

Vorgehensmodelle

- Definieren notwendiges systematisches Vorgehen
- Ein Vorgehensmodell definiert den tatsächlichen Ablauf in der Softwareentwicklung
- Ein Vorgehensmodell strukturieren den Entwicklungsprozess „im Großen“ – die Feinheiten muss man im Unternehmen selbst festlegen
- Vorgehensmodelle bauen auf den grundlegenden Arbeitsschritten Analyse, Entwurf, Implementierung und Test auf

Vorgehensmodelle

Darstellung der Grundlegenden Bereiche des Softwareengineering



Vorgehensmodelle



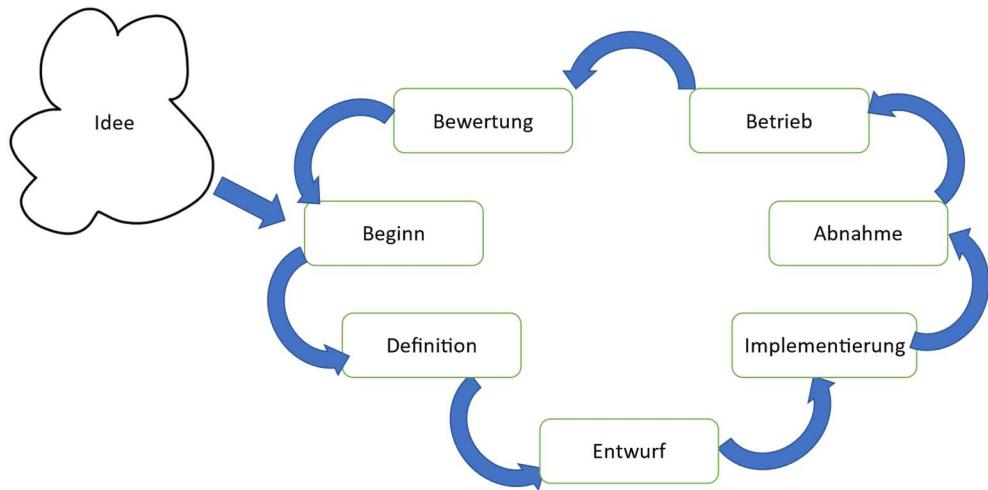
Build-and-Fix-Cycle

- Bekanntes Vorgehensmodell bei allen Programmierern
- Einfachstes Vorgehensmodell, dass sich nur für ein sehr kleines Team eignen kann
- Jemand hat eine Idee für eine Software, implementiert diese nach seinen Vorstellungen
- System wird so lange entwickelt, bis es den Qualitätsansprüchen des Programmierers genügt
- Treten Änderungswünsche auf, werden diese vom Programmierer eingebbracht.
- Es existiert keinerlei Dokumentation, kein-kaum strukturiertes Vorgehen
- Wartung der Software ist meist nur durch den Programmierer möglich

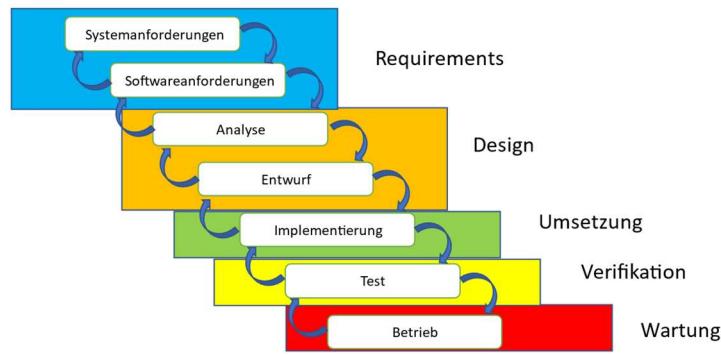
Software-Life-Cycle

- Software-Life-Cycle ist die grundlegende Idee aller Vorgehensmodelle, die ein strukturiertes Vorgehen bei der Erstellung von Software vorsieht.
- Durchführung aller notwendigen Arbeitsschritte zumindesten mal
- Änderungen von Anforderungen erst im Zuge eines neuen Projektes
- Größtes Problem ist, dass es nicht möglich ist ein oder mehrere Schritte zurück zu gehen.

Software-Life-Cycle



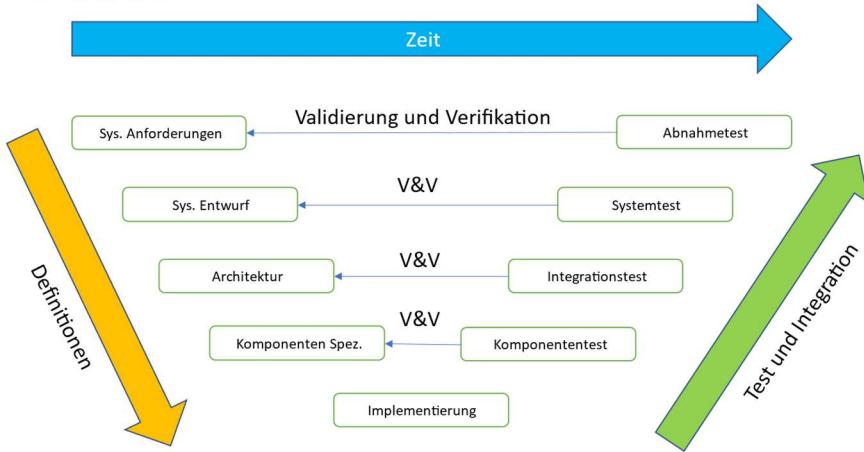
Wasserfallmodell



Wasserfallmodell

- Verbesserung des Software-Life-Cycle
- Lässt einen Rückschritt von einem Arbeitsschritt in den vorigen zu
- Aber 1: Ein Arbeitsschritt erst dann abgeschlossen, wenn ALLE vorhergehenden abgeschlossen sind
- Aber 2: Das Wasserfallmodell unterstützt nicht den Rückschritt, wenn ein Arbeitsschritt bereits abgeschlossen war
- → Diese Bedingungen soll zu einer Minimierung des Risikos für den nächsten Arbeitsschritt führen
- Gut für Projekte, in denen Arbeitsschritte gut voneinander getrennt werden können
- Gut wenn Arbeitsgruppen klein sind, die einen Arbeitsschritt bearbeiten → Sonst aufgrund der Anzahl und der untersch. Qualifikationen zu schwierig
- Im Allgemeinen nicht für komplexe Projekte geeignet, da man damit kaum auf geänderte Anforderungen reagieren kann

V-Modell



V-Modell

- Das V-Modell enthält für jede „Phase“ in der die Richtung nach unten zeigt eine dazu korrespondierende Testphase in Richtung nach oben.
- Das V-Modell strukturiert den Software-Entwicklungsprozess ähnlich dem Wasserfallmodell in einer sequentiellen Abfolge
- Als Fortschritt betont es die Zusammengehörigkeit der Designphasen und den dazu gehörigen Testphasen
- Allerdings ist das V-Modell ebenso wie das Software-Life-Cycle Modell und das Wasserfallmodell anfällig für Fehler in frühen Projektphasen oder Change-Requests
- Bei später Entdeckung von Fehlern oder späteren Änderungen ist der Aufwand für eine Korrektur hoch
- V-Modell eignet sich für kleine Projekte

Spiral-Modell



Spiral-Modell

- Gesamte Prozess in vier Phasen gegliedert, die mehrmals durchlaufen werden
- In jedem Durchlauf werden bestimmte Produkte/Module entwickelt, die auf das Produkt/Modul im Vorlauf aufbauen und als Basis für den nächsten Durchlauf dienen
- Vier Phasen:
 - Zielbestimmung
 - Risikoanalyse
 - Arbeitsschritte durchführen
 - Nächste Phase planen

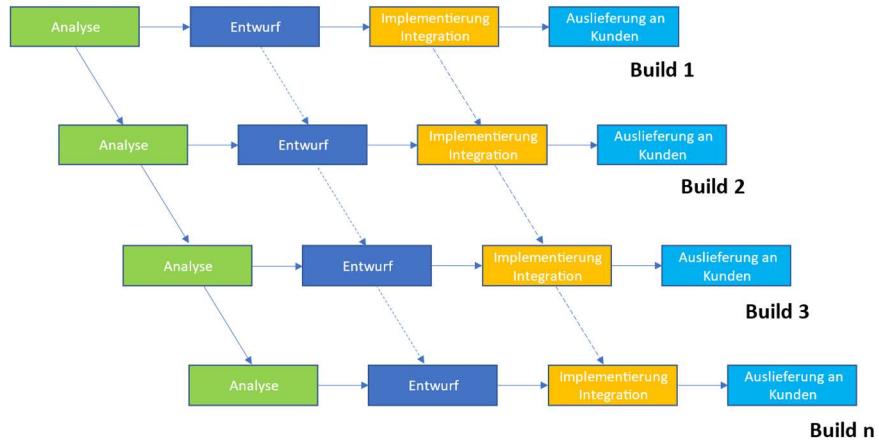
Spiral-Modell

- Zielbestimmung: Jede Phase beginnt mit der Bestimmung von Zielen. Es werden auch Entwurfsvarianten und Restriktionen (z.B. aufgrund von Zeit) festgehalten
- Risikoanalyse: Ziele und Alternativen werden aufgrund von Restriktionen bewertet. Für gefundene Risiken werden Lösungsstrategien zur Beseitigung der Ursachen entwickelt
- Arbeitsschritte durchführen: Stellt die Implementierungsphase für das jeweilige Modul dar
- Nächste Phase planen: Basierend auf das Ergebnis des Reviews eines Durchlaufs, wird die nächste Phase geplant

Spiral-Modell

- Kann für große und komplexe Projekte verwendet werden – weil Risikogesteuert
- Anzahl der Durchläufe ergibt sich während des Projektes und wird durch die auftretenden Risiken bestimmt
- Zeit- und Kostenplanung ist deshalb nur schwer zu erstellen
- → Erfahrene Projektleiter notwendig!!
- Beim Zaghaften Vorgehen wird das Projekt verlängert, bei zu schnellem Vorgehen werden Risiken vernachlässigt

Das inkrementelle Modell



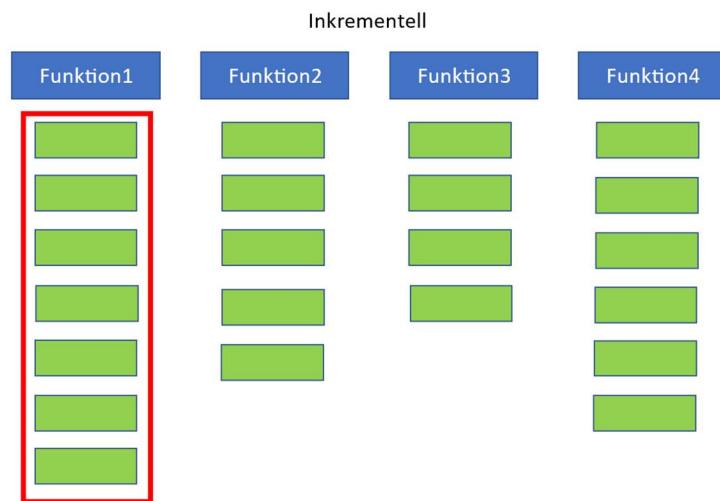
Das inkrementelle Modell

- Wasserfall und V-Modell haben den Nachteil, dass eine Auslieferung des Produktes nur am Ende erfolgt und dauert. Bei Auslieferung kann es sein, dass sich die Anforderungen bereits geändert haben und das Produkt gleich überarbeitet werden muss
- Im inkrementellen Modell startet man mit dem Entwicklungsprozess sobald eine bestimmte Menge an Anforderungen da sind.
- Je mehr Anforderungen hinzukommen, desto mehr wird das System schrittweise gebaut
- Die Entwicklung erfolgt Stufenweise und basiert auf die Erfahrungen der Entwickler und des Kunden
- Im inkrementellen Modell werden Funktionen „der Reihe nach“ implementiert.
- Gutes Modell, wenn der Kunde seine Anforderungen noch nicht ganz überblickt hat
- Entwicklung wird hauptsächlich durch den Code getrieben – wichtig sind lauffähige Systeme zu generieren
- Es ist möglich, Teile des Systems beim Kunden einzuführen → muss nicht bis zur „kompletten“ Fertigstellung warten
- Auf neue Anforderungen kann besser und schneller reagiert werden
- Nachteil: Dieses Modell benötigt eine strategische Planung (welche Requirements wann) und mehr Ressourcen

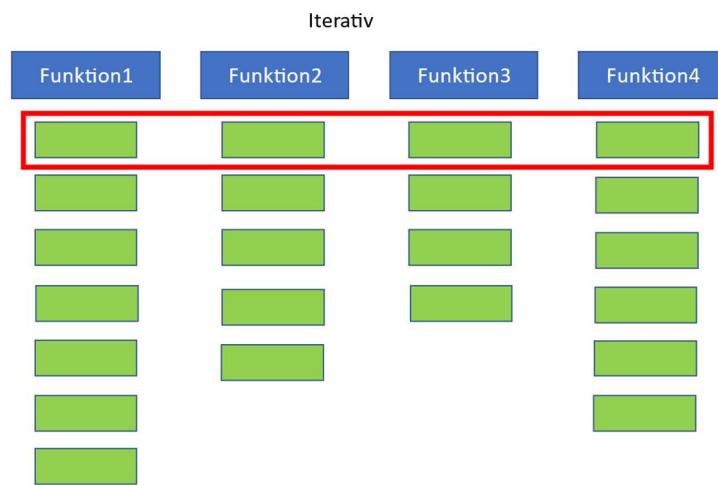
Das iterative Modell

- Das iterative Modell funktioniert vom Ablauf her ähnlich wie das inkrementelle Modell
 - Im Gegensatz zum inkrementellen Modell wird nicht ein Feature nach dem anderen realisiert, sondern Teile von mehreren Features, die stetig verbessert werden
 - Vorteil: Da das Produkt stufenweise erstellt wird, werden Probleme früh bemerkt und man kann schneller darauf reagieren
 - Nachteil: Es ist sehr starr und kann zu höheren Kosten führen

Inkrementell vs. Iteratives Modell



Inkrementell vs. Iteratives Modell



Agile Softwareentwicklung

- Erhöht Transparenz und Entwicklungsgeschwindigkeit im Projekt
- → Führt zu einem schnelleren Einsatz des Systems und soll so Risiken und Fehlentwicklungen minimieren
- Entwurfsphase wird auf ein Mindestmaß reduziert, um so früh wie möglich mit der Entwicklung zu beginnen
- Ausführbare Software wird in regelmäßigen kurzen Abständen mit dem Kunden abgestimmt – damit kann man schneller auf Kundenwünsche reagieren
- Agile Entwicklung ist eine Kombination aus iterativer- und inkrementeller Entwicklung
- Agile Softwareentwicklung ist ein Sammelbegriff für eine Reihe von Methoden und Praktiken

Agile Leitsätze

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

Agile Frameworks

- Scrum
- Extreme Programming (XP)
- Kanban
- Feature Driven Development (FDD)
- Adaptive Software Development (ASD)
- Crystal
- Lean Startup
-

Scrum

- Umsetzung von Lean Development
- Beruht auf Erfahrungen im Entwicklungsprozess
- Projekt ist „zu groß“, um es genau durch zu planen
- Mit vielen „Zwischenlösungen“ werden Unklarheiten beseitigt
- Scrum besteht aus wenigen Regeln, die vier Ereignisse (=Meetings – haben feste Zeitfenster - **Sprint Planning, Daily Scrum, Sprint Review, Sprint Retrospektive**), drei Artefakte (**Product Backlog, Sprint Backlog, Product Increment**) und drei Rollen (**Product Owner, Entwickler, Scrum Master**) definieren
- Der langfristige Plan (Product Backlog) wird kontinuierlich verbessert, Detailplan (Sprint Backlog) wird für jeden Zyklus (Sprint) neu definiert

Scrum

- In Scrum sind Requirements eher Eigenschaften der Anwendersicht als Pflichten- u. Lastenhefte
- Liste der Anforderung ist das sog. (Product) Backlog
- Diese werden in Intervallen von normalerweise 2 Wochen abgearbeitet (=Sprints)
- Am Ende eines Sprints steht Produkt vor einer Auslieferung (**Product Increment**)
- Ein Scrum Team sollte in der Lage sein alle Vorgaben, innerhalb eines Sprints zu erreichen
- Ein ideales Teammitglied ist sowohl Spezialist als auch guter All-Rounder
- Ein Scrum Team muss den Umfang der jeweiligen Requirements im Backlog abschätzen
- Arbeitsschritte werden auf einzelne Tasks heruntergebrochen – Ergebnis ist das sog. Sprint-Backlog

Product Owner (PO)

- Ist für Erfolg des Produktes verantwortlich
- Gestaltet das Produkt, sodass es am Ende den meisten Nutzen hat
- Er erstellt und priorisiert die zu entwickelten Projektentscheidungen
- Produkteigenschaften kommen alle ins und aus dem Backlog

Entwickler

- Entwickler liefern Software anhand der vom PO gewünschten Reihenfolge

Scrum Master

- Der Scrum Master ist dafür verantwortlich, dass der Scrum Prozess erfolgreich durchgeführt wird
- Diese Position wird in vielen Fällen auch vom Projektleiter übernommen
- Der Scrum Master führt Regeln ein, um den Scrum Prozess zu realisieren und beseitigt Probleme, die dabei hindern
- Scrum Master moderiert die Sprint Retrospektive und oft auch das Sprint Planning und Backlog Refinement („StandUp“ Meetings)

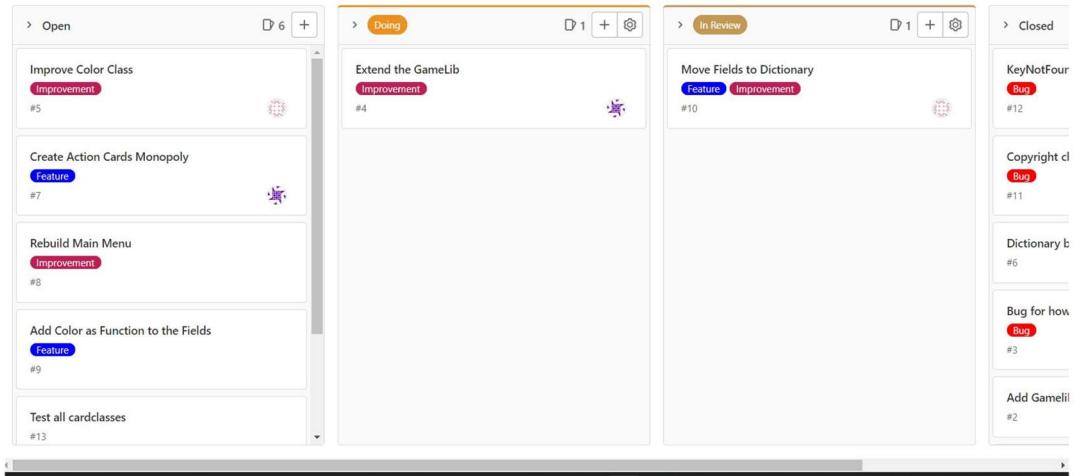
Scrum Ereignisse – Sprint Planning

- Was kann im kommenden Sprint entwickelt werden?
 - PO bereitet Backlog im Product Backlog Refinement vor, sodass er diese dem Team für einen Sprint vorlegen kann
 - PO entscheidet mit Team, wann neue Funktionalitäten fertig sind (**Definition of Done**)
 - Ziel ist das Produktinkrement
 - Team entscheidet dann, welche Backlog-Einträge es im Sprint liefern kann
- Wie wird die Arbeit im kommenden Sprint erledigt?
 - Team plant im Detail das Vorgehen zum Erreichen der prognostizierten Backlog-Einträge
 - Ergebnis ist das Sprint Backlog, dass oft in Form eines **Taskboard (Kanban Board)** dargestellt ist

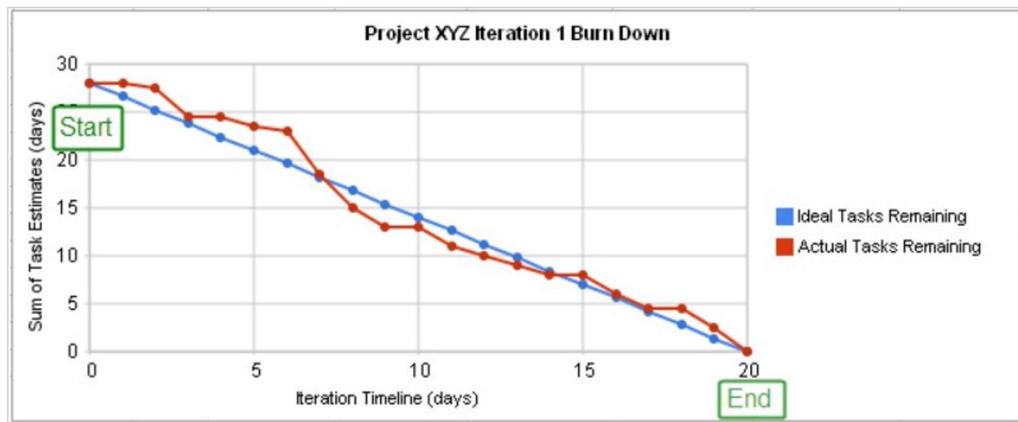
Scrum Ereignisse – Daily Scrum

- 15-minütiges Meeting mit Scrum Master und PO
- Zweck ist ein Informationsaustausch – es wird ein Überblick über den Arbeitsstand gegeben
- Dazu wird mittels Taskboard erläutert, welche Tasks (User Stories) umgesetzt werden konnten (und welche nicht oder nur Teilweise) und welche als nächstes erreicht werden möchten
- Im Taskboard wird vom jeweiligen Entwickler auch geschätzt, wie lange ein Task vermutlich dauern wird – die Schätzung erfolgt zumeist in Stunden erfolgen (man kann zusätzlich die Anzahl der Stunden auf Fibonacci Zahlen limitieren – d.h. dass nur Stunden, die aus der Fibonacci Reihe kommen vergeben werden können – dient um komplexe Aufgaben nach Aufwand besser zu parametrieren)
- Am Ende eines Tages kann der Scrum Master durch einen sog. **Burn Down Chart** sehen, wie erfolgreich, die Umsetzung anhand der Schätzung waren

Scrum Taskboard - Beispiel



Burn – Down - Chart



Scrum Ereignisse – Sprint Review

- Am Ende des Sprints wird Inkrement überprüft und das Backlog angepasst
- PO muss entwickelte Funktionalität begutachten
- Nicht erfüllte oder fertige User Stories wandern ins Backlog zurück

Scrum Ereignisse – Sprint Retrospektive

- Scrum Team reviewed seine Arbeitsweise und bespricht mögliche Verbesserungen im Sprint
- Manche Teams nutzen eine Liste mit Hindernissen und Verbesserungsmaßnahmen (=**Impediment Backlog**)
- Steht wie das Sprint Review am Ende eines Sprints– kann sein, dass es nicht nach jedem Sprint gemacht wird

Scrum Artefakte - Product Backlog

- Geordnete Liste von Anforderungen des Produkts
- Für Pflege ist PO verantwortlich
- Backlog ist dynamisch und auch nicht unbedingt von Beginn an vollständig
- Risiko einer neuen Anforderung wird anhand der Abhängigkeit zu anderen Anforderungen bewertet und im Backlog verwaltet (Issue Tracker)
- Da die Anforderungen im Backlog nicht technisch sondern anwenderorientiert sind, hat es sich bewährt Produkteigenschaften als sog. User Stories zu betrachten.
- Wenn eine User Story abgeschlossen ist, dann ist eine Produkteigenschaft abgeschlossen

Scrum Artefakte - Product Backlog

- Die gewünschten Eigenschaften einer User Story lauten wir folgt:
Independent – unabhängig.
Negotiable – verhandelbar.
Valueable – nützlich.
Estimateable – schätzbar.
Small – klein.
Testable – überprüfbar.
= ideale Requirements!

Scrum Artefakte - Product Backlog Refinement

- Fortlaufender Prozess in dem das Backlog gewartet und weiterentwickelt wird
 - Ordnen der Einträge
 - Löschen von Einträgen, die nicht mehr wichtig sind
 - Hinzufügen von neuen Einträgen
 - Detaillieren von Einträgen
 - Zusammenfassen von Einträgen
 - Schätzen von Einträgen
 - Planung von Releases

Scrum Artefakte – Sprint Backlog

- Das Sprint Backlog ist der aktuelle Plan der für einen Sprint zu erledigenden Aufgaben und wird laufend aktualisiert

Scrum Artefakte – Product Increment

- = die Summe aller Backlog Einträge, die in einem Sprint abgearbeitet werden konnten