**Daily-Scrum Meeting (Scrum Meeting)**

Wie der Name andeutet, findet das Daily Scrum Meeting an jedem Arbeitstag statt. Es dauert nicht länger als 15 Minuten (Time-Box) und dient dem Team dazu, sich abzustimmen und gegenseitig zu informieren.

**Ablauf des Daily Scrum Meeting (kurz: Daily Scrum)**

* Kurzes, tägliches Status-Meeting des Teams
* Time-boxed (15 min)
* Stand-Up-Meeting, d.h. die aktiven Teilnehmer stehen (am besten im Kreis), sie sitzen nicht
* ScrumMaster nimmt teil, notiert sich genannte *Impediments* (s.u.) auf seiner *Blocks List* und greift moderierend(!) ein, wenn unbedingt nötig
* Product Owner nimmt nach Möglichkeit auch teil, um auf dem neuesten Stand zu bleiben und bei Bedarf Fragen zu beantworten
* Nur die Team Members sprechen und berichten einander jeweils Folgendes:  
  -  Was habe ich seit dem letzten Daily Scrum getan?  
  -  Was plane ich, bis zum nächsten Daily Scrum zu tun?  
  -  Was hat mich bei der Arbeit behindert *(Impediments)*?

**Hinweise**

* Ein fruchtbares Daily Scrum erzeugt durch seine Kürze bewusst weiteren Gesprächs- und Informationsbedarf, der in anschließenden Einzelgesprächen gedeckt werden kann - ein positiver Nebeneffekt der Time-Box.
* Es ist wichtig, dass die Team Members *einander* berichten und nicht ihren evtl. anwesenden Vorgesetzten oder dem ScrumMaster oder dem Product Owner. Sie schauen einander an, alle anderen Anwesenden sind nur Zuhörer. Das *Command-and-Control*-Muster zu durchbrechen, erfordert ein wenig Übung. Die Umstellung dauert eine Weile für neue Scrum-Teams. Innerhalb kurzer Zeit sind Daily Scrums jedoch hocheffiziente und zielgerichtete Veranstaltungen.
* Dass jedes Team Member weiß, woran die anderen arbeiten, ist wichtig, damit nicht jeder in seinem Elfenbeinturm bleibt, sondern ein Gesamtbild des Projekts erhält. Wer ein Problem hat, dem kann vielleicht ein anderer helfen.
* Dieser Hinweis gilt für alle Scrum-Meetings: Es wird pünktlich begonnen, nicht fünf Minuten später. Wer zu spät kommt, hat Pech gehabt und wird, falls das Team es so festgelegt hat, entsprechend einer team-internen Regelung sanktioniert, z.B. zwei Euro in eine Kasse zahlen, aus der am Ende des Projekts für einen wohltätigen Zweck gespendet wird. Das Geld hinterher als "Belohnung" fürs Zuspätkommen gemeinsam zu "verfressen", wäre kein gutes Signal.

**Sprint Backlog**

Eine Liste von → Tasks, welche den Arbeitsumfang des → Teams für den → Sprint festlegt. Die Liste präzisiert sich während des Sprints und wird täglich von allen → Team Members gepflegt, so daß sie immer den aktuellen Bearbeitungsstand reflektiert. Der Sprint Backlog ermöglicht es dem → *ScrumMaster*, jederzeit zu erkennen, wo das Team steht und ggf. steuernd einzugreifen, damit das → Sprint Goal nicht in Gefahr gerät.

**Product Backlog**

Priorisierte Liste von Anforderungen (→ Requirements) mit Schätzwerten (Estimates), welche den jeweiligen Funktionsumfang, ggf. auch ihre Komplexität, relativ zueinander widerspiegeln. Wichtig ist, dass es um eine "gefühlte Größe" geht, nicht um eine absolute Aufwandschätzung in Personentagen o.ä. Als Einheit der Schätzwerte wird oft der sog. → Story Point - nicht zu verwechseln mit einem Function Point - verwendet.

Je höher die Priorität einer Anforderung, desto feingranularer sind tendenziell die Schätzwerte, da das Product Backlog im Wesentlichen nach absteigender Priorität abgearbeitet wird. Das Product Backlog ändert sich im Lauf der Zeit, ist also keine endgültige, vorab festgelegte Spezifikation, sondern eine Anforderungsliste, die im Fluss ist und sich den Gegebenheiten des Projekts kontinuierlich anpasst : In einem → Sprint realisierte oder verworfene Requirements werden entfernt, Schätzwerte aufgrund Teilbearbeitung oder neuer Erkenntnisse aktualisiert, neue Anforderungen aufgenommen, Prioritäten verändert, grob geschätzte Themen oder Funktionsblöcke verfeinert, also in kleinere heruntergebrochen etc.

**Burndown Chart**

Graphik, die den Projektfortschritt eines Produktes, → Sprints oder → Releases in einer Kurve visualisiert. Die Kurve gibt für jeden Punkt auf dem horizontalen Zeitstrahl an, wieviel Arbeit nach jeweils aktueller Schätzung zu jedem Zeitpunkt noch übrig war/ist, um das Ziel zu erreichen. Mittels einer auf den vergangenen Schätzwerten basierenden Trendlinie durch die Kurve lässt sich das voraussichtliche zeitliche Ende des Projektes vorhersagen bzw. können sich andeutende Abweichungen vom Zeitplan innerhalb eines Sprints prognostiziert und Gegenmaßnahmen ergriffen werden.

**–––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––**

**Scrum-Rollen**

**Scrum-Rollen - Pigs & Chickens 🡪** In Scrum werden direkt am Prozeß beteiligte Personen Pigs (Schweine) genannt, Außenstehende heißen Chickens (Hühner). Das hat seinem Ursprung in einem englischen Witz.

**Scrum-Rollen - Product Owner 🡪**Immer, wenn Software entwickelt wird - egal ob im Kundenprojekt oder für ein Standardprodukt - gibt es einen Auftraggeber, der die fachliche Sicht vertritt, Anforderungen stellt und die spätere Umsetzung seiner Wünsche im Hinblick auf Funktionalität, Benutzbarkeit (Usability), Performanz und Qualität beurteilt.

**Scrum-Rollen – ScrumMaster 🡪** Agile Prozesse zeichnen sich - deshalb auch diese Bezeichnung - durch eine hohe Dynamik aus. Damit der Prozeß zielgerichtet verläuft und aus Dynamik nicht Chaos wird, gibt es den ScrumMaster. Er ist sozusagen die Seele des Prozesses und sorgt dafür, daß die Regeln eingehalten werden.

**Scrum-Rollen – Team 🡪** Das Scrum-Team, meistens kürzer einfach Team genannt, steht im Zentrum des Scrum-Prozesses, weil es für die Umsetzung der Anforderungen in Produktfunktionalität sorgt. Das Team insgesamt ist einer von drei Managern in Scrum.

**Windows Presentation Foundation** (kurz WPF), auch bekannt unter dem Codenamen Avalon, ist ein Grafik-Framework und Teil des .NET Frameworks von Microsoft, das mit Windows Vista, Windows 7 und Windows 8 ausgeliefert wird, sich aber auf Windows XP (bis zur Version 4.0) und Server 2003 nachinstallieren lässt.

Bei WPF handelt es sich „um eine 2006 gänzlich neu eingeführte Bibliothek von Klassen, die zur Gestaltung von Oberflächen und zur Integration von Multimedia-Komponenten und Animationen dient. Sie vereint die Vorteile von DirectX, Windows Forms, Adobe Flash, HTML und CSS“ [1]

WPF stellt ein umfangreiches Modell für den Programmierer bereit. Dabei werden die Präsentation und die Geschäftslogik getrennt, dies wird vor allem durch die Auszeichnungssprache XAML (basierend auf XML) unterstützt. XAML beschreibt Oberflächen-Hierarchien deklarativ als XML-Code. WPF-Anwendungen können sowohl Desktop- als auch Web-Anwendungen sein und benutzen, wenn möglich, auch Hardwarebeschleunigung. Das Framework versucht, die verschiedenen Bereiche, die für die Präsentation wichtig sind (Benutzerschnittstelle, Zeichnen und Grafiken, Audio und Video, Dokumente, Typographie), zu vereinen.

Zudem gibt es mit Microsoft Silverlight (ehemals WPF/E) eine Plattform, die explizit für das Web entwickelt wurde. Silverlight setzt dabei nicht auf das .NET-Framework auf, sondern benutzt eine reduzierte Version dessen. Silverlight kann auch mit XAML entwickelt werden.

**–––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––**

**MVC: Model View Controller**

Der englischsprachige Begriff model view controller (MVC, englisch für Modell-Präsentation-Steuerung) ist ein Muster zur Strukturierung von Software-Entwicklung in die drei Einheiten Datenmodell (engl. model), Präsentation (engl. view) und Programmsteuerung (engl. controller). Manche Autoren stufen es als Architekturmuster ein, andere als Entwurfsmuster.[1] Ziel des Musters ist ein flexibler Programmentwurf, der eine spätere Änderung oder Erweiterung erleichtert und eine Wiederverwendbarkeit der einzelnen Komponenten ermöglicht. Es ist dann zum Beispiel möglich, eine Anwendung zu schreiben, die dasselbe Modell nutzt und es dann für Windows, Mac, Linux oder für das Internet zugänglich macht. Die Umsetzungen nutzen dasselbe Modell, nur Controller und View müssen dabei jeweils neu implementiert werden.

**–––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––**

**Modell (model) 🡪** Das Modell enthält die darzustellenden Daten (und in manchen Umsetzungen des MVC-Musters auch die Geschäftslogik). Es ist von Präsentation und Steuerung unabhängig. Die Bekanntgabe von Änderungen an relevanten Daten im Modell geschieht nach dem Entwurfsmuster „Beobachter“. Das Modell ist das zu beobachtende Subjekt, auch Publisher, also „Veröffentlicher“, genannt.

**Präsentation (view) 🡪** Die Präsentationsschicht ist für die Darstellung der benötigten Daten aus dem Modell und die Entgegennahme von Benutzerinteraktionen zuständig. Sie kennt sowohl ihre Steuerung als auch das Modell, dessen Daten sie präsentiert, ist aber nicht für die Weiterverarbeitung der vom Benutzer übergebenen Daten zuständig. Im Regelfall wird die Präsentation über Änderungen von Daten im Modell mithilfe des Entwurfsmusters „Beobachter“ unterrichtet und kann daraufhin die aktualisierten Daten abrufen. Die Präsentation verwendet oft das Entwurfsmuster „Kompositum“.

**Steuerung (controller) 🡪** Die Steuerung verwaltet eine oder mehrere Präsentationen, nimmt von ihnen Benutzeraktionen entgegen, wertet diese aus und agiert entsprechend. Zu jeder Präsentation existiert eine eigene Steuerung. Die Steuerung sorgt dafür, dass Benutzeraktionen wirksam werden, z. B. durch Änderung der Präsentation (z. B. Verschieben des Fensters) oder durch Weiterleiten an das Modell (z. B. Übernahme von Eingabedaten oder Auslösen von Verarbeitungen). Als es noch keine Objektorientierung gab, bestand ein Modell nur aus Daten, und die Steuerung hat die Daten oft direkt aktualisiert. In einer objektorientierten Umgebung ist es dagegen besser, wenn das Modell die Geschäftsobjekte enthält und die Steuerung sich darauf beschränkt, Benutzereingaben (Daten und Methodenaufrufe) weiterzuleiten, von der Präsentation an das Modell. Die Steuerung enthält weiterhin Mechanismen, um die Benutzerinteraktionen der Präsentation einzuschränken. Die Steuerung kann in manchen Implementierungen ebenfalls zu einem „Beobachter“ des Modells werden, um bei Änderungen der Daten die Präsentation direkt zu manipulieren.