# Starten eines Scrum-Teams für Softwareentwicklung

## Anhand eines Beispiels wird erläutert wie Team-Coaching für Entwicklungsteams aussehen kann

18.09.2012, *Von Stephan Schwab*

Es gibt eine Vielzahl unterschiedlicher Definitionen und Angebote für Team-Coaching. Für viele ist der Begriff mit Maßnahmen zur Stärkung des Wir-Gefühls im Team verknüpft. Diese Maßnahmen, bei denen die Mitarbeiter außerhalb des Arbeitsplatzes gemeinsam eine Herausforderung meistern, machen Sinn. Doch helfen Sie nicht technische und organisatorische Fähigkeiten im Umfeld der eigenen Arbeit zu entwickeln. Sie bleiben auf der metaphorischen Ebene und somit quasi "obendrüber".

Das nachfolgend detailliert dargestellte Beispiel beschreibt wie das Starten eines Scrum-Teams über einen Zeitraum von zwölf Wochen aussehen könnte. Es geht um Softwareentwicklung und das Coaching findet direkt am Arbeitsplatz statt. Ein oder zwei Coaches - je nach Größe des Teams - unterstützen das Team beim Erlernen neuer technischer und organisatorischer Fähigkeiten.

Ziel des Coachings ist das Team in die Lage zu versetzen am Ende jeder Iteration, z.B. alle zwei Wochen, funktionsfähige Software in die Hände realer Anwender geben zu können.

### Erste Woche

Die erste Woche ist von der theoretischen Einführung in agile Entwicklung und Vorbereitungen für die Aufnahme der praktischen Arbeit geprägt. Neben einer Menge Erklärungen und praktischer Hilfe nimmt sich der Coach auch den Bedenken einzelner Mitarbeiter an, da für manche Mitarbeiter agiles Arbeiten im Team durchaus als bedrohlich empfunden werden kann. Wer z.B. bislang in einem Einzelbüro gearbeitet hat, kann Schwierigkeiten durch den scheinbaren Verlust der Privatsphäre in einem weitgehend offenen Teamraum haben. Auch ständige direkte Zusammenarbeit mit den Kollegen kann für manche sich zunächst als unangenehm darstellen. Nicht zuletzt wird auch auf die spezielle Situation von besonders extrovertierten oder introvertierten Personen eingegangen.

Am ersten Tag wenden wir uns den nachfolgend beschriebenen Themen und Aufgaben zu:

**Einführung in agile Entwicklung**
In einem Vortrag wird dem gesamten Team bestehend aus Analytikern, Testern, Programmierern und den für das Team zuständigen Führungskräften agile Entwicklung und der Ablauf der Coaching-Maßnahme vorgestellt.

**Umzug in den Teamraum**
Der Teamraum ist eine offene Schreibtischlandschaft mit viel beschreibbarer Wandfläche (whiteboards). Verbale und non-verbale Kommunikation wird durch räumliche Nähe ermöglicht. Der Coach hilft den Mitarbeitern sich an die neue Umgebung zu gewöhnen.

**Technische Einsatzbereitschaft**
Der Coach unterstützt das Team beim Einrichten aller technischen Hilfsmittel in einer Art und Weise, die beständigen Wechsel des Sitzplatzes der Mitarbeiter ermöglicht.

**Wählen von Teamname, Verständigung über Ziele, Festlegung von Teamnormen**
Kein Team ohne Name. Gemeinsam wählen die Mitglieder etwas, das ihnen eine gemeinsame Identität gibt. Im Gruppengespräch werden sich alle über die Ziele für das Team klar - natürlich in Einklang mit dem an das Team erteilten Auftrag. Und schließlich gehören auch ein paar Verhaltensregeln zu den Dingen, die abgesprochen werden wollen. Das ist z.B. die Frage wie mit privaten Telefongesprächen oder Gleitzeitregelungen umgegangen werden soll.

**Bildung des Teams**
Durch Einzelgespräche wird allen für das Team vorgesehenen Mitarbeitern Gelegenheit zum Stellen von Fragen und Ausräumen von eventuellen Bedenken gegeben. Es kann dabei vorkommen, daß Mitarbeiter sich als nicht geeignet herausstellen oder von sich aus nicht zum Team gehören möchten.

## Fit für die Zukunft

Arbeitsgruppen aus Spezialisten bringen unterdurchschnittliche Leistung, weil die darin versammelten Spezialisten nicht wirklich zielgerichtet zusammenarbeiten können. Die Grundlage für Erfolg ist immer technisches Können. Die Grundlage für dauerhaften Erfolg ist organisatorisches Können.

Gern unterstütze ich Sie bei der Entwicklung organisatorischer und technischer Fähigkeiten. Beispiele für dazu hilfreiche Maßnahmen sind:

**Lernendes Team**
Die Fähigkeit schnell zu lernen erlaubt Unternehmen jeder Art sich auf **schnelle Veränderungen im Markt** gut anzupassen und dadurch fortwährend die Erwartungen von Kunden und Mitarbeitern zu erfüllen. **Begeisterte Kunden beweisen den Erfolg!**

**Lean Startup**
**Beobachtung**, **Schlußfolgerung** und **Anpassung** sind Schlüsseltechnologien. Lernen Sie durch kontrollierte **Experimente**, sorgsames Messen und Schlußfolgern Erkenntnisse für die weitere Unternehmensentwicklung zu gewinnen.

**Agile Vorgehensweisen**
Das Ziel agiler Vorgehensweisen war **niemals** das Ziel mehr Software schneller zu entwickeln (**Effizienzsteigerung**). Stattdessen geht es darum **zielgerichteter** die **Arbeit** im Unternehmen zu **organisieren** - ganz egal, ob es dabei um Softwareentwicklung oder **andere Tätigkeiten** geht.

**Defekte durch ATDD vermeiden**
Auftraggeber und Entwicklungsteam definieren gemeinsam in Form einer **ausführbaren Spezifikation** was die zu entwickelnde Software tun soll.

**Activity-Centered Design**
Produkte mit gutem Design sind aus einem **tiefen** und **umfassenden Verständnis** der **Tätigkeiten** des Anwenders hervorgegangen. Das macht sie **fit** für einen anspruchsvollen Markt und **besser** als der Mitbewerb. In einigen Fällen kann dieser **Unterschied** zu **gewöhnlichen** Produkten einen ganzen Industriezweig auf den Kopf stellen.

Der zweite Tag ist ganz für einen Workshop mit praktischen Übungen reserviert.

**Workshop und Simulation**
Behandelt werden die Themen *release planning*, *backlog*, Verantwortlichkeiten der einzelnen Rollen, Iterationen, *standup*, Vorführung fertiger Software und anderer Ergebnisse, Retrospektive.

Am dritten Tag steigen wir dann in die praktische Arbeit ein.

**Koordinierung der Zusammenarbeit des gesamten Teams**
Wir üben uns mit Hilfe eines kurzen Treffens im Stehen untereinander zu koordinieren. Dabei geht es primär um die Frage wer macht heute was und wo klemmte es gestern. Da dies das erste Mal ist, ist dies zunächst nur eine Übung und Einführung.
**Einführung in *Acceptance Test Driven Development***
Eine kurze Präsentation mit anschließendem Gruppengespräch führt in ATDD ein. Daraus entwickelt sich dann eine praktische Übung zum Erstellen von *user stories* und dem zielgerichteten Entdecken von Akzeptanzkriterien.
**Aufgaben entdecken**
Basierend auf der Einführung in ATDD beginnt das Team unter Anleitung *user stories* zu schreiben und Akzeptanzkriterien zu entdecken.
**Arbeit nach Wichtigkeit mittels einer Anforderungskarte ordnen**
Das Team lernt eine Anforderungskarte zur übersichtlichen Darstellung der Anforderungen für das Projekt zu nutzen. Mit Hilfe der Karte kann die sinnvollste Reihenfolge der Ausführung ermittelt werden.

Der vierte Tag beginnt mit der Koordinierung der Aufgaben des Tages im Stehen und steht ansonsten ganz im Zeichen der Arbeit mit *user stories*.

**Entdecken der Komplexität der Anforderungen und wie man mit dem Unbekannten umgeht**
Das Team lernt wie man die Größe von *user stories* einschätzt. Es geht uns um die Größe und nicht um die Zeit für die Implementierung.
**Planen**
Zum Planen der Arbeit über mehrere Iterationen hinweg erstellen die Teammitglieder eine Anforderungskarte basierend auf Wichtigkeit und geschätzter Größe der *user stories*.
**Testgetriebene Entwicklung**
Programmierer und Tester erhalten eine Einführung in testgetriebene Entwicklung und paarweise Zusammenarbeit.
**Exemplarisches Bearbeiten der ersten *user stories* mittels 3-Amigo Technik**
Das gesamte Team bearbeitet unter Anleitung gemeinsam die ersten *user stories*. So erfahren alle wie das vorher theoretisch gelernte tatsächlich in der Praxis umgesetzt wird. Es wird dabei in die 3-Amigo Technik eingeführt. Analytiker, Tester und Programmierer lösen die durch die *user story* repräsentierte Aufgabe gemeinsam.
**Auslieferungsplan (*release plan*)**
Basierend auf der Anforderungskarte wird für das Team und den Auftraggeber ein dynamischer Auslieferungsplan erstellt. Das Team lernt wie man diesen jeweils aktuell hält.

Wie die Tage zuvor beginnt auch der fünfte Tag mit einem kurzen Koordinierungsgespräch im Stehen. Während die Teammitglieder als 3-Amigos weiterhin an den ersten *user stories* arbeiten, erhält nun die Rolle des *Product Owner* besondere Aufmerksamkeit.

**Aufgaben und Verantwortlichkeiten der Rolle *Product Owner***
Im Einzelgespräch am Rande des Teamraumes wird der für die Rolle des *Product Owner* vorgesehene Mitarbeiter auf seine Rolle vorbereitet. Er erhält mehr Informationen über die Rolle und hat Gelegenheit Antworten auf seine Fragen zu erhalten. Aus didaktischen Gründen erfolgt dies am Ende der ersten Woche, weil so die Gelegenheit besteht durch Mitwirken und Beobachten der technischen Mitglieder des Teams leichter zu verstehen wie die Zusammenarbeit gestaltet sein soll.
**Vorführung funktionsfähiger Software (*show & tell*)**
Am Ende des Tages führt das gesamte Team seinem Auftraggeber funktionsfähige Software vor. Selbstverständlich wird die Software nach nur zwei Tagen aktiver Entwicklung noch nicht viele Funktionen enthalten. Aber was da ist, ist getestet und funktioniert!

## Zweite Woche

Die zweite Woche steht ganz im Zeichen des Entdeckens neuer Anforderungen und zielgerichteter Entwicklung ohne Rückschritte. Wie es schon in der ersten Woche üblich war, beginnt jeder Tag mit einem kurzen Koordinierungspräch im Stehen. Die Woche endet mit der Vorführung einer neuen Version der Software (getestet und funktionsfähig) und dem Sammeln von Rückmeldungen des Auftraggebers.

**Entdecken neuer Anforderungen, Anforderungskarte, Auslieferungsplan**

Das gesamte Team beteiligt sich am Entdecken neuer Anforderungen und Schreiben von *user stories*, die nach Größenschätzung und Erkennen der Wichtigkeit der Anforderungskarte und dem Auslieferungsplan hinzugefügt werden.

***Walking Skeleton***

Um das Team in die Lage zu versetzen flexibel auf Veränderungen in den Anforderungen reagieren zu können und vom Auftraggeber sinnvolle Rückmeldung zu erhalten, führen wir die Technik *walking skeleton* ein.

**Fortwährende Integration aller Komponenten (*continuous integration*)**

Zur Festigung und Sicherung der technischen Fähigkeit fortwährend funktionierende Software am Ende jeder Iteration abliefern zu können, führen wir *continuous integration* ein.

**Fortwährende Ausführung aller Tests**

Wir führen Techniken zur ständigen Ausführung automatisierter Tests ein, damit was einmal funktioniert hat auch in der Zukunft funktionsfähig bleibt. Damit schaffen wir uns auch ein Frühwarnsystem, welches uns alamiert, wenn wir existierende Funktionalität gefährden.

**Definition von *Fertig***

*Fertig* kann schnell ein dehnbarer Begriff werden, wenn wir im Team nicht eine klare Definition haben. Unter Anleitung des Coaches legt das Team fest was *fertig* bedeutet.

**Fortwährende Verbesserung durch Retrospektive**

Am Ende der zweiten Woche führen wir unsere erste Retrospektive durch. Wir wollen gemeinsam sehen was gut funktioniert hat, was nicht so richtig klappte und wie wir uns verbessern können.

## Dritte Woche

In der dritten Woche vertiefen wir unsere Kenntnisse bzgl. der technischen und organisatorischen Fähigkeiten, die wir in den beiden vorangehenden Wochen kennengelernt haben. Das Team beginnt einen Rhythmus einzuhalten. Zusätzlich führen wir die folgende Technik ein, wenn es vom Kenntnisstand und Erfahrungsgrad mit den gegebenen Teammitgliedern bereits möglich ist.

***Emerging Design***

Eine anspruchsvolle Technik für erfahrene Entwickler ist *emerging design*. Damit können wir unser Softwaredesign flexibel halten und schnell auf neue Anforderungen reagieren ohne in technische Sackgassen zu geraten.

## Vierte Woche

In Abhängigkeit der Erfahrung und Leistungsfähigkeit der Tester und Programmierer im Team führen wir weitere fortgeschrittene Techniken der Softwareentwicklung ein. Darüberhinaus wird der bisherige Rhythmus aus den vorigen Wochen beibehalten.

**Fortwährende Auslieferung (*continuous deployment*)**

Wäre es nicht wunderbar jederzeit innerhalb kürzester Zeit (Stunden) neue Fähigkeiten der Software hinzufügen und an die Anwender ausliefern zu können? Sofern die Programmierer und Tester im Team die notwendigen Grundlagen mitbringen, beginnen wir diese technische Fähigkeit im Team zu entwickeln.

**Fortwährende Einbindung des Auftraggebers**

Durch die technische Fähigkeit zur fortwährenden Auslieferung wird das Team in die Lage versetzt neue Versionen der Software täglich in einer Testumgebung für den Auftraggeber zur Verfügung zu stellen. Dort kann dieser beständig Einsicht nehmen und Rückmeldung an das Team geben.

***Exploratory Testing***

*Exploratory Testing* ist eine Technik für fortgeschrittene Tester. Es werden ungewöhnliche Kombinationen aus Bedienung und Eingaben gesucht, um zu sehen wie die Software in diesen Fällen reagiert.

## Fünfte bis zehnte Woche

Zu Beginn der fünften Woche hat das Team bereits zwei Iterationen - bei einer Länge von zwei Wochen - hinter sich. Die erste Iteration war hauptsächlich durch Lernen geprägt, aber die zwei Iteration enthielt bereits eine Menge "echter" Arbeit. Daher können wir nun das Konzept *gestriges Wetter* zur Planung einführen. Wir ermitteln wieviel bisher geleistet wurde und nutzen diese Information für eine Prognose.

In den folgenden Wochen begleitet der Coach das Team zur Vertiefung der neuen Techniken. Er steht bei Fragen und Schwierigkeiten als Ratgeber, Gesprächspartner und bei Bedarf auch als Trainer zur Verfügung.

Die praktische Hilfe erfolgt in der Regel durch paarweises Arbeiten. Teammitglied und Coach arbeiten gemeinsam an einer Aufgabe. Dabei gibt der Coach nicht die Richtung vor, sondern

hilft dem Mitarbeiter, der ja eine Fachkraft ist, durch Fragen oder auch mal Vorschläge selbst die richtige Vorgehensweise oder Lösung zu finden. Sofern erforderlich kann der Coach aber auch kurzfristig in die Rolle eines Trainers wechseln und dem Mitarbeiter zeigen wie er eine Aufgabe lösen würde. Im Falle eines Programmierers wird dann z.B. *pair programming* benutzt.

## Loslassen

Ab der zehnten Woche oder auch schon früher beginnt der Coach loszulassen und das Team darauf vorzubereiten ohne ihn auszukommen. Es mag sinnvoll erscheinen - quasi als Test - für eine Woche nicht beim Team zu sein, um zu sehen wie weit sie sind. Oder der Coach setzt nach der zehnten Woche für eine Iteration aus und kommt für die anschließende Retrospektive wieder.

Generell wird mittels Retrospektiven festgestellt, ob das Team im Sinne des Dreyfus-Modelles zur Wissensaneignung erfolgreich eine höhere Stufe erreicht hat. Wenn das noch nicht der Fall sein sollte, empfiehlt es sich über spezielle Trainingsmaßnahmen nachzudenken. Der Coach kann hier Empfehlungen abgeben.

**Hinweis:** Ich habe in diesem Artikel hier und da einige englischsprachige Begriffe verwendet. Diese sind in der einschlägigen Literatur gang und gäbe. Wer aber nicht aus dem technischen Bereich kommt, mag diese nicht kennen. Wenn möglich, habe ich auf eigene Hintergrundartikel zur Erklärung verwiesen. Für die Fälle, in denen solch ein Artikel noch nicht verfügbar war, möchte ich auf die Möglichkeit der Web-Recherche via Google, etc. verweisen.

**DIES IST EIN BEISPIEL** und als solches auf keinen Fall als Fahrplan für konkretes Team-Coaching zu sehen. Welche Vorgehensweise richtig ist, hängt von den existierenden Fähigkeiten des Teams, seinem Umfeld und der Art der zu entwickelten Software ab. Vor Beginn des Team-Coachings sollte immer mittels Retrospektive und SWOT-Analyse der Ist-Zustand geklärt werden.

**"Aber das ist nicht Scrum"** mag mancher nach dem Lesen dieses Artikels ausrufen. Scrum ist ein sehr einfaches Rahmenmodell, welches durch gewisse Zeremonien dem Team einen Rhythmus geben möchte. In gewisser Weise ist Scrum mit Stützrädern zum Erlernen des Fahrradfahrens vergleichbar. Im Grunde genommen will man möglichst schnell lernen nicht mehr Scrum machen zu müssen. Und da das Einführen eines bestimmten organisatorischen Prozesses ohne Erweiterung der technischen Fähigkeiten kaum nachhaltige Wirkung für das Software-Team haben kann (alter Wein in neuen Schläuchen), kommt in meiner Form des Team-Coachings der technische Bereich nicht zu kurz. Selbstverständlich gehört die "korrekte" Anwendung von Scrum mit dazu, steht aber nicht im Vordergrund.





Caimito Technologies
Brandenburger Str. 11
25980 Sylt
Deutschland

Telefon +49 151 61623277
email info@caimito.net
Impressum