



Agiles Projektmanagement

Agile Scrum Foundation

Version 2.1.1 – Deutsch

© Bernhard Schütz – info@edubs.info



Inhalt

Nr.	Kapitel	Seite
1.	Klassische und agile Methoden im Projektmanagement	3
2.	Das Agile Manifest	8
3.	Scrum	12
4.	Rollen	24
5.	Sprint	38
6.	Meetings & Artefakte	46
7.	Planen	84
8.	Schätzen	92
9.	Monitoring	105
10.	Rahmenbedingungen	111
11.	Weitere agile Methoden und Verfahren	127
12.	eXtreme Programming (XP)	130
13.	Dynamic Systems Development Method (DSDM)	145
14.	Kanban, ScrumBan und ScrumBut	151
15.	Abschluss	155
16.	Anhang – Musterprüfung	159

Taskboard / Scrum Board

- ▶ Mit einem Taskboard können die Inhalte eines Sprint Backlogs, also die für den Sprint ausgewählten Einträge des Product Backlogs, dargestellt werden. Dazu gehören:
 - Die zu bearbeitenden Aufgaben
 - Der Status der zu bearbeitenden Aufgaben

Definition of „Done“

1. _____
2. _____

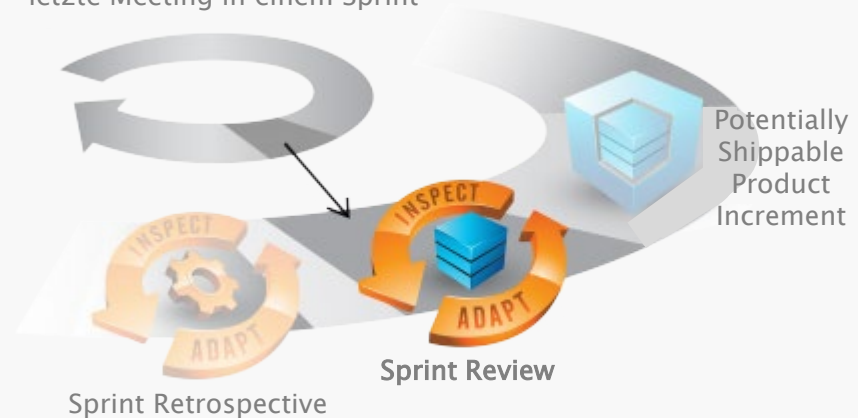
User Story	Zu tun (To Do)	In Arbeit (Work in Progress / WiP)	Erledigt (Done)
<u>User Story 1</u> Als < Rolle > möchte ich < Funktion >, um < Nutzen >.	<u>User Story 1</u> Aufgabe D <u>User Story 1</u> Aufgabe C	<u>User Story 1</u> Aufgabe A	<u>User Story 1</u> Aufgabe B
<u>User Story 2</u> Als < Rolle > möchte ich < Funktion >, um < Nutzen >.	<u>User Story 2</u> Aufgabe E <u>User Story 2</u> Aufgabe D	<u>User Story 2</u> Aufgabe C	<u>User Story 2</u> Aufgabe B <u>User Story 2</u> Aufgabe A
<u>User Story 3</u> Als < Rolle > möchte ich < Funktion >, um < Nutzen >.		<u>User Story 3</u> Aufgabe D <u>User Story 3</u> Aufgabe C	<u>User Story 3</u> Aufgabe A <u>User Story 3</u> Aufgabe B

Sprint Review (1)

- ▶ Meeting am Ende eines Sprints
Dauer: 1 Stunde pro Sprintwoche
Max. 4 Stunden bei einem vierwöchigen / einmonatigen Sprint.
- ▶ Beteiligte sind das Scrum Team sowie die Stakeholder
(auf Einladung des Product Owners).
Verantwortlich: Scrum Master
- ▶ Vorstellung des fertigen (!)
Produktinkrements durch das
Entwicklungsteam anhand der
Akzeptanzkriterien (Abnahme-
kriterien).
- ▶ Ein Produktinkrement, das zu
99 % fertig ist, wird nicht
abgenommen.

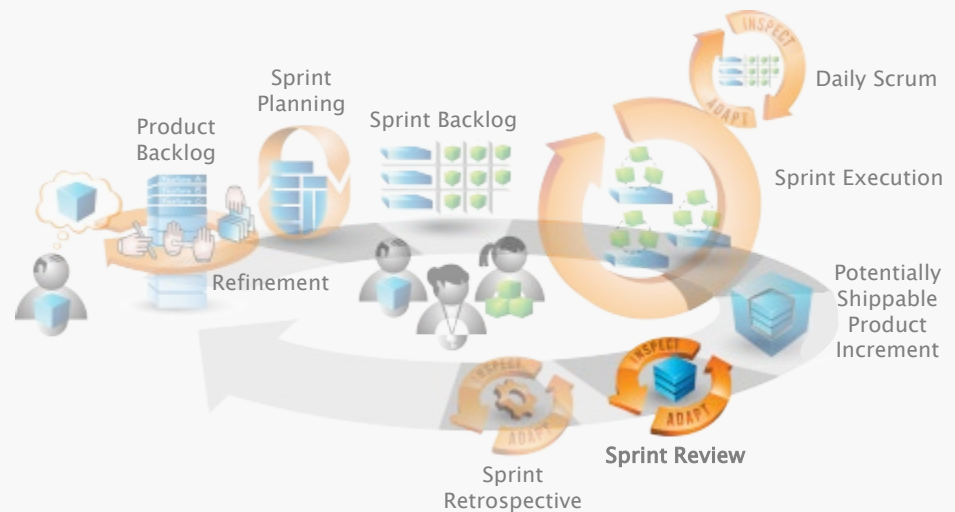


Das Sprint Review ist das vor-
letzte Meeting in einem Sprint



Sprint Review (2)

- ▶ Vorstellung, welche Sprint Backlog Items gem. der Definition of „Done“ nicht fertig geworden sind, warum nicht und was mit diesen weiter geschehen soll.
- ▶ Das Product Backlog wird vom Product Owner vorgestellt und ggf. für die kommenden Sprints angepasst.
- ▶ Klärung der weiteren Vorgehensweise in den kommenden Sprints (Sprint Planning).



Produktinkrement (1)

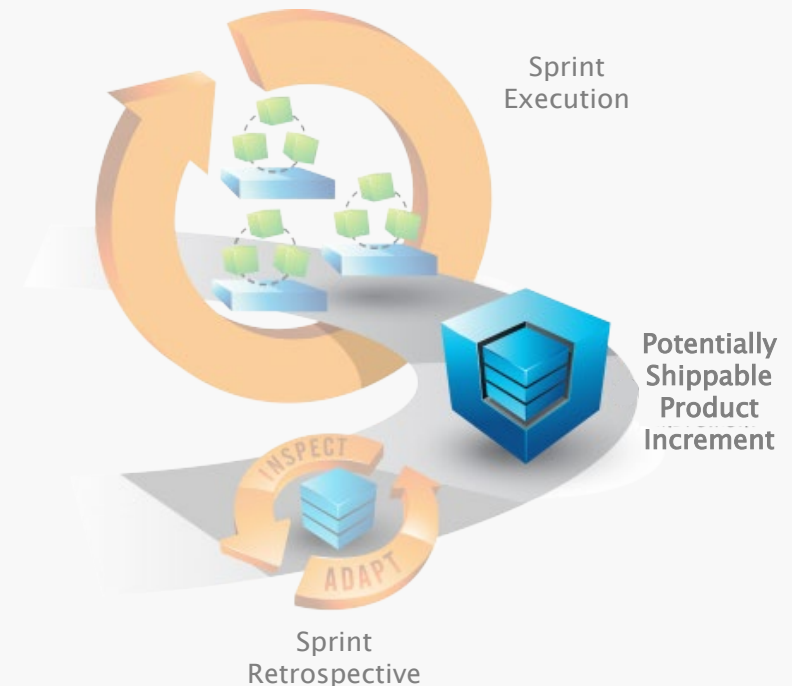
- ▶ Das Produktinkrement ist das Ergebnis aller im Sprint fertig gestellter Einträge des Product Backlog sowie aller Inkremente aus früheren Sprints des Projekts.
- ▶ Jedes neue Inkrement baut auf den vorhergehenden Inkrementen auf und enthält somit alle bereits implementierten Funktionalitäten.
- ▶ Sofern erforderlich, müssen dazu während des laufenden Sprints zusätzlich zu den Funktionstests auch Integrationstests vorgenommen werden.



Produktinkrement (2)

- ▶ Jedes Inkrement ist am Ende des Sprints in einem potentiell einsetzbaren Zustand – es ist „Done“.
- ▶ Das Entwicklungsteam übergibt dem Product Owner somit ein einsatzfähiges Produkt.
- ▶ Dies ist unabhängig davon, ob der Product Owner es tatsächlich ausliefern oder der Kunde es operativ einsetzen wird.

Potentially Shippable Product Increment (PSPI)



Definition of „Done“ (1)

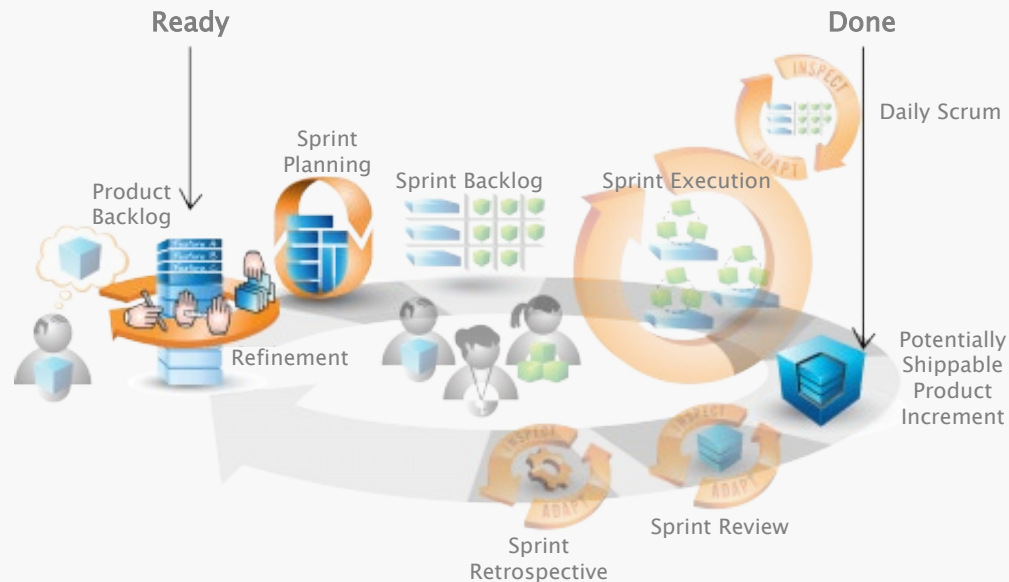
- ▶ Wenn ein Eintrag des Product Backlogs oder ein Produktinkrement „fertig“ entwickelt ist, muss es allen Beteiligten transparent und verständlich sein, was darunter zu verstehen ist.
Dafür ist die Definition of „Done“ zu formulieren.
- ▶ In der Definition of „Done“ sind einzuhaltende Kriterien, Richtlinien, Standards und Konventionen enthalten, z. B.
 - Qualitätsrichtlinien
 - Security Policies
 - Release Policies
- ▶ Die Definition of „Done“ wird fortlaufend weiterentwickelt, beginnend mit den wichtigsten einzuhaltenden Kriterien.

Definition of „Done“

1. -----
2. -----

Definition of „Done“ (2)

- ▶ Einzuhaltende Kriterien können sich sowohl auf funktionale als auch auf nicht funktionale Aspekte beziehen.
- ▶ Arbeiten mehrere Scrum Teams gemeinsam an einem Projekt gibt es eine von allen entwickelte Definition of „Done“.



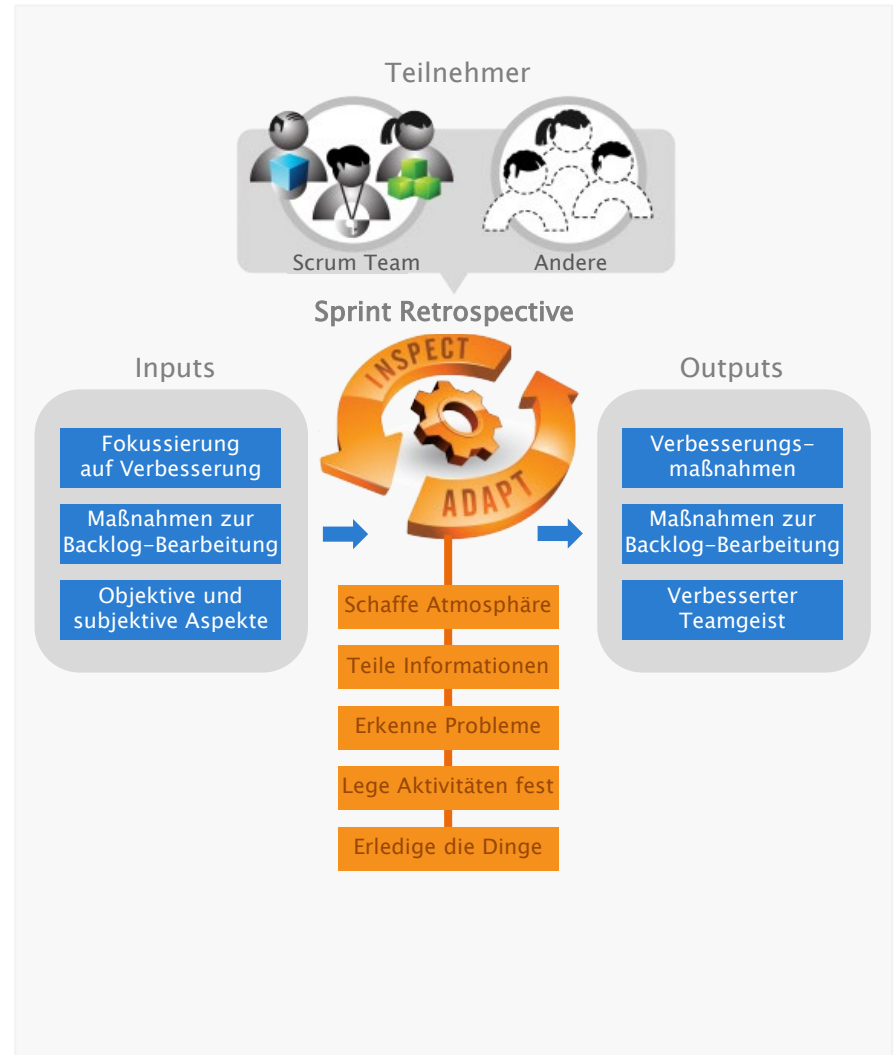
Sprint Retrospektive (1)

- ▶ Letztes Meeting eines Sprints.
Dauer: 45 Minuten pro Sprintwoche
Max. 3 Stunden bei einem vierwöchigen / einmonatigen Sprint.
- ▶ Beteiligte sind das Entwicklungsteam und der Scrum Master.
Das Team entscheidet, wer noch beteiligt wird
(in der Regel der Product Owner).
Verantwortlich: Scrum Master.
- ▶ Fokus der Sprint Retrospektive liegt auf der kontinuierlichen Verbesserung der verwendeten Methoden, Verfahren und Tools (inspect and adapt).



Sprint Retrospektive (2)

- ▶ Überprüfung der Arbeitsweise und Zusammenarbeit des Entwicklungsteams im letzten Sprint.
- ▶ Erstellung eines Verbesserungsplans für den nächsten Sprint bzgl.
 - Prozesse
 - Werkzeuge
 - Qualität
 - Kommunikation (innerhalb des Teams sowie nach außen, z. B. zu Stakeholdern)
 - Definition of „Done“





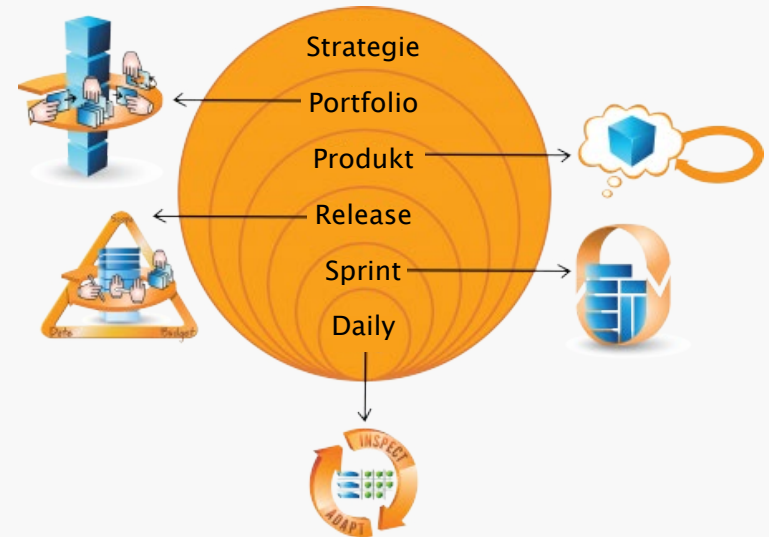
Agile Scrum Foundation

7. Planen



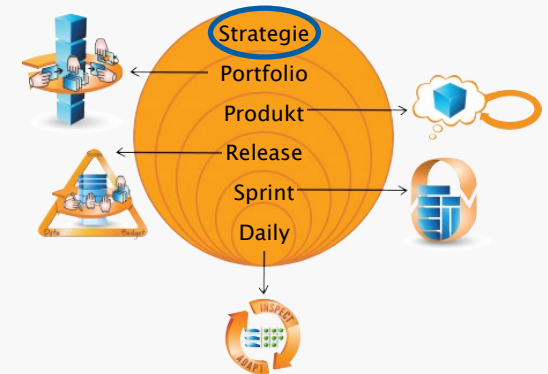
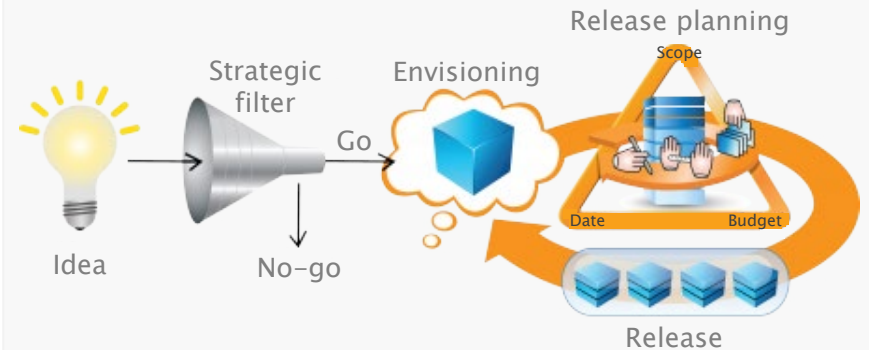
Planning Onion – Überblick

- ▶ Herunterbrechen der Planung von der Strategie bis zu täglichen Aktivitäten.
- ▶ Jede Schicht gibt den Rahmen für die jeweils darunter liegende Schicht vor bzgl. ...
 - Planungsziele
 - Zeitrahmen
 - Verantwortlichkeiten
 - Schnittstellen



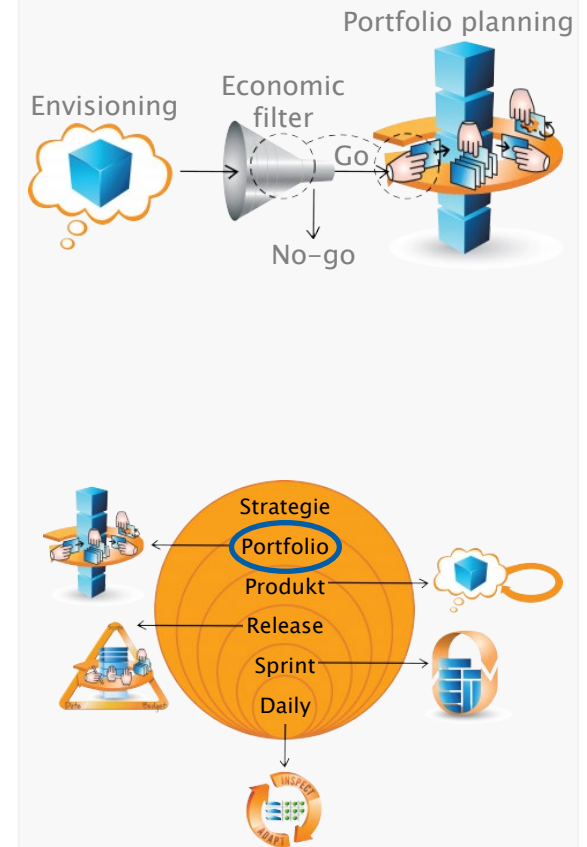
Planning Onion – Strategie

- ▶ Oberste Ebene für die Planung strategischer Geschäftsziele
- ▶ Üblicherweise 1-Jahresplan und 5-Jahresplan
- ▶ Verantwortlich:
C-Level Management
(CEO, CFO, CIO, ...)



Planning Onion – Portfolio

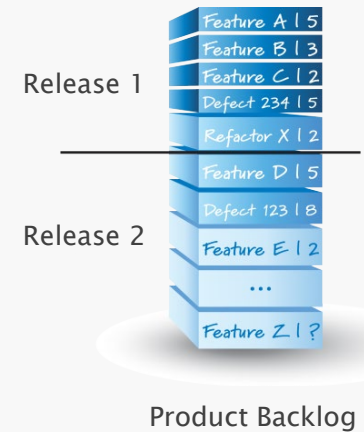
- ▶ Planungsebene für das gesamte Angebot aller Services, Produkte, Applikationen und Tools mitsamt ihren wechselseitigen Abhängigkeiten und Beeinflussungen.
- ▶ Portfolio-Entscheidungen unterstützen direkt die Erreichung strategischer Ziele.
- ▶ Langfristiger Planungszeitraum, jedoch kürzer als der strategische.
- ▶ Macht Vorgaben für die Road Maps der einzelnen Produktteams.
- ▶ Verantwortlich:
Oberes Management der Produkt- und Produktlinienentwicklung



7. Planen

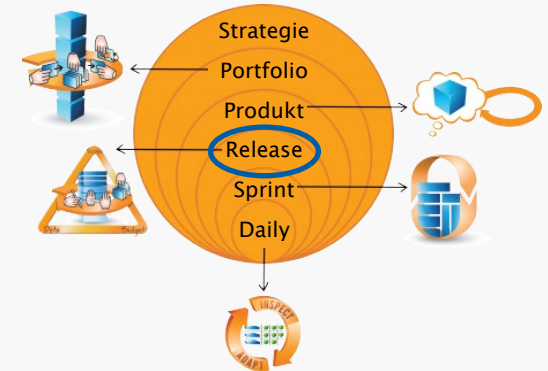
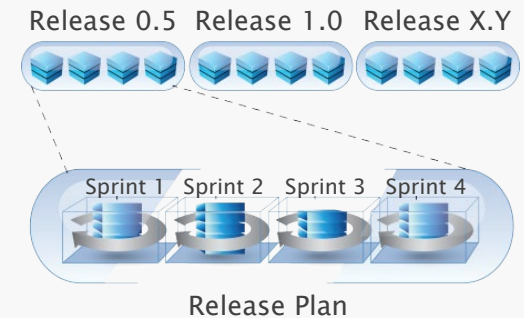
Planning Onion – Produkt

- ▶ Planungsebene für Produktsuiten und einzelne Produkte
- ▶ Berücksichtigt Produktvisionen und Road Maps
- ▶ Unterstützt die Entwicklung von Portfolio und Strategie
- ▶ Verantwortlich:
Product Owner – Produkt- /Applikationsmanager



Planning Onion – Release

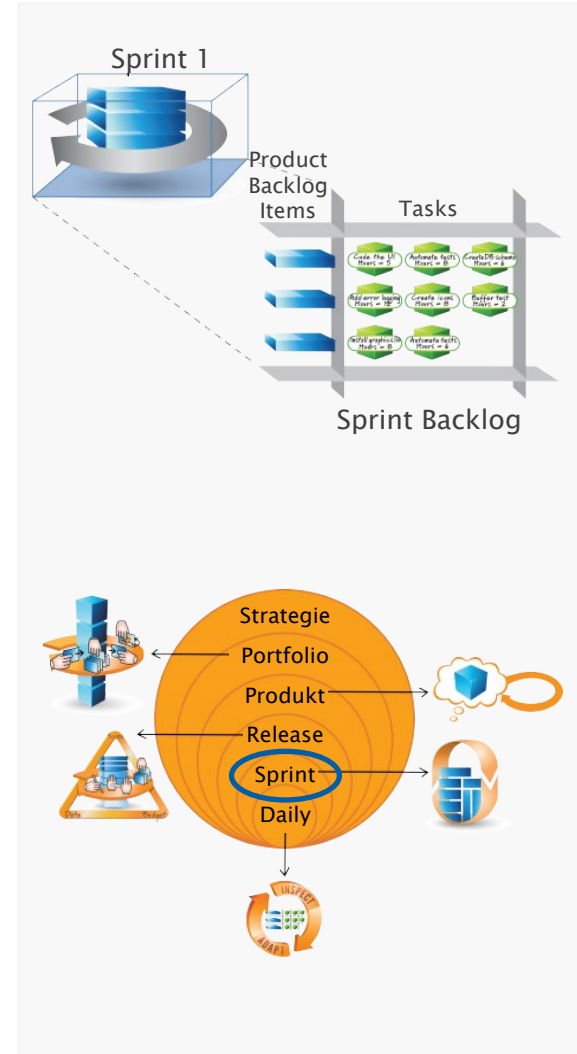
- ▶ Planungsebene für einzelne Produktversionen
- ▶ Besteht aus einer priorisierten Liste (backlog) einzelner Funktionalitäten
- ▶ Priorisierung nach Grad der Unterstützung der Product Road Map
- ▶ Release-Art bestimmt Zeitrahmen (major/minor/...)
- ▶ Verantwortlich:
Product Owner – Produkt- /Applikationsmanager



7. Planen

Planning Onion – Sprint

- ▶ Planungsebene für die einzelnen Iterationsschritte
- ▶ Unterstützt die Erreichung des Release (< 10 Sprints)
- ▶ Zeitrahmen (time box) hängt von Sprintdauer ab
- ▶ Verantwortlich:
Product Owner, Entwicklungsteam



Planning Onion – Daily

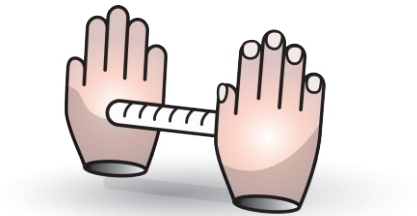
- ▶ Tägliches Planungsmeeting (daily scrum / daily stand-up)
- ▶ Unterstützt die Erreichung der Iterationsziele
- ▶ Inhalte
 - Was wurde seit dem letzten Meeting erreicht?
 - Was soll bis zum nächsten Meeting erreicht werden?
 - Welche Hindernisse treten dabei auf und müssen beseitigt werden?
- ▶ Verantwortlich: Scrum Master, Entwicklungsteam
- ▶ Aktualisierung von Taskboard und Burn down charts





Agile Scrum Foundation

8. Schätzen



Schätzen

- ▶ Voraussetzung für das Schätzen ist ein gleiches Verständnis aller Entwickler über die Bedeutung und Umsetzung einer User Story.
- ▶ Schätzungen werden ausschließlich vom Entwicklungsteam vorgenommen.
- ▶ Davon abweichende Vorstellungen von anderen Rollen, z. B. Product Owner oder Scrum Master spielen keine Rolle und werden nicht berücksichtigt.
- ▶ Insbesondere Kunden oder ihre Vertreter möchten die Entwicklung eines Produktinkrements häufig schnell abgeschlossen sehen und versuchen das gesamte Scrum Team bzgl. seiner Schätzungen zu beeinflussen.

Abstraktes Schätzen

- ▶ Geschätzt wird die Komplexität, **nicht** der zeitliche Aufwand.
- ▶ Komplexität wird in „Story Points“ ausgedrückt.
- ▶ Bezugsgrundlage ist die Erfahrung des Entwicklungsteams.

Vorteile des abstrakten Schätzens

- ▶ Komplexitätsschätzungen sind unabhängig von der Leistung des Teams und somit objektiv. Sie müssen nicht korrigiert werden.

Relatives Schätzen

- ▶ Verhältnismäßigkeit von Schätzungen – Komplexität und Story Points von User Stories stehen im selben Verhältnis zueinander.

Vorteile des relativen Schätzens

- ▶ Relative Schätzungen sind einfacher und erfolgen schneller als absolute Schätzungen.

Relatives Schätzen (Beispiel)

Die Komplexität einer mit 40 Story Points bewerteten Aufgabe ist fünf Mal so groß, wie die einer mit 8 Story Points bewerteten Aufgabe.

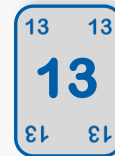
Genauigkeit

- ▶ Die Genauigkeit einer Schätzung nimmt mit steigender Komplexität ab.
- ▶ Zugelassene Story Points folgen der Systematik von Fibonacci-Zahlen.
- ▶ Dies sorgt für eine schnellere Entscheidungsfindung als durch die Verwendung von linearen Zahlen.



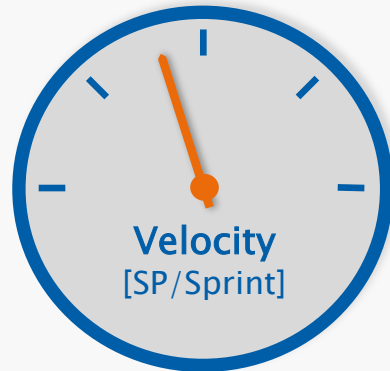
Fibonacci-Zahlen

- Fibonacci-Zahlen bilden eine Folge natürlicher Zahlen.
- Die Summe zweier aufeinanderfolgender Zahlen ergibt die jeweils nachfolgende Zahl.
- $f_n = f_{n-1} + f_{n-2}$,
wobei $f_1 = f_2 = 1$.
- Fibonacci-Zahlenfolge:
1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, ...
- Häufig verwendete, gerundete Zahlen gem. Fibonacci-Systematik:



Vom relativen zum absoluten Schätzwert

- ▶ Um von der Schätzung der Komplexität zur Schätzung des Aufwands zu kommen, werden Erfahrungswerte des Entwicklungsteams benötigt.
- ▶ Die „Velocity“ (Geschwindigkeit) eines Teams sagt aus, wie viele Story Points das Team in welchem Zeitraum entwickeln kann.
- ▶ Die Geschwindigkeit eines gut funktionierenden Entwicklungsteams bleibt nahezu konstant.



Beispiel

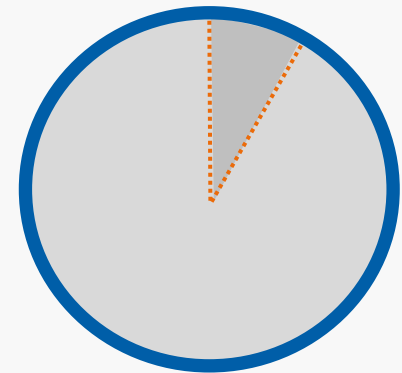
- Im letzten Projekt entwickelte ein Team 210 Story Points, verteilt über 10 Sprints.
- Die (durchschnittliche) Geschwindigkeit des Teams beträgt somit $210/10 = 21$.
- Die Sprints dauerten jeweils 3 Wochen.
- Pro Sprintwoche wurden (durchschnittlich) 7 Story Points entwickelt.

Ideale Arbeitsstunde (ideal hour)

- ▶ Eine „ideal hour“ ist eine Arbeitsstunde, innerhalb derer ein Entwickler sich ausschließlich seiner Entwicklertätigkeit widmet.
- ▶ Innerhalb dieser „idealen“ Arbeitsstunde gibt es keine Pausen, Unterbrechungen oder Ablenkungen.
- ▶ Aufwände werden von Entwicklerteams in „ideal hours“ angegeben, da dieses Vorgehen erfahrungsgemäß dem Denkansatz von Entwicklern entspricht.
- ▶ Allgemein werden für einen normalen 8-Stunden-Arbeitstag (Vollzeit) 6 „ideal hours“ angesetzt.
- ▶ Konkret kann jedem Entwickler unter Berücksichtigung der individuellen Situation eine bestimmte Anzahl von „ideal hours“ zugewiesen werden.

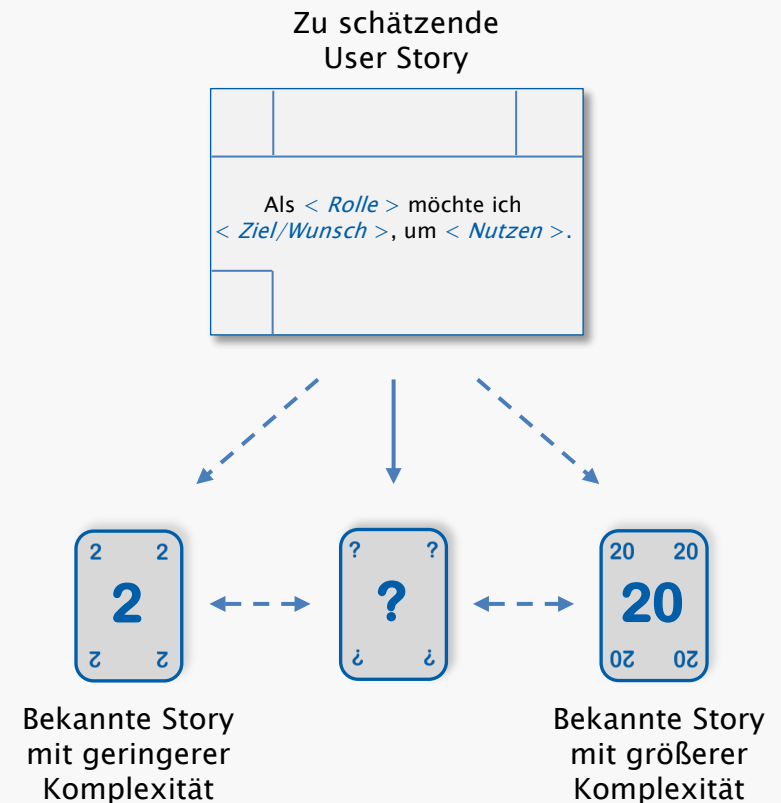
Ideal day

Statt „ideal hours“ können auch „ideal days“ verwendet werden.



Triangulation

- ▶ Schätzungen erfolgen über Vergleiche der Komplexität der zu schätzenden User Story mit anderen, bereits bekannten User Stories (Referenz - User Stories).
- ▶ Als Referenz werden zwei bereits bekannte Stories verwendet: Eine mit geringerer, die andere mit größerer Komplexität als die zu schätzende User Story.
- ▶ Auf Triangulation basiert die Anwendung verschiedener Schätzverfahren, z. B. Planning Poker.



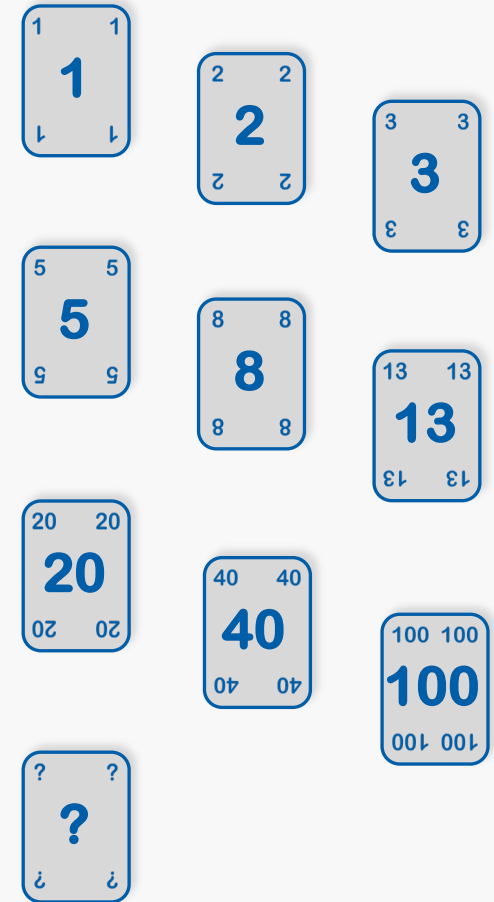
Planning Poker (1)

- ▶ Mit Planning Poker können Aufgaben unterschiedlicher Komplexität in einer akzeptablen Zeit und mit guter Genauigkeit geschätzt werden.
- ▶ Die Mitglieder des Entwicklungsteams geben unabhängig und damit auch unbeeinflusst voneinander Schätzungen für die Komplexität einer Entwicklungsaufgabe ab.
- ▶ Das Ergebnis ist vom gesamten Entwicklungsteam akzeptiert und gültig.

Planning Poker (2)

Ablauf des Planning-Poker-Verfahrens (1)

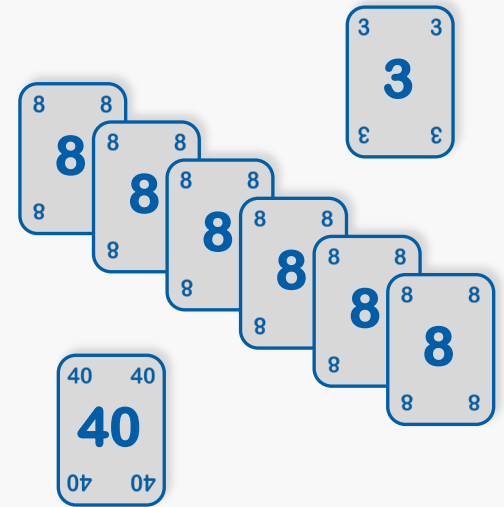
- ▶ Der Product Owner stellt dem Entwicklungsteam die zu schätzende User Story vor.
- ▶ Der Product Owner beantwortet Fragen des Entwicklungsteams zu der Story.
- ▶ Jedes Mitglied des Entwicklungsteams erhält einen Kartensatz und wählt verdeckt eine Karte aus, die es bzgl. der Komplexität der Story für angemessen hält.
- ▶ Anstelle der exakten Fibonacci-Zahlen werden häufig gerundete Zahlen verwendet.



Planning Poker (3)

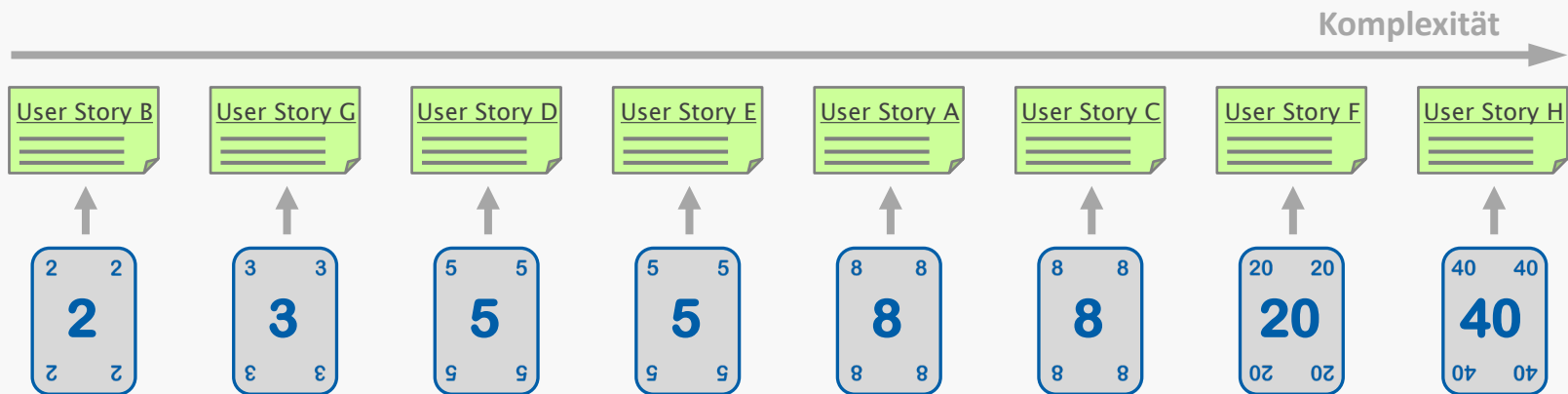
Ablauf des Planning-Poker-Verfahrens (2)

- ▶ Alle Karten werden gleichzeitig aufgedeckt.
- ▶ Die Entwickler mit der niedrigsten und höchsten Schätzung begründen ihre Entscheidung.
- ▶ Der Schätzvorgang wird so lange wiederholt, bis ein Konsens im Entwicklungsteam gefunden oder die Abweichung der einzelnen Schätzungen untereinander gering ist.



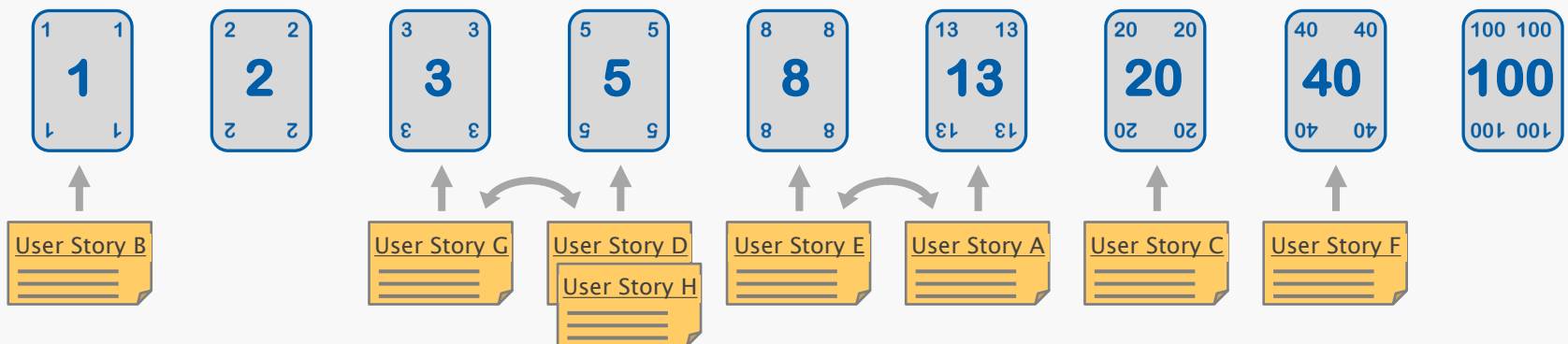
Estimation Game (Affinity Estimation)

- ▶ Es werden die einzelnen User Stories miteinander verglichen und in eine Reihenfolge bzgl. ihrer Komplexität gebracht.
- ▶ Danach erfolgt die Schätzung ihrer Komplexität mit Hilfe der Fibonacci-Zahlen durch ein Mitglied des Entwicklungsteams mit kurzer Begründung und ohne Diskussion.
- ▶ Dies ermöglicht unerfahrenen Entwicklungsteams langwierige Diskussionen zu vermeiden und so zu vielen Schätzungen in kurzer Zeit zu kommen (ca. 100 User Stories / h).



Magic Estimation

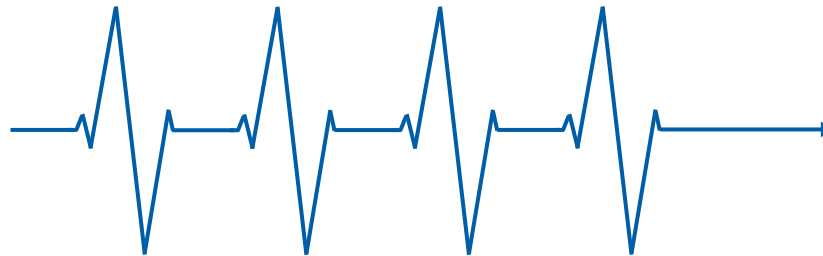
- ▶ Zuerst werden die Karten mit der Fibonacci-Zahlenfolge ausgelegt.
- ▶ Die User Stories werden unter den Mitgliedern des Entwicklungsteams verteilt und jedes Mitglied ordnet stillschweigend seine Karten der Fibonacci-Zahlenfolge zu.
- ▶ Jedes Mitglied des Entwicklungsteams kann die Zuordnung von Karten verändern.
- ▶ Karten, deren Zuordnung umstritten ist, legt der Product Owner beiseite; sie werden anschließend gemeinsam geklärt.
- ▶ Diese erste Schätzung sollte später nochmal genauer durchgeführt werden.





Agile Scrum Foundation

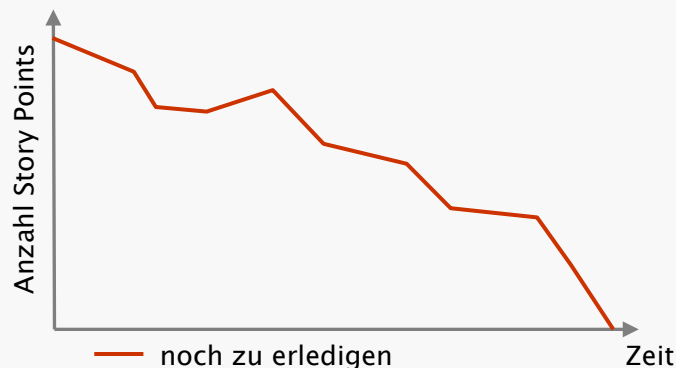
9. Monitoring



Burn down Chart / Burn up Chart

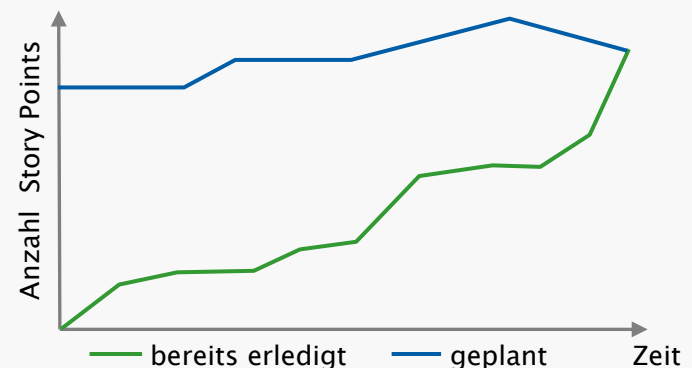
Burn down Chart

- ▶ stellt noch zu erledigende Arbeiten über einen Zeitverlauf dar.
- ▶ bereits erledigte Arbeiten sind irrelevant.
- ▶ betrachtet Story Points, Arbeitszeiten (Aufwände) oder Aufgaben (Anzahl).



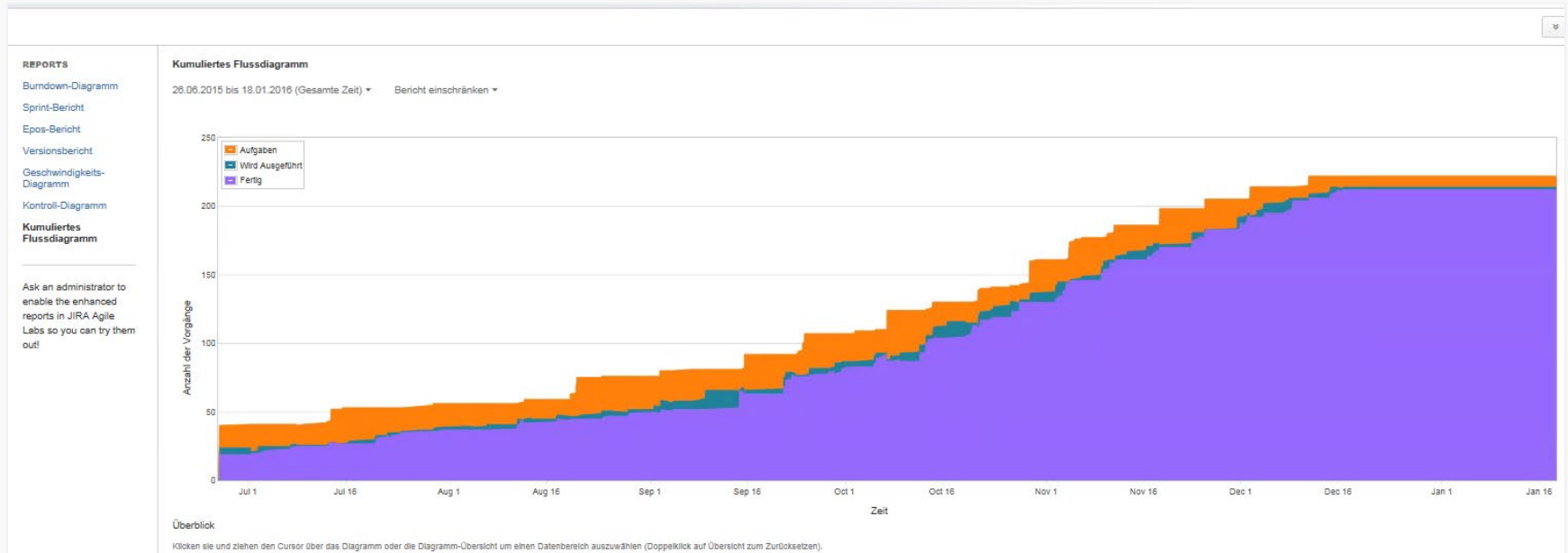
Burn up Chart

- ▶ stellt bereits erledigte Arbeiten über einen Zeitverlauf dar.
- ▶ noch zu erledigende Arbeiten sind irrelevant.
- ▶ betrachtet Story Points, Arbeitszeiten (Aufwände) oder Aufgaben (Anzahl).



Kumuliertes Flussdiagramm

Beispiel eines Burn up Charts als kumuliertes Flußdiagramm (in JIRA)



Information Radiator

























- ▶ Große Tafel mit allerlei gedruckten, gezeichneten, handschriftlichen oder elektronisch (Displays) dargestellten Informationen über das Projekt.
- ▶ Ist an einem öffentlichen Ort aufgestellt, z. B. im gemeinsamen Entwicklungsraum (shared workspace), so dass alle Beteiligten sie jederzeit sehen können.
- ▶ Unterstützt das Entwicklungsteam bei der Konzentration auf das Wesentliche.
- ▶ Verkörpert Transparenz auf einen Blick.
- ▶ Auch bekannt als Big Visible Chart (BVC).

Beispiele

- verschiedene Charts
- Taskboards
- Planungen
- Tests
- Integrationsstatus
- aktuelle Geschwindigkeit
- ...

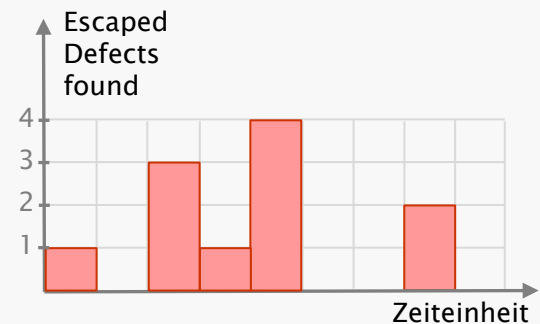
Niko-niko Calendar

- ▶ Besondere Ausprägung eines information radiator, auch „mood board“ genannt.
- ▶ Ermöglicht Mitgliedern des Entwicklungsteams am Ende des Arbeitstages ihren Gefühlszustand über den Tag darzustellen.
- ▶ Misst die Zufriedenheit und Motivation des Teams und gibt diese objektiv wieder.
- ▶ Ein zufriedeneres Team ist ein leistungsfähigeres Team.
- ▶ Aus dem japanischen „niko“ – lächeln, niko-niko – „Smiley“.

	Mo	Di	Mi	Do	Fr
Lisa					
Lea					
Paul					
Eva					
Kim					

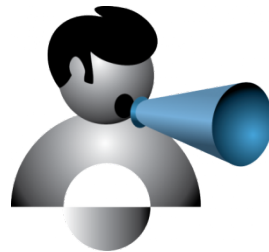
Escaped Defect

- ▶ Ein Fehler (Bug) in der Entwicklung, der in der Qualitätsprüfung, z. B. während des Testens, nicht gefunden wurde. Er kann ...
 - auch während des operativen Betriebs weiterhin verborgen bleiben oder
 - sich während des operativen Betriebs in einer Funktionsstörung äußern.
- ▶ Ein Escaped Defect bzw. seine Auswirkung wird häufig erst von Anwendern entdeckt.
- ▶ Die Anzahl entdeckter Escaped Defects ist ein Maßstab für die Güte der Qualitätssicherung.
- ▶ Die Behebung eines Escaped Defects ist meist kostspielig, da ein produktives Release geändert werden muss.



Agile Scrum Foundation

10. Rahmenbedingungen



Voraussetzungen für Scrum

- ▶ **Agiles Scrum Team**
 - Erfahrener Scrum Master
 - Ausreichend befugter Product Owner
 - Interdisziplinär und selbstorganisiertes Entwicklungsteam
- ▶ **Agiles Produkt**
- ▶ **Agile Organisation**
- ▶ **Agiler Kunde**
 - Stellt Anforderungen
 - Arbeitet eng mit Product Owner zusammen
 - Führt Tests durch
 - Akzeptiert Berechnung nach tatsächlichem Aufwand

Agile oder nicht agile Produktentwicklung?

- ▶ Produkte können determiniert sein.
- ▶ Determinierte Produkte eignen sich nicht für eine agile Entwicklung.
- ▶ Beispiele:
 - Vom Gesetzgeber vorgeschriebene Produkte/Funktionen
 - Über einen Werkvertrag beauftragte Produkte

Scrum Management

- ▶ Das Management ist verantwortlich, das Wertesystem von Scrum in der Organisation zu implementieren und gedeihen zu lassen.
- ▶ Es unterstützt die Arbeitsweise von Teams nach agilen Prinzipien und Methoden
 - selbstorganisiert – nicht hierarchisch
 - eigenverantwortlich
- ▶ Ein agiler Manager ist Coach und Mentor. Er ...
 - ermöglicht die Weiterentwicklung aller Teammitglieder.
 - geht selbst mit gutem Beispiel voran und lebt agile Werte vor, z. B. Transparenz, Respekt, Entscheidungen treffen, Mut zeigen, Kommunikation, ...

Perspektiven des agilen Managementmodells

- Gib Menschen Energie
- Ermächtige Teams
- Justiere die Rahmenbedingungen
- Entwickle Kompetenz
- Lasse Strukturen wachsen
- Verbessere alles

Entwicklungsteams zusammenstellen (1)

Zusammenstellung ausgerichtet an organisatorischen Strukturen

- ▶ Ein Entwicklungsteam ist interdisziplinär aufgestellt. Es müssen alle für die Entwicklung des Produkts benötigten Fähigkeiten im Team vorhanden sein (cross functionality).
- ▶ Dies ist innerhalb einer Organisationseinheit (fast) **nie** der Fall.

Entwicklungsteams zusammenstellen (2)

Zusammenstellung ausgerichtet an technischen Komponenten

- ▶ Eine Ausrichtung an technischen Komponenten ist möglich, wenn die Entwicklung des Produkts technisch eindeutig und klar abgegrenzt ist.
- ▶ Ist dies nicht gegeben, lässt sich die Forderung nach interdisziplinärer Aufstellung im Team nicht erfüllen.
- ▶ Eine Ausrichtung an technischen Komponenten ist **gelegentlich** möglich.

Entwicklungsteams zusammenstellen (3)

Zusammenstellung ausgerichtet an fachlichen Themen

- ▶ Eine Ausrichtung an fachlichen Themen schließt alle für die Produktentwicklung erforderlichen Aspekte ein.
Einzelne Themen können dann einzelnen Scrum Teams zugeordnet werden.
- ▶ Eine Ausrichtung an fachlichen Themen ist i. d. R. **möglich** und deshalb zu empfehlen.

Beispiel: Produkt „Bildungsreise“

- Team 1:
Marketing und Vertrieb
- Team 2:
Organisation der Reise
- Team 3:
Inhaltliche Ausarbeitung der Bildungsthemen

Räumlich verteiltes Scrum Team

Umgang mit einem räumlich verteilten / verstreuten Scrum Team (distributed team)

- ▶ Stärkung des Teamgedankens durch häufige persönliche Kontakte und Zusammenarbeit.
- ▶ Regelmäßige, gemeinsame Veranstaltungen und Reisekosten sind Investitionen in die Stärkung des WIR-Gefühls eines Teams und damit in die Teamleistung.
- ▶ Vermeidung von Konflikten, insbesondere durch die räumliche Verteilung bedingte Konflikte, z. B.
 - kulturelle Konflikte
 - regionale Konflikte
 - sprachliche Konflikte
 - soziale Konflikte

Neue Scrum-Teams

- ▶ Die Notwendigkeit mehrerer Teams ist bei Projektbeginn nicht immer bereits bekannt.

Erstellen neuer Teams

- ▶ Erstellen vollständig neuer Teams
 - Vollständig neu erstellte Teams sind mit dem zu erstellenden Produkt nicht vertraut und arbeiten zu Beginn ineffizient.
 - Nur in erfahrenen Organisationen sollten Teams vollständig neu erstellt werden.

Aufteilen bestehender Teams

- ▶ Aufteilen eines bestehenden Teams und Hinzufügen neuer Mitglieder (split & seed)
 - 😊 Projektkenntnisse werden auf alle neuen Teams übertragen.
 - ☹️ Ein funktionierendes Team wird auseinander gerissen.
- ▶ Hinzufügen neuer Mitglieder zu einem Team und anschließendes Aufteilen (grow & split).
Danach können ggf. weitere Mitglieder hinzugefügt werden.

Skaliertes Scrum

- ▶ Bei größeren Projekten sind typischerweise mehrere Scrum-Teams beteiligt.
 - ▶ Pro Team gibt es einen Product Owner und einen Scrum Master.
 - ▶ Pro Team gibt es ein (Team) Product Backlog (selected product backlog) mit den aus dem (Haupt) Product Backlog ausgewählten Aufgaben für das jeweilige Team.
 - ▶ Die Product Owner kommunizieren regelmäßig untereinander und stimmen sich über die Verteilung von Aufgaben auf die jeweiligen (Team) Product Backlogs ab.
- ▶ Für das (Haupt) Product Backlog ist ein eigener Product Owner verantwortlich.
 - ▶ Abhängigkeiten zwischen den einzelnen Teams sind möglichst zu vermeiden
 - Voraussetzungen können im vorausgehenden Sprint erledigt werden.
 - Abhängigkeiten werden durch definierte Schnittstellen verringert (interface driven design).

Scrum of Scrums (SoS)

- ▶ Sind mehrere Scrum Teams an einem Projekt beteiligt, ist die Kommunikation und Zusammenarbeit zwischen den Teams zu organisieren, insbesondere bei bestehenden Abhängigkeiten.
- ▶ Einzelne Vertreter der beteiligten Teams treffen sich regelmäßig, um die Zusammenarbeit zu organisieren (Scrum of Scrums).
- ▶ Wichtigster Aspekt sind die Schnittstellen zwischen den Teams.
- ▶ Unterschiedliche Teilnehmer am Scrum of Scrums sind sinnvoll, um durch entstandene Routine ineffiziente Meetings zu vermeiden.

Release Train

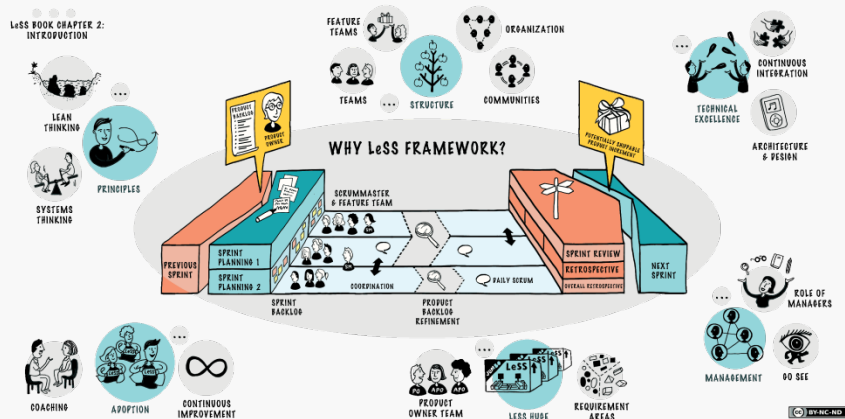
Bei großen Produkten und vielen Scrum Teams wird die Organisation der Zusammenarbeit auch „Release Train“ genannt.

Scrum of Scrums – Inhalte

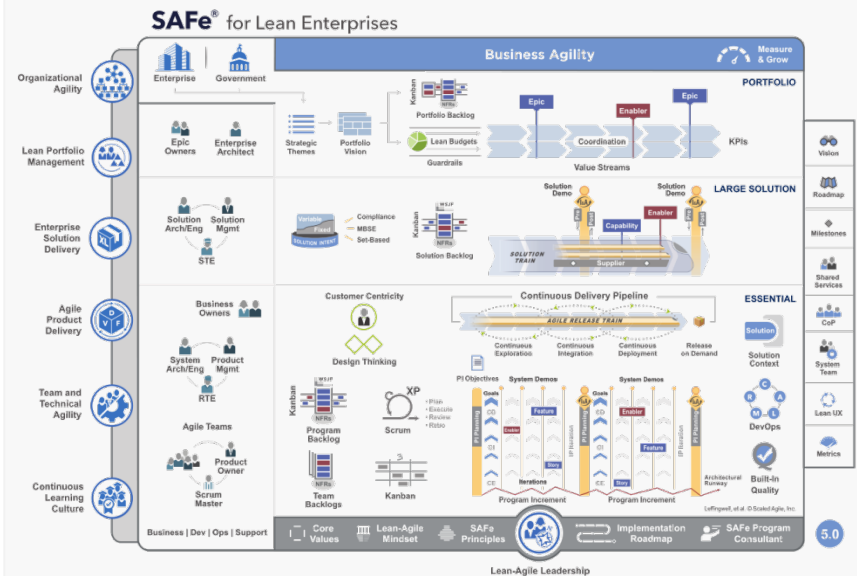
- ▶ In einem SoS-Meeting beantwortet jeder Vertreter eines Teams stellvertretend für sein Team die Fragen (analog zum Daily Scrum)
 - Was hat mein Team seit dem letzten Scrum of Scrums getan?
 - Was wird mein Team bis zum nächsten Scrum of Scrums voraussichtlich erledigen?
 - Was behindert mein Team bei seiner Arbeit?
 - Beeinflussen oder behindern Tätigkeiten meines Teams ein anderes Team bei seiner Arbeit?

Weitere Skalierungsansätze

- ▶ Large-Scale Scrum – LeSS™
less.works



- ▶ Scaled Agile Framework – SAFe™
www.scaledagileframework.com



- ▶ Nexus™
www.scrum.org/resources/nexus-guide

- ▶ Scrum@Scale™
scrumatscale.scruminc.com

Vertragstypen

Zeit & Mittel / feste Einheit

(Time & means / fixed unit)

- ▶ Berechnung der Projektleistung nach
 - tatsächlicher Anzahl von Personentagen / -stunden oder
 - tatsächlicher Anzahl von Sprints. Teamgröße und Sprintdauer sind dabei zu vereinbaren.
- ▶ Häufig als Dienstvertrag bezeichnet.
- ▶ Eine Berechnung nach tatsächlich erbrachtem Aufwand ist mit einem agilen Ansatz vereinbar.

Werkvertrag / Festpreis

(fixed price)

- ▶ Ein Festpreis basiert auf einem bekannten und vereinbarten Projektumfang.
- ▶ Festpreise können vom Gesetzgeber vorgeschrieben sein.
- ▶ Festpreise sind letztlich mit einem agilen Ansatz **nicht** vereinbar.

Scrum – Missverständnisse

- ▶ Es gibt in Scrum keine Planung.
- ▶ Der Kunde weiß nie, wann er was bekommt.
- ▶ Mein Code wird durch andere „verschmutzt“.
- ▶ Mit Scrum macht jeder nur noch was er/sie will.
- ▶ Es gibt keine Führung mehr.
- ▶ Hurra – wir müssen nicht mehr dokumentieren!



Scrum – Typische Fehler

- ▶ Scrum Team nutzt gewonnene Erkenntnisse nicht für Verbesserungen des Entwicklungsprozesses.
 - ▶ Scrum Master gibt dem Team Anweisungen, WIE es seine Arbeit zu erledigen hat.
 - ▶ Product Owner ist nicht bevollmächtigt oder entscheidungsfähig.
 - ▶ Mitglieder des Entwicklungsteams übernehmen nicht die vielfältigen Aufgaben eines Sprints.
 - ▶ Management bekennt sich nicht zu Scrum.
- ▶ Organisation stellt den erforderlichen (kulturellen) Rahmen nicht bereit.
 - ▶ Scrum Prozess bzw. Regeln werden nicht eingehalten. Kleine Auswahl:
 - Der Klassiker:
Daily Scrum 2x pro Woche (!)
 - Meetings werden nicht regelmäßig durchgeführt.
 - Meeting startet und endet nicht pünktlich (timebox).
 - Ablauf und Inhalte der Meetings werden nicht eingehalten.
 - Neue User Stories werden in laufenden Sprint eingebracht.

Scrum – 10 Gebote für den Erfolg

- ▶ Eine klare, verständliche und begeisternde Vision.
- ▶ Ein gepflegtes, nach Business Value priorisiertes Product Backlog.
- ▶ Vom Team gemeinsam geschätzte Backlog Items.
- ▶ Daily Scrums täglich und time boxed abhalten.
- ▶ Burn up Chart laufend pflegen und interpretieren.
- ▶ Sprints nicht vom Management / Kunden stören lassen.

- ▶ Das Team liefert Produkte / Produktinkremente gemäß „Definition of Done“.
- ▶ Ein produktives, gemeinsames Sprint Review abhalten.
- ▶ Sprint Retrospektiven mit Fokus auf Verbesserungen im Arbeitsprozess des Teams und der Organisation.
- ▶ Gemeinsame Problemlösung statt gegenseitige Schuldzuweisungen.

