

Kent Beck: eXtreme programming explained. Embrace Change. Addison Wesley: 1999.

- „Agile Methoden“ ermöglichen einen **konstanten und nachhaltigen Entwicklungsprozess**, bei dem sich weder Entwickler noch Kunde verausgaben.
- Ständiges Augenmerk auf **technische Kompetenz** und **gutes Design** verbessern die Agilität.
- Die **Einfachheit der Lösungen** unter bestmöglicher Vermeidung von (nicht wirklich erforderlicher) Arbeit ist wesentlich.
- Die besten Entwürfe und Architekturen entstehen in **selbstorganisierenden Teams**.
- **Regelmäßige Reflexion im Team** ermöglicht eine kontinuierliche Verbesserung der Performance.

Da „Agile Methoden“ primär auf die praktische Umsetzung ausgerichtet sind, werden natürlich zahlreiche Beispiele für die mögliche Umsetzung der Grundsätze und Prinzipien im Entwicklungsprojekt gegeben. Viele der Prinzipien finden sich auch im Ansatz des **Extreme Programings (XP)**.

Beim Einsatz „Agiler Methoden“ im Software-Projekt müssen alle Prinzipien und Regeln umgesetzt werden – erst im Zusammenspiel ergeben sich jene Synergien, welche zu einem flexiblen, ballastfreien und für alle Beteiligten möglichst zufriedenstellenden Software-Entwicklungsprozess führen.

2 Scrum

Ein Begriff aus dem Rugby benennt diese team-orientierte, sehr bewegliche Methode für Softwareprojekte.



Scrum beschreibt einen Vorgehensrahmen, nach welchem Projekte jeder Art **agil**, also beweglich und unbürokratisch, **abgewickelt** werden können.



Der Begriff **Scrum** bezeichnet eine bestimmte Spielsituation im Rugby; frei übersetzt bedeutet er etwa „Gedränge“.

Im Folgenden wird die Anwendung von Scrum für Software-Entwicklungsprojekte dargestellt.

Die **Hauptmerkmale von Scrum** sind:

- lediglich drei Rollen (in der Grundform)
- Sammlung der Anforderungen im Product Backlog
- iterative Produktentwicklung in zeitlich klar definierten Zyklen
- ein autonom arbeitendes Team gleichberechtigter Teammitglieder

Die **zentralen Rollen bei Scrum**:

- **Product Owner**: Er vertritt den Auftraggeber (Kunden, Anwender) innerhalb des Teams. Seine Aufgabe ist die Erhebung, Beschreibung und Priorisierung der Anforderungen.
- **Scrum-Master**: Er gestaltet und moderiert den Entwicklungsprozess nach der Scrum-Methode und unterstützt Product Owner und Team. Als „Moderator“ ist der Scrum-Master nicht Mitglied des Teams.
- **Team**: Gruppe von Entwicklern, die für die Umsetzung der vom Product Owner beschriebenen Anforderungen sorgt. Das Team besitzt dabei eine große Gestaltungsfreiheit hinsichtlich seiner Arbeitsweise im Rahmen der Scrum-Abläufe.

Die Aufgaben eines **Projektmanagers** werden in Scrum von Product Owner und Team übernommen.

Product Backlog

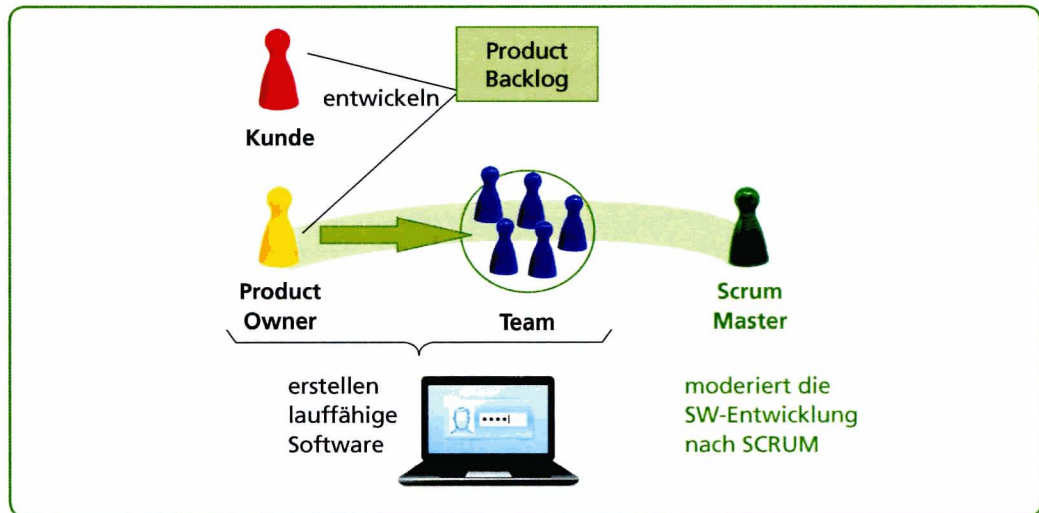


Der **Product Backlog** bildet die **Sammlung aller Anforderungen**.

Am Beginn der Anforderungsbestimmung erfolgt eine Zielbestimmung in Form einer „**Vision**“, in welcher der Product Owner gemeinsam mit dem Kunden festlegt, worin für diesen der Nutzen des zu erstellenden Programms liegt. Darauf aufbauend beschreibt der Product Owner mit Hilfe von User Stories (s. u.) die **Funktionen**, welche das System erfüllen soll. Auch Teammitglieder können Spezifikationen beitragen.

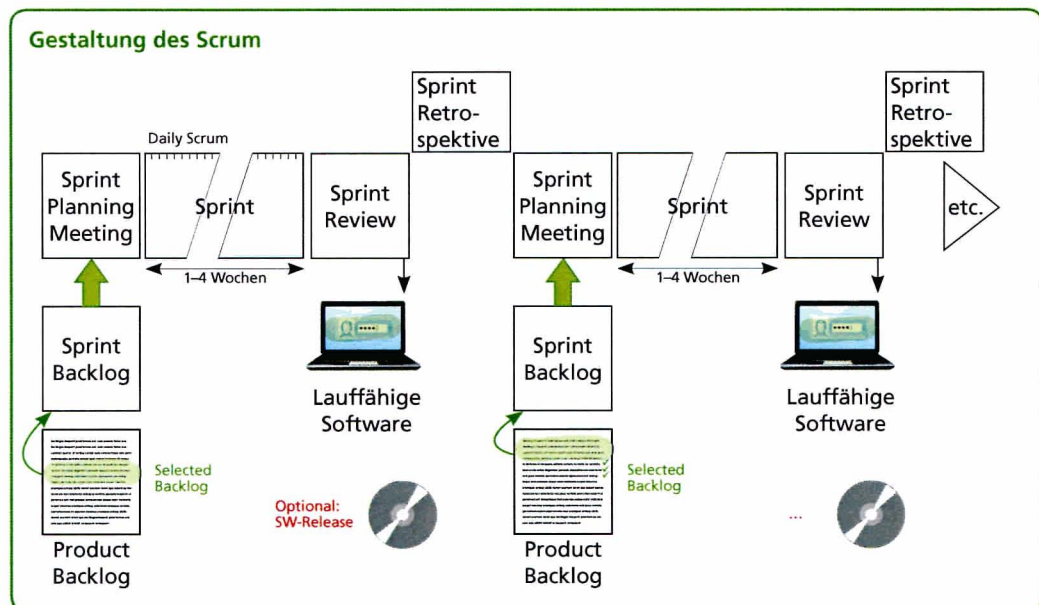
Anforderungen, welche für die Umsetzung im folgenden Schritt ausgewählt wurden (Selected Backlog) dürfen nicht mehr verändert werden.

Die folgende Abbildung zeigt das Zusammenspiel der Rollen:



Die Hauptelemente bei der Projektabwicklung nach Scrum sind:

- das Sprint Planning Meeting
- der Sprint
- der Daily Scrum
- das Sprint Review und
- die Sprint Retrospektive



1. Sprint Planning Meeting

Im Sprint Planning Meeting wird das Ziel des folgenden Sprints festgelegt. Der Product Owner wählt die umzusetzenden Anforderungen nach ihren Prioritäten aus und stellt sie dem Team vor. Das Team wählt jenen Umfang an Anforderungen, welche im kommenden Sprint realistisch umgesetzt werden können (Selected Backlog). Der Aufwand für die zu erstellenden Aufgaben wird im Team im „Planungspoker“ geschätzt. Anschließend werden die ausgewählten Anforderungen in einzelne Tasks (Aufgaben) zerlegt, welche den Sprint Backlog – die Aufgaben für den folgenden Sprint – bilden.

2. Sprint

Ein Sprint ist eine Entwicklungsphase, an dessen Ende immer verwendbare Software steht. „Verwendbar“ bedeutet, dass sie im vorgesehenen Funktionsumfang einsatzbereit wäre. Im Sinne einer iterativen Softwareentwicklung werden im Rahmen eines Projekts mehrere Sprints, meist

mit derselben Dauer (1–4 Wochen) durchgeführt. Ein Release-Sprint ist ein Sprint, an dessen Ende die verwendbare Software auch dem Kunden übergeben wird. Jeder Sprint wird durch Daily Scrums strukturiert.

3. Daily Scrum

Am Beginn jedes Arbeitstages trifft das Team zu einem kurzen Meeting, dem Daily Scrum, zusammen. Der Daily Scrum ist ein „Standup Meeting“, welches stehend in maximal 15 Minuten durchgeführt wird. Pünktliches Erscheinen ist natürlich absolute Pflicht jedes Teammitglieds. Jedes Teammitglied beantwortet dabei die drei Fragen:

- Was habe ich seit dem letzten Daily Scrum abgeschlossen?
- Woran werde ich bis zum nächsten Daily Scrum arbeiten?
- Gibt es etwas, das mich bei meiner Arbeit/am Vorankommen hindert?

Der aktuelle Projektfortschritt (s. u. Burndown-Chart) wird bewertet. Einfache Maßnahmen können während des Daily Scrum beschlossen werden („... ich helfe dir heute eine Stunde bei ...“), für umfangreichere Probleme muss eine gesonderte Sitzung vereinbart werden.

4. Sprint Review

Am Sprint Review nehmen Product Owner, Team, Scrum-Master, Vertreter des Kunden, Manager des eigenen Unternehmens, Mitglieder anderer Teams etc. teil. In dieser öffentlichen Form wird die im (bzw. bis zum) letzten Sprint fertiggestellte Software „live“ präsentiert, erklärt und diskutiert. Ziel des Sprint Review ist es, festzustellen, ob die Erwartungen des Kunden erfüllt wurden oder ob eine Anpassung für den kommenden Sprint erfolgen sollte.

5. Sprint Retrospektive

Zentrales Thema der Sprint Retrospektive ist der Arbeitsprozess selbst. In einer vom Scrum-Master moderierten Sitzung werden sowohl positive als auch negative Aspekte der Arbeit im vergangenen Sprint herausgearbeitet und analysiert. Ziel ist die kontinuierliche Verbesserung des Scrum-Prozesses und der Zusammenarbeit im Team. Das Ergebnis der Sprint-Retrospektive sind konkret und umsetzbar formulierte Maßnahmen, durch die die folgenden Sprints besser verlaufen sollen.

Fortschrittskontrolle in Scrum

Die Planung, Durchführung und Kontrolle der vereinbarten Arbeitsleistung wird in Scrum durch folgende Methoden unterstützt:

- User Stories
- Sprint Backlog
- Burndown Chart
- Timeboxing
- Definition of done

User Story







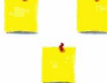





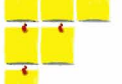
User Stories sind eine kunden- bzw. anwenderorientierte Form, Anforderungen an die zu erstellende Software zu beschreiben. Eine typische User Story (für eine Lagerverwaltung) könnte z. B. lauten:

„Als Lagerarbeiter möchte ich einen Artikel nach seiner Artikelnummer finden können.“

Die Verwendung von User Stories hilft bei der Kommunikation mit dem Kunden und bildet nach der Umsetzung einen Mehrwert für den Kunden. Sie macht auch den Projektfortschritt für den Kunden deutlich sichtbar. Die Formulierung der User Stories wird durch den Product Owner durchgeführt; für eine gleichmäßige Umsetzbarkeit der Stories bedarf es einiger Erfahrung.

Sprint Backlog

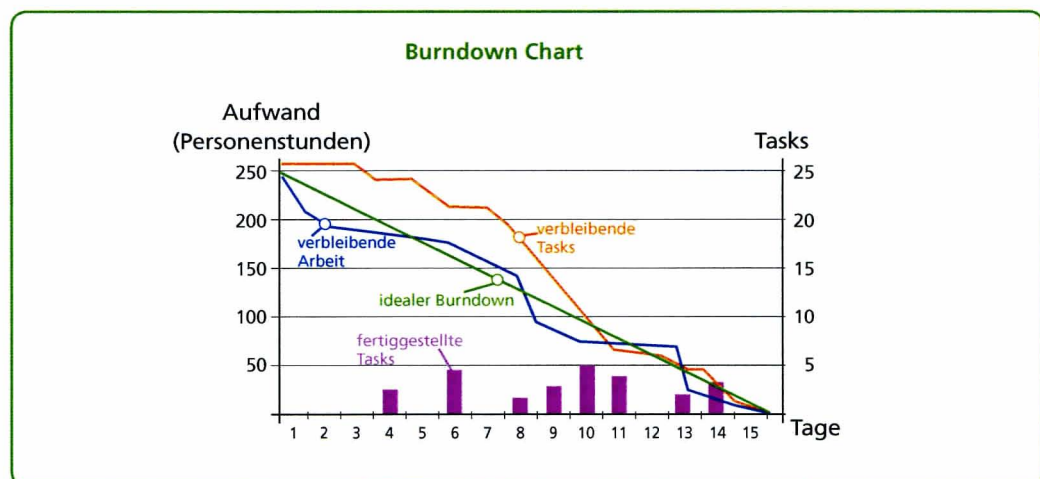
Im Rahmen des Sprint Planning Meeting werden die Anforderungen, z. B. User Stories, in einzelne Tasks (Aufgaben) für die Umsetzung im geplanten Sprint, zerlegt. Dieser Sprint Backlog wird meist mithilfe einer Pinwand visualisiert. Der Weg der einzelnen Tasks von „offen“ zu „fertig“ kann im Laufe des Sprints optisch mitverfolgt werden.

User Story	Offen	In Arbeit	Fertig
			
			
			
			

Burndown Chart

Der Burndown Chart ist eine Grafik, welche die Abarbeitung der für den Sprint vorgesehenen Arbeit bzw. Tasks darstellt. Im Gegensatz zur Earned Value Analyse wird nicht der Wertzuwachs dargestellt, sondern die Menge an noch abzuarbeitenden Tasks bzw. der laut Planung dafür zur Verfügung stehende Aufwand.

Die Entwicklungsgeschwindigkeit des Teams wird im Scrum als **Velocity** bezeichnet. Die Geschwindigkeit eines Teams wird in Story-Points je Sprint angegeben. (Story-Points sind eine relative Maßzahl für den Aufwand, den eine User-Story benötigt.)



Timeboxing

Der Einsatz von Timeboxing in einem Projekt bedeutet, dass Deadlines (Endtermine, vereinbarte Dauern) unbedingt einzuhalten sind. Der Daily Scrum darf z. B. maximal 15 Minuten dauern, danach wird abgebrochen; ein Sprint dauert z. B. 21 Tage, eine Verlängerung wird grundsätzlich nicht gemacht. Timeboxing hat den wichtigen Effekt, dass Termine immer ernst genommen werden, vereinbarte Termine verlässlich sind. Als Konsequenz daraus kann z. B. bei einem Sprint nicht die gewünschte Funktionalität ausgeliefert werden, auch fast fertige User Stories werden in das Product Backlog zurückgestellt. Damit in Zusammenhang steht auch die Definition, was als fertig gilt, die „Definition of Done“.

Definition of Done

Jedes Scrum-Team erstellt seine eigene „Definition of Done“. Diese enthält mehrere klar formulierte Kriterien, wann eine Anforderung oder User Story als fertig gelten darf. Dazu gehören z. B. verschiedene Tests, Code-Reviews durch Teammitglieder, Abnahme durch den Product Owner etc.