden gemeinsam besprochen und in Relation zu den bekannten Stories geschätzt und festgelegt.

Affinity Estimation / Estimation Game

- Basiert auf der Triangulation
- Alle Story Cards werden auf einem Stapel verdeckt abgelegt.
- Jeder Entwickler nimmt der Reihe nach eine Karte und hängt sie an ein Board. Links wenig komplex, rechts sehr komplex. Sobald die erste Karte hängt,
  muss die nächste Karte in Relation zu vorhandenen Karten gehängt werden.

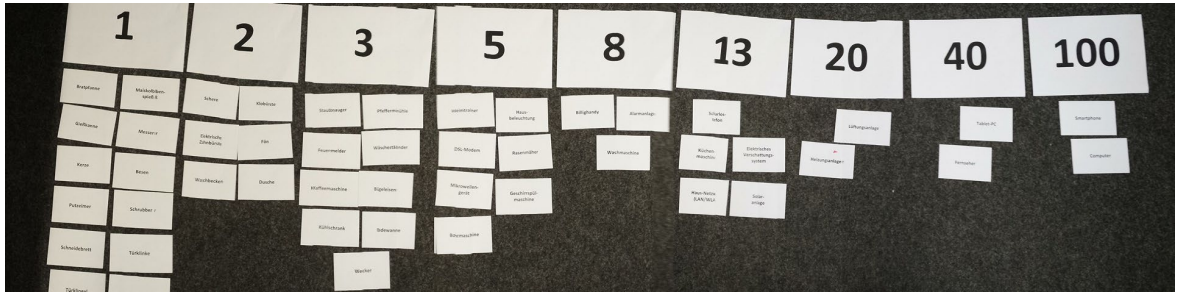


- Wenn alle Karten verteilt sind, werden Gruppen ähnlich komplexer Stories gebildet und einer Größe an Story Points zugeordnet.



Magic Estimation (nach Boris Gloger)

- Die Karten der Fibonacci-Zahlenfolge werden ausgelegt.
- Jeder Entwickler erhält einen Stapel mit User Stories.
- Die Entwickler ordnen stillschweigend ihre Stories den Fibonacci-Karten zu.
- Jeder Entwickler kann die Zuordnung von Karten ändern.
- Karten, deren Zuordnung umstritten ist, legt der PO zur Seite. Sie werden anschließend gemeinsam geklärt.
- Diese erste Schätzung sollte später noch einmal genauer durchgeführt werden. Vor allem bei Stories der Größen 1 bis 5.
  Größere werden wahrscheinlich ohnehin weiter aufgeteilt und neu geschätzt.

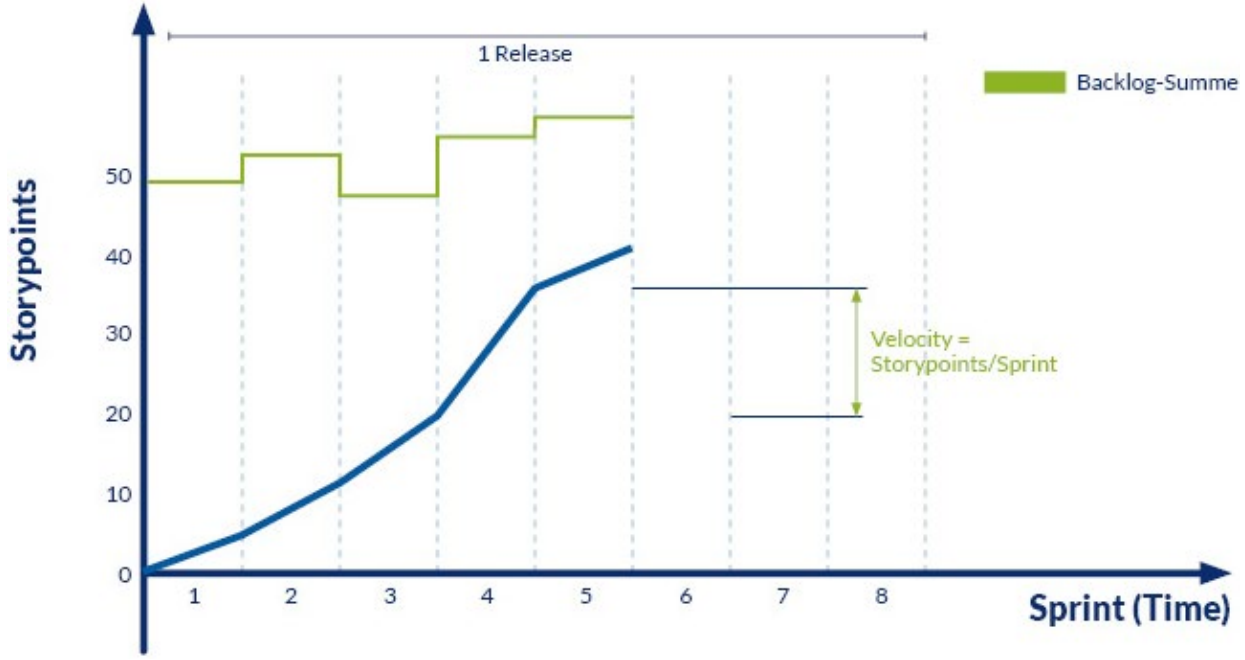


Neu schätzen – eine Story ist anders, als geschätzt…

- Das ist kein Problem, sondern eine Lernerfahrung.
- Darum wird eine begonnene Story nicht neu geschätzt.
  Weder während der Arbeit noch nach Abschluss.
- Um den Fehler nicht zu wiederholen, können unerledigte Stories
  neu geschätzt werden, die auf der falschen Einschätzung beruhen.
- So wenig Stories neu schätzen wie nötig,
  um das **Gleichgewicht** wieder herzustellen.

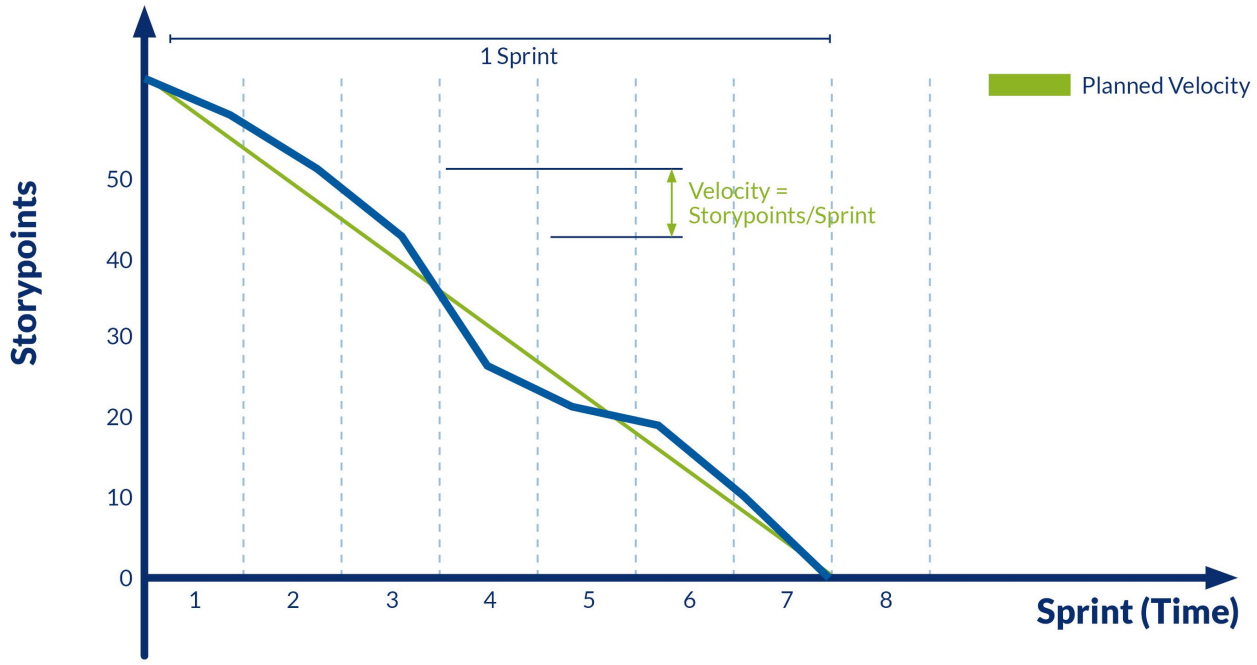
Burn-Up Chart

- Zeigt den Fortschritt eines Projekts.
- Verantwortung: Sprint – Entwickler, Projekt – Product Owner.
- Zeigt die Menge an **erledigter Arbeit**.
- Möglich: Auch Ziellinie anzeigen (SP gesamt im Backlog).



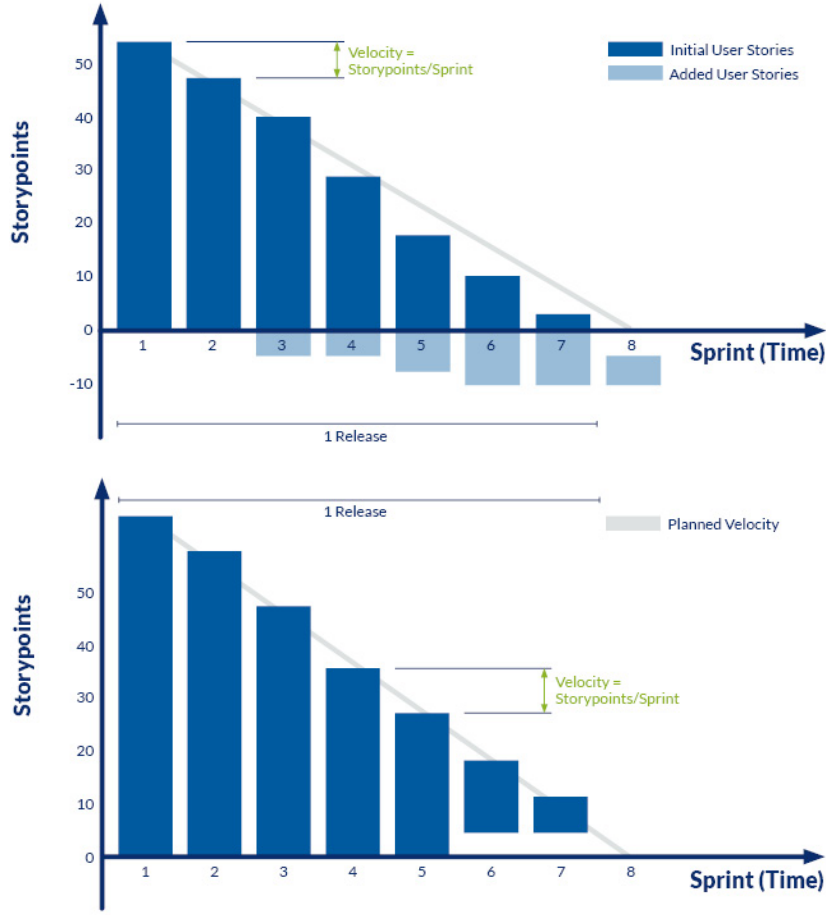
Burn-Down Chart

- Zeigt den Fortschritt eines Projekts.
- Verantwortung: Sprint – Entwickler, Projekt – Product Owner.
- Zeigt die Menge an noch **ausstehender Arbeit**.



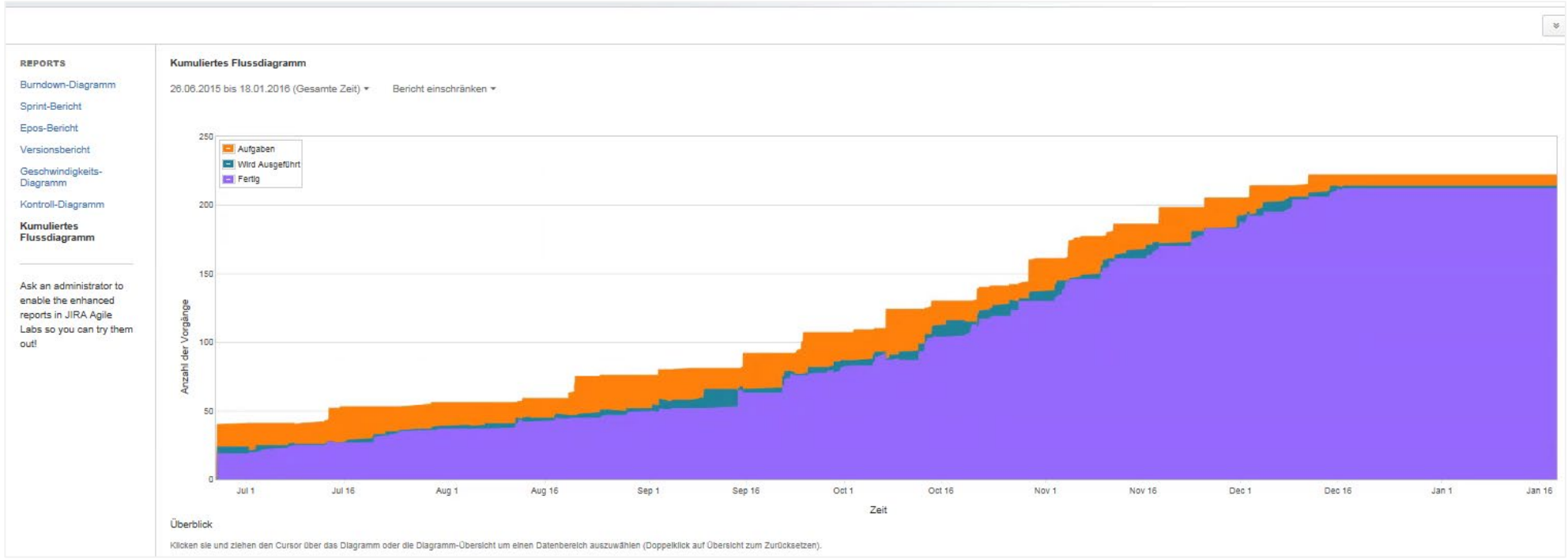
Burn-Down Bar-Chart (Balkendiagramm)

- Ein Balken zeigt in gesamter Länge die noch zu erledigende Arbeit.
- Wenn Arbeit erledigt ist, sinkt die obere Linie.
- Wenn Arbeit neu geschätzt wird, bewegt sich die obere Linie (nach oben oder unten).
- Wenn neue Arbeit dazu kommt, senkt sich die untere Linie des Balkens.
- Wenn Arbeit entfernt wird, geht die untere Linie des Balkens nach oben.



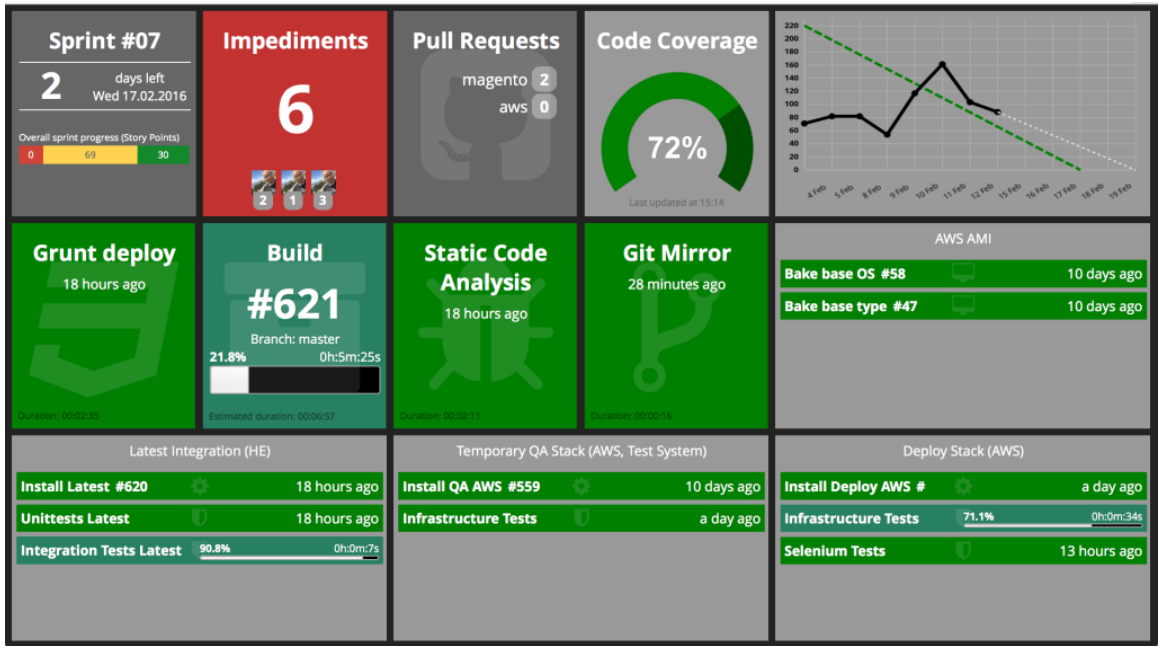
Kumulatives Flussdiagramm

- Beispiel aus Jira.
- Flussdiagramm eher Kanban als Scrum, aber anwendbar.



Information Radiator

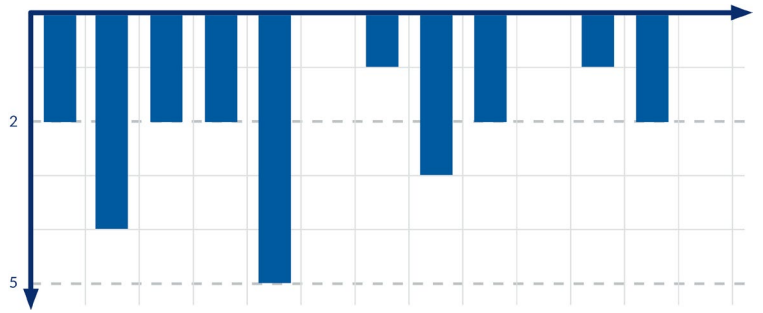
- Großes öffentlich sichtbares Board oder Screen, z.B. im Entwicklerraum (auch „Big Visible Chart - BVC“)
- Enthält relevante, aktuelle Informationen zum Projekt, z.B. Burndown-Chart, Sprint Backlog, Task-Board, Testergebnisse, Code-Statistiken, Defekte, Blocker.



Escaped Defects I Niko-Niko Kalender

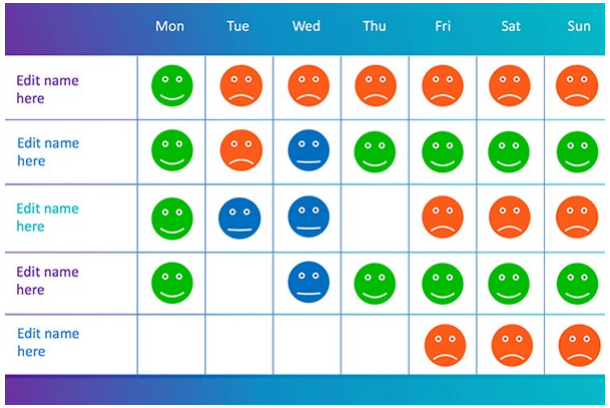
Escaped Defects

- Fehler, die im Testprozess unbemerkt blieben und von Kunden oder Endanwendern entdeckt wurden.
- Visualisierung durch Balkendiagramme.
- Beseitigte „Escaped Defects“ sind kein schlechtes Zeichen, sondern Ausdruck eines guten Qualitätsmanagements.



Niko-Niko Kalender

- Visualisierung der Teammoral durch Smilies (ggf. anonym).
- Erkennen von Korrelationen mit bestimmten Ereignissen im Sprint.
- Wichtiges Instrument für den Scrum Master, um dem Team zu helfen.
- Ein motiviertes Team ist ein erfolgreiches Team.



Feste Einheit I Festpreis

Zeit & Material / feste Einheit (Time & Means / Fixed Unit)

- Projektleistung wird berechnet nach
  - tatsächlicher Anzahl von Personentagen/-stunden oder
  - tatsächlicher Anzahl von Sprints.
    Teamgröße und Sprintdauer werden ebenfalls festgelegt.
- Häufig als Dienstvertrag bezeichnet.
- Mit agilem Ansatz vereinbar, weil erbrachte Zeit und Aufwand berechnet werden.

Werkvertrag / Festpreis (Fixed Price)

- Voraussetzung: bekannter und vereinbarter Projektumfang.
- Festpreise können von bestimmten Organisationen vorgeschrieben sein.
- Mit agilem Ansatz nicht vereinbar.

Fachbegriffe

- Return on Investment (ROI)
- Net Present Value (NPV) / Kapitalwert
  - Gesamtbetrag der Investitionen abzüglich der Erträge. Für die Anpassung an die Geldentwertung erfolgt ein „Abzinsung“.
- Payback Period / Amortisationszeit
  - Zeitraum, der benötigt wird, um mit dem Produkt so viel Geld zu verdienen, wie man dafür investiert hat.
- Internal Rate of Return (IRR) / Zinsfuß-Methode
  - Diskontsatz, bei dem der NPV oder Kapitalwert Null wird.
    Invest = Ertrag inkl. Abzinsung.
- Total Cost of Ownership (TCO)
  - Kombination aus den Bereitstellungs- und Betriebskosten des Produkts.

9. Extreme Programming (XP)

Extreme Programming (XP)

- Methodik für die Software-Entwicklung, die den Fokus auf die Kundenzufriedenheit legt
  - Reduzierung der Kosten
  - Mehrere kurze Entwicklungszyklen
  - Änderungen sind: natürlich, unausweichlich, wünschenswert
- Grundsätze
  - Einfachheit voraussetzen
  - Den Wandel annehmen
- Praktiken
  - Paarweise Programmierung
  - Kontinuierliche Integration
- Herzstück ist das "Konzept der Regeln" mit den Schwerpunkten
  - Planung, Management, Design, Codeschreiben und Testen

Praktiken

- **Paarprogrammierung (Pairing / Pair Programming)**
- **Kollektives Eigentum (Code Sharing)**
- Einfaches Design (Simple Design)
- **Testgetriebene Entwicklung (Test Driven Development - TDD)**
- Standardisierter Code (Code Standards)
- **Refactoring**
- **Laufende Integration (Continuous Integration)**
- Go Home! (Constant Pace)
- Daily Standup
- Tracking
- Risikomanagement (Risk Management)
- **Spiking**

Paarprogrammierung

- Pair Programming ist Teil der agilen Softwareentwicklung und findet insbesondere im Extreme Programming Anwendung. Dabei sitzen zwei Programmierer zusammen an einem Arbeitsplatz und entwickeln gemeinsam Software.
- Alle Teammitglieder verstehen und bearbeiten den Code → kollektives Eigentum
- Zwei Teammitglieder, eine Aufgabe: "Fahrer" und "Navigator" wechseln sich ab.
- Kontinuierliche Integration ermöglicht permanente Reflexion ("Echtzeit-Reviewing")
- Vorteile:
  - weniger Fehler
  - kleinere Programme
  - höhere Disziplin
  - bessere Qualität des Code
  - belastbarerer Flow
  - mehr Freude an der Arbeit
  - geringeres Risiko im Projekt (Bus Factor)
  - synergetischer Wissenszuwachs
  - Teambildung
  - Paare werden seltener unterbrochen als jemand, der allein arbeitet

Kollektives Eigentum

- Entwickler / Entwicklerpaar wählt selbständig seine Aufgaben („pull“ statt „push“).
- Verantwortung für Aufgaben liegt im gesamten Team und somit bei jedem einzelnen Entwickler (Collective Code Ownership).
- Es gibt kein Monopol von Einzelwissen.
- Wissensaustausch wird durch regelmäßige, geeignete Maßnahmen, z. B. Pair Programming erreicht.

Test-Driven Development (TDD)

- Der Programmierer erstellt Softwaretests konsequent vor den zu testenden Komponenten.
- Vorteil: es wird gerade so viel Code geschrieben, bis der Test erfolgreich ist.
- Getesteter neuer Code wird laufend zum bisherigen Code hinzugefügt und als Ganzes erneut getestet (Continuous Integration).
- Ergebnis: Kontinuierliche Verbesserung.
- Testtypen:
  - Modultests
  - Integrationstests
  - Performancetests
  - Akzeptanztests
  - Regressionstests

Refactoring

- Code-Refactoring ist der Prozess der Umstrukturierung von bestehendem Computercode ohne Änderung seines äußeren Verhaltens.
- Vorteile:
  - Systematische Verbesserung
  - Verbesserung der Lesbarkeit
  - Reduktion der Komplexität
  - Verbesserung der Wartungsfreundlichkeit
  - Verbesserung der Erweiterbarkeit

Laufende Integration

- Einzelne Komponenten werden regelmäßig und in kurzen Zeitabständen in ein lauffähiges Gesamtsystem (Produktinkrement) eingebunden (Continuous Integration).
- Hohe Integrationsroutine bis hin zur Automatisierung.
- Integrationskosten werden reduziert.
- Frühzeitige Fehlererkennung und -behebung verringert Kosten.
- Auch „Permanent Integration“ genannt.

- In Abgrenzung dazu:
  - Continuous Delivery
  - Continuous Deployment

Spiking

- Bezeichnet das schnelle Ausprobieren (quick & dirty) eines Stücks Entwicklungscode oder eines kleinen Prototyps.
- In XP übliche Techniken / Praktiken (einfaches Design, Paarprogrammierung, testgetriebene Entwicklung etc.) spielen keine Rolle.
- Der verwendete Code wird nicht in die Produktentwicklung übernommen.
- Erkenntnisse des Spikings fließen in die Produktentwicklung ein.

10. Weitere agile Methoden

Definitionen

- Agile Werte bilden das Fundament.
- Agile Prinzipien basieren auf den agilen Werten und bilden Handlungsgrundsätze.
- Agile Techniken sind konkrete Verfahren zur Umsetzung der agilen Prinzipien.
- Agile Methoden geben den agilen Techniken eine Gesamtstruktur hin zum Projektmanagement.

- Agile Methoden sind Vorstrukturierungen auf der Ebene von Prozessmodellen. Prinzipien und Techniken werden zu einem schlüssigen Prozess kombiniert.
- Agile Methoden müssen im Allgemeinen für jedes Projekt und Projektumfeld adäquat angepasst werden.

Quelle: Haufe.de

Crystal-Methodiken (1/2)

- Gruppe leichtgewichtiger Agiler Methoden
- Entwickelt in den 1990er Jahren von Alistair Cockburn
- Hauptaugenmerk liegt nicht auf Prozessen, sondern auf:
  - Menschen / Interaktionen / Gemeinschaft / Fertigkeiten / Talenten / Kommunikation

- Wesentliche Aspekte:
  - **Methodik**: eine Gruppe von Elementen (z. B. Praktiken, Werkzeuge)
  - **Techniken**: die Kompetenzbereiche (z. B. Entwicklung von Anwendungsfällen)
  - **Leitlinien**: Definitionen, wie sich die Organisation verhalten muss

Crystal-Methodiken (2/2)

- Crystal-Methodenfamilie: acht verschiedene Farben von Crystal Clear bis Saphire

- Gemeinsame Eigenschaften aller Varianten:
  - Häufige Releases (Versionen)
  - Reflektive Verbesserung
  - Enge oder osmotische Kommunikation
  - Persönliche Sicherheit
  - Fokussiertes Arbeiten
  - Leichter Zugang zu kundigen Anwendern/Anwenderinnen

Cockburn Skala

Kritikalität

	Clear	Yellow	Orange	Red	Maroon
Life (L)	**L6**	**L20**	**L40**	**L80**	**L200**
Essential Money (E)	**E6**	**E20**	**E40**	**E80**	**E200**
Discretionary Money (D)	**D6**	**D20**	**D40**	**D80**	**D200**
Comfort (C)	**C6**	**C20**	**C40**	**C80**	**C200**
	1 – 6 Members	7 – 20 Members	21 – 40 Members	41 – 80 Members	81 – 200 Members

Teamgröße

Weitere agile Methoden

Häufige Releases

- Reales Feedback kommt vor allem von Endbenutzern (weniger von Kunden).
- Releases sind notwendig, um Feedback über Nutzerbedürfnisse und Nutzerverhalten zu erfahren.
- Jedes Release liefert Wert.

Osmotische Kommunikation

- Ermöglichung durch räumlich kollaborierende Teams
- Virtuelle Zusammenarbeit erfordert besondere Instrumente, um osmotische Kommunikation zu ermöglichen.

Walking Skeleton

- Die einfachste Umsetzung eines kompletten Funktionsbereiches einer Software, auch vertikaler Durchstich genannt.

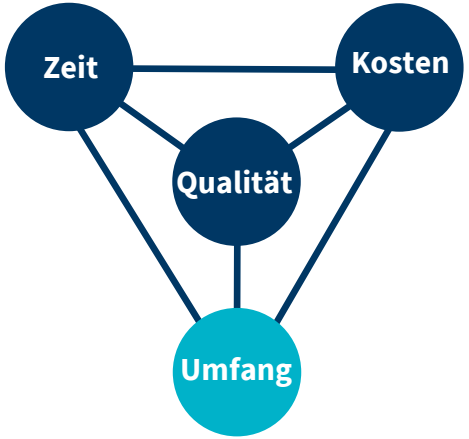
Dynamic Systems Development Method (DSDM) (1/2)

- Neuer Name ist **ABC** nach der zugehörigen Organisation Agile Business Consortium.
- Prinzipien:
  1. Aktive Beteiligung der Anwender
  2. DSDM-Teams müssen Entscheidungsbefugnisse erhalten
  3. Fokus liegt auf der häufigen Lieferung von Produkten
  4. Maßgebliches Kriterium für die Ergebnisse: Eignung für den Geschäftszweck
  5. Iterative und inkrementelle Entwicklung
  6. Während der Entwicklung sind alle Veränderungen reversibel
  7. Anforderungen beruhen auf High-Level-Basisszenario
  8. Tests werden über den Lebenszyklus hinweg integriert
  9. Essenziell: Kooperativer Ansatz, Zusammenarbeit aller Stakeholder

Dynamic Systems Development Method (DSDM) (2/2)

- DSDM-Framework umfasst 6 Phasen:
  - Phase 1 – Pre-Project Phase (Projektvorbereitung)
  - Phase 2 – Feasibility (Machbarkeit)
  - Phase 3 – Foundations (Grundlagen)
  - Phase 4 – Evolutionary Development (Evolutionäre Entwicklung)
  - Phase 5 – Deployment (Bereitstellung)
  - Phase 6 – Post-Project Phase (Projektnachbereitung)

Hohe Bedeutung: **MoSCoW**

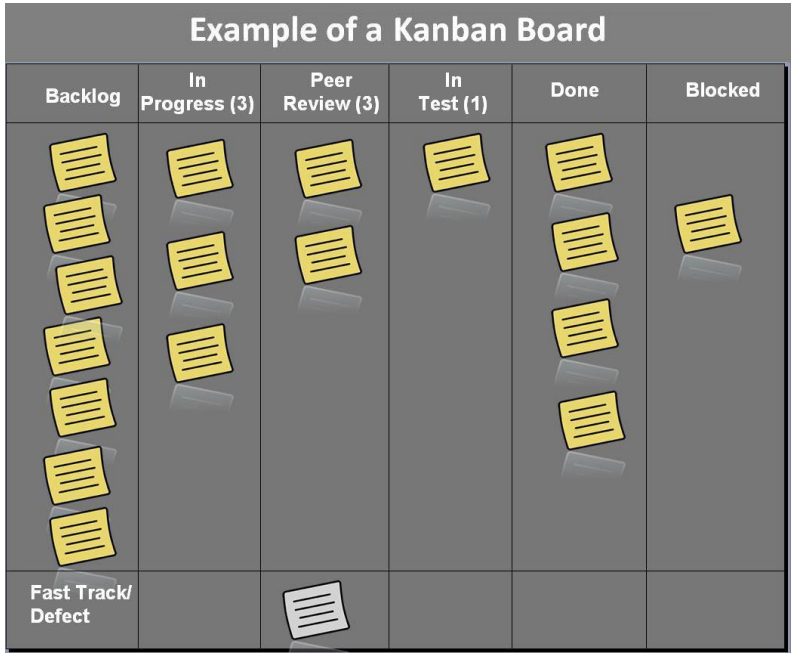


DSDM-Vorgabe: Nur das Minimum an Arbeit leisten, um zum nächsten Eintrag (Item) zu gelangen!

Kanban

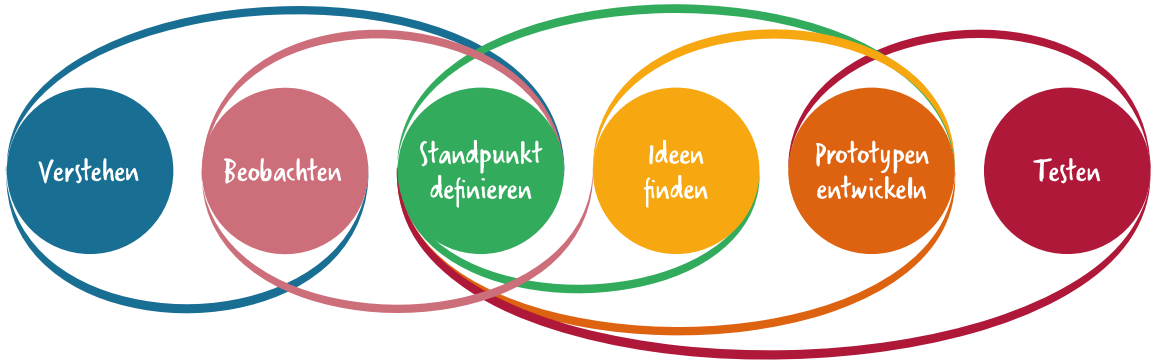
- Kanban, japanisch für „Signalkarte“, ist ein agiles Framework, das die Optimierung des Wertstroms in den Mittelpunkt rückt.
- Visualisierung von Tätigkeiten führt zu mehr Transparenz, besserer Zusammenarbeit und besserem Feedback
- Verwendung von “Work-In-Progress-Limits” (WIP)
  (Ist ein WIP-Limit erreicht, unterstützen Entwickler in dieser Spalte, um drohende Blockaden zu vermeiden)
- Nutzung von “Blocker-Tickets”
- Konsequentes Pull-Prinzip:
  Entwickler ziehen ihre Aufgaben selbstständig
- Ziel: Verfolgung eines optimalen Arbeitsflusses

Kanban-Techniken finden in vielen agilen Methoden Anwendung.



Design Thinking

- Annahme: Probleme können besser gelöst werden, wenn Menschen unterschiedlicher Disziplinen in einem die Kreativität fördernden Umfeld zusammenarbeiten.
- Spezielle Herangehensweise zur Bearbeitung komplexer Problemstellungen und zur Entwicklung neuer Ideen, die aus Anwendersicht überzeugend sind.
- Sechs Schritte (Hasso-Plattner-Institut, Potsdam)
  - Verstehen
  - Beobachten
  - Standpunkt definieren
  - Ideen finden
  - Prototyp
  - Testen

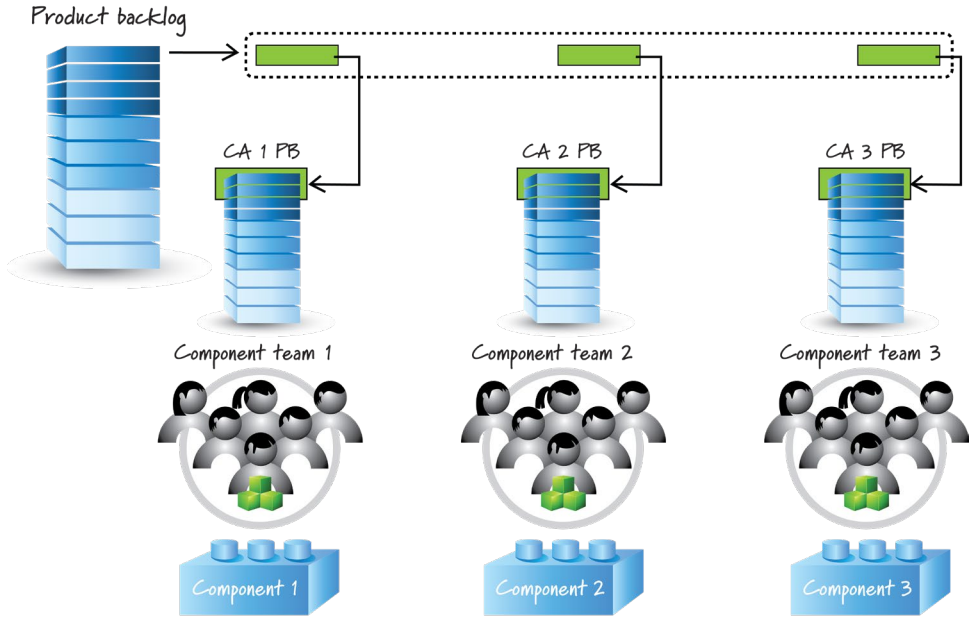


Quelle: hpi-academy.de

11. Skalierbare agile Methoden

Skaliertes Scrum

- In komplexen Projekten mit vielen Beteiligten können **multiple Teams** an einem Produkt arbeiten.
- Scrum bevorzugt beständige Teams! In einer skalierten Umgebung können Entwickler mit speziellen Fähigkeiten zwischen mehreren Teams wechseln.
- Aufstellung interdisziplinärer Teams, die ohne technische Abhängigkeiten zu anderen Teams Product Backlog Items (PBIs) fertigstellen können.
- Typen von Teams:
  - Komponenten-Teams
    (arbeiten an einzelnen Komponenten)
  - Feature-Teams
    (arbeiten an vollständigen Features)



Skaliertes Scrum | Skalierung von Teams

- Bei Unerfahrenheit oder fehlender Methode Start mit einem oder wenigen Teams

Drei unterschiedliche Vorgehensweisen bei der Skalierung:

- **Split-and-Seed:**
  - Ein Team eines Pilotprojekts wird verteilt. Die erfahrenen Mitarbeiter bilden die Basis für mehrere (bis zu 4) neue Teams.
- **Grow-and-Split:**
  - Einem Pilot-Team werden zusätzliche Mitglieder hinzugefügt, die zwei bis drei Sprints zusammenarbeiten. Anschl. Aufteilung in zwei neue Teams mit erfahrenen und weniger erfahrenen Mitgliedern.
- **Internes Coaching:**
  - Erfahrene Mitglieder erster Teams coachen zeitweise neue Teams

Skaliertes Scrum | Scrum Master und Product Owner

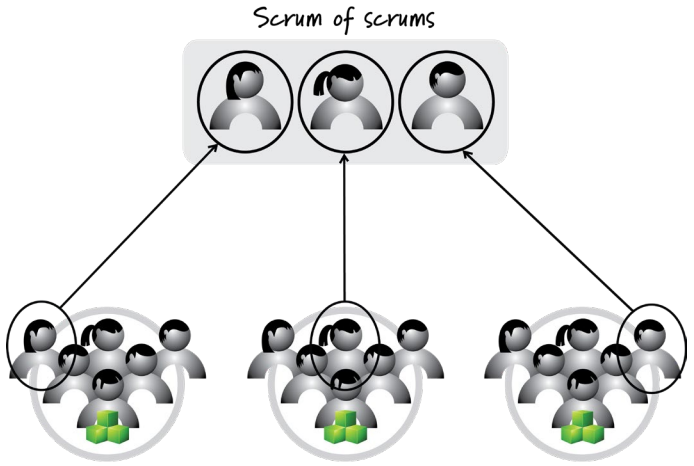
- Jedes Team benötigt einen **Scrum Master**.
  (In der Praxis kommt es vor, dass ein Scrum Master mehreren Teams dient.)

- Wie viele **Product Owner** gibt es?
  - Variante 1: „The One and Only“ – es gibt nur einen Product Owner
    (üblich in LeSS und Nexus)
  - Variante 2: Ein Product Owner für jedes Team.
    Ein **Chief Product Owner** für Alle.

- ***Ein Produkt – ein Product Backlog!***
  - Filterung eines komplexen Backlogs mit großer Anzahl von Einträgen:
    Anforderungen werden in **Epics oder Themen** zusammengefasst.

Skaliertes Scrum | Scrum of Scrums

- Event, um die Arbeit mehrerer Scrum Teams zu synchronisieren
- Jedes Scrum Team wird von einem Mitglied repräsentiert (i.d.R. Entwickler).
- Die rotierende Teilnahme unterschiedlicher Mitglieder aus einem Team ist empfehlenswert.
- Kann ähnlich wie ein Daily Scrum täglich stattfinden
- Andere Formate (Häufigkeit, Timebox, Mitglieder) sind möglich
- Übliche Fragen:
  - Was hat mein Team seit dem letzten Scrum of Scrums getan?
  - Was wird mein Team bis zum nächsten Scrum of Scrums voraussichtlich erledigen?
  - Was behindert mein Team bei seiner Arbeit?
  - Beeinflussen oder behindern Tätigkeiten meines Teams ein anderes Team bei seiner Arbeit?



Nexus

- Framework für die nachhaltige Ausführung einer skalierten Produktentwicklung
- Scrum ist ein grundlegender Baustein
- Nexus ändert Scrum nicht, sondern ergänzt es
- Ein **Nexus Sprint** ist eine Entwicklungseinheit, die aus vielen Iterationen besteht, die in Form eines „integrierten Inkrements“ Wert für Kunden liefert
- Nexus-Ansatz konzentriert sich immer nur auf jeweils einen Sprint oder eine Timebox → De Facto Abkehr vom Release-Ansatz
- Grund: Eine Release-Planung beruht zu Beginn eines Projekts auf unvollständigem Wissen (Wasserfallplanung ist nahezu unmöglich)
- Nexus konzentriert sich immer auf eine einzige Iteration

Large Scale Scrum (LeSS)

- Kann angewendet werden, wenn mehrere Teams gemeinsam an der Entwicklung eines Produkts oder Services arbeiten.
- Eignet sich für folgende Fälle:
  - Für eine spezifische Entwicklung sind mehrere Teams erforderlich.
  - Mehrere Teams arbeiten zusammen an einem gemeinsamen Ziel.
  - Mehrere Teams arbeiten gleichzeitig an einem Produkt oder Service.
- Erfüllt ein Projekt diese Anforderungen nicht, ist es besser, die Standardvariante von Scrum zu verwenden.

Die **10 Prinzipien** von LeSS:

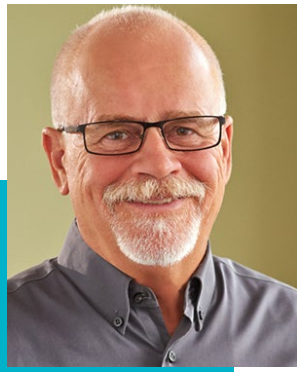
Figure 38 Concepts used in LeSS (Picture created by EXIN based on https://less.works/img/principles/principles.pdf)

Scaled Agile Framework® (SAFe®)

- Frei verfügbarer Wissensfundus (Body of Knowledge) für die Skalierung von Agile über ein Team hinaus
- Unternehmensweite Entwicklungsmethodik verbindet Prinzipien aus Lean und Agile
- Konzeptionen in den Bereichen Organisation und Arbeitsfluss unterstützen Organisationen bei der Skalierung ihrer Lean- und Agile-Praktiken
  (Eines von neun Prinzipien: Laufende Arbeit (WIP) visualisieren und limitieren)

- Vier Konfigurationen:
  - Essential SAFe
  - Portfolio SAFe
  - Large Solution SAFe
  - Full SAFe

Last but not least…



Every business is a software business now.
Agility isn't an option, or a thing just for teams,
it is a business imperative.
But we struggle building big systems …

Dean Leffingwell, Creator of SAFe

It is not enough that management commit themselves to quality and productivity, they must know what it is they must do.
Such a responsibility cannot be delegated.

W. Edwards Deming

Gemeinsam weiterbilden

Vielen Dank für die Aufmerksamkeit.

Quellenangabe

- Alle Bilder ohne Quellenangabe in der Präsentation stammen aus der Microsoft Onlinebilder Bibliothek: "Dieses Foto" von Unbekannter Autor ist lizenziert gemäß CC BY-NC-ND
- Weitere Grafiken: Copyright © 2012, Kenneth S. Rubin and Innolution, LLC. All Rights Reserved.
- Johann Botha | Das EXIN-Handbuch für Scrum Master und Product Owner
- Ken Schwaber & Jeff Sutherland | The Scrum Guide 2020
- Nader K. Rad | Agile Scrum Handbook | Van Haren Publishing (3nd edition, 2021)