Eine Beispiel-Checkliste für Scrum Master

Von Michael James, übersetzt von Urs Reupke und Peter Rössler (@p_roessler)
(mj4scrum@gmail.com, ur@it-agile.de, pr@it-agile.de)

Fassung 1.2.2 vom 16.03.2015, auf Basis der englischen Version 1.2 vom 24. Juli 2012

Ist der Scrum Master ein Vollzeit-Moderator?

Ein *anständiger* Scrum Master kann zwei bis drei Teams gleichzeitig betreuen, wenn man die Rolle darauf beschränkt Meetings zu organisieren, Fristen durchzusetzen und sich um die Hindernisse zu kümmern, die jemand ausdrücklich anspricht. Die Teams schaffen vermutlich immer noch mehr als das, was die Organisation vor Scrum von ihnen erwartet hat, und eine Katastrophe wird es auch nicht geben.

Wollen Sie aber ein Team, das mit Begeisterung Dinge erreicht, an die es vorher nicht geglaubt hätte, dann sollten Sie in Betracht ziehen, ein *außerordentlicher* Scrum-Master zu werden:
Ein außerordentlicher Scrum-Master kann genau *ein* Team betreuen. Wir empfehlen einen fest zugeordneten Scrum-Master für ein Scrum-Team mit etwa 7 Mitgliedern, insbesondere wenn sie Scrum zum ersten Mal begegnen.
Organisationen, die sich ständig verändern, brauchen außerordentliche Scrum-Master ganz besonders.

Wenn Ihnen noch nicht bewusst ist, welche Aufgaben es für einen Scrum Master zu tun gibt, schauen Sie doch einmal auf Ihren Product Owner, das Team und die Entwicklungsmethoden, die das Team verwendet. Achten Sie auch auf die Organisation außerhalb des Teams. Da es kein Patentrezept für einen außerordentlichen Scrum Master gibt, habe ich im Folgenden typische Dinge zusammengefasst, die Scrum-Master immer wieder übersehen.

Bitte markieren Sie die Punkte mit √, ∆, ? oder nz. Die Erklärungen dafür stehen auf der letzten Seite.

Teil 1 – Was macht Ihr Product Owner?

Scrum Master verbessern die Effektivität des Product Owners, indem sie ihm helfen, eine geeignete Vorgehensweise zu finden, das Product Backlog und den Release-Plan zu pflegen. Wichtig ist, dass Sie ihm die Priorisierung nicht abnehmen – das darf nur der PO!

- ❏ Entspricht die Priorisierung des Product Backlogs der aktuellen Ansicht und dem Wissen des POs? Sind die Anforderungen und Wünsche aller Stakeholder im Product Backlog eingepflegt?
  Denken Sie daran, dass sich das Backlog ständig entwickelt[1].

[1] Auf Englisch heißt diese Eigenschaft *emergent*.

- ❏ Ist die Größe des Product Backlog überschaubar?
  Die Menge der Einträge bleibt überschaubar, wenn Sie darauf achten, dass Einträge nach oben hin feiner werden, und weiter unten allgemein gefasste Epen stehen. Es ist unproduktiv, die Dinge weiter unten zu genau zu analyiseren, denn Anforderungen ändern sich jedesmal, wenn die Stakeholder/Kunden das wachsende Produkt sehen und darüber sprechen.
- ❏ Könnte der PO eine Anforderung besser als eine unabhängige, verhandelbare, wertvolle, schätzbare, kleine und testbare User-Story beschreiben? Schauen Sie insbesondere die Anforderungen an, die weit oben im Product Backlog stehen. (INVEST-Kriterien)[2]
- ❏ Weiß Ihr Product Owner um *technische Schulden* und haben Sie darüber gesprochen, wie man diese vermeidet? Ein Puzzlestück könnte hier sein, die *Definition of Done* um automatisierte Tests und *Refactoring* zu erweitern.
- ❏ Ist das Backlog ein *information radiator*? Können alle Stakeholder es leicht einsehen?
- ❏ Falls Sie ein elektronisches Werkzeug benutzen, um das Backlog zu verwalten: Kann es jeder leicht benutzen? Elektronische Werkzeuge werden leicht zu *information refrigerators*, die nur der Scrum Master benutzen kann.
- ❏ Können Sie helfen, Informationen zu verbreiten, indem Sie Ausdrucke zeigen?
- ❏ Können Sie helfen, Informationen zu verbreiten, indem Sie große und gut sichtbare Diagramme bereitstellen?
- ❏ Haben Sie Ihrem PO geholfen, die Einträge im Backlog in geeignete Releases zu sortieren? Sind die Einträge nach Wichtigkeit sortiert?
- ❏ Wissen alle, ob der Release-Plan noch der Wirklichkeit entspricht? Im Sprint Review könnten Sie versuchen, Produkt- oder Release-Burndown-Diagramme[3] zu zeigen, nachdem alle sich darauf geeinigt haben, welche Backlogeinträge fertig sind. Diese Diagramme zeigen die Menge der erledigten und neuen Einträge und helfen so, Veränderungen im Umfang oder Zeitplan früh zu erkennen.
- ❏ Hat Ihr PO den Release-Plan nach dem letzten Sprint-Review angepasst? Nur wenige POs, die hinreichend getestete Produkte termingemäß herausbringen, planen ihre Releases nach jedem Sprint neu. Vermutlich stellen sie dafür aber jedesmal einen Teil der Arbeit zurück, weil sie Aufgaben entdeckt haben, die dringender sind.

Teil II – Wie geht es Ihrem Team?

Einerseits ist es eine gute Idee, mit gutem Beispiel voran zu gehen und mit einzelnen Teammitgliedern an deren Aufgaben zu arbeiten. Andererseits riskieren Sie dadurch, dass Sie sich mit technischer Arbeit ablenken.

Ihre wichtigsten Pflichten gegenüber dem Team sind die folgenden:

- ❏ Arbeitet Ihr Team im Fluß?
  Sie erkennen diesen Zustand an verschiedenen Merkmalen:[4]

---

[2] http://xp123.com/articles/invest-in-good-stories-and-smart-tasks/
[3] Mike Cohn, Agile Estimation and Planning. (2005).
[4] Mihaly Csikszentmihalyi, Flow: The Psychology of Optimal Experience (1990).

    - ❏ Klare Ziele: Erkennbare Erwartungen und Regeln sowie erreichbare Ziele, die zu den persönlichen Fähigkeiten passen
    - ❏ Konzentration und Fokus auf einen beschränkten Arbeitsbereich, der volle Aufmerksamkeit bekommt
    - ❏ Verlust von Befangenheit: Das Team handelt, ohne darüber nachzudenken.
    - ❏ Direktes und schnelles Feedback: Erfolge und Fehlschläge bei der Zusammenarbeit liegen auf der Hand, und das Team passt sein Verhalten daran an.
    - ❏ Fähigkeiten und Herausforderung halten sich die Waage: die Aufgaben sind weder zu einfach noch zu schwer.
    - ❏ Das Gefühl, die Situation oder Tätigkeit selbst zu kontrollieren
    - ❏ Die Tätigkeit an sich ist belohnend, deswegen geht sie leicht von der Hand
- ❏ Kommen die Teammitglieder gut miteinander aus, blödeln sie herum und feiern sie gegenseitige Erfolge?
- ❏ Messen sich die Teammitglieder gegenseitig an ihren hohen Standards, und fordern sie sich heraus, daran zu wachsen?
- ❏ Gibt es Angelegenheiten oder Themen, über die Teammitglieder nicht sprechen, weil es ihnen unangenehm ist?[5]
- ❏ Haben Sie unterschiedliche Formate und Orte für die Retrospektive ausprobiert?[6]
- ❏ Hat das Team das Sprintziel im Blick? Vielleicht könnten Sie in der Mitte des Sprints mit dem Team überprüfen, ob alle die Akzeptanzkriterien der Product-Backlog-Einträge im aktuellen Sprint verstanden haben.
- ❏ Ist das Taskboard des Teams aktuell? Zeigt es, was das Team gerade macht? Achten Sie dabei auch darauf, ob es geheime Tätigkeiten gibt oder Teilaufgaben, die länger als einen Tag brauchen. Aufgaben, die sich nicht mit den Absprachen für den Sprint decken, sind in Hinblick auf diese Absprachen Hindernisse (*Impediments*).
- ❏ Besteht das Team aus 3-9 Leuten und hat es alle Fähigkeiten, die es benötigt, um auslieferbare Produktinkremente zu bauen?
- ❏ Sind die Dinge, die dem Team helfen sich selbst zu organisieren, für das Team sichtbar und einfach zu verwenden? Denken Sie dabei z.B. an das Taskboard, das Burndown/-up und die Liste der Hindernisse.
- ❏ Sind diese Dinge ausreichend vor externen Störenfrieden geschützt? Übertriebenes Interesse an der täglichen Arbeit kann die Transparenz des Teams zerstören und behindert die Selbstorganisation.
- ❏ Melden sich Teammitglieder freiwillig für Aufgaben?
- ❏ Zeigen die Backlogeinträge klar, bei welchen technische Schulden zurückgezahlt werden müssen? Wird der Code dadurch schrittweise angenehmer zu bearbeiten?
- ❏ Vergessen die Teammitglieder ihre Arbeitstitel und fühlen sie sich gemeinsam verantwortlich für alle Gesichtspunkte der Arbeit? Denken Sie insbesondere an Testen und Dokumentation.

---

[5] Kerry Patterson, Crucial Conversations: Tools for Talking When Stakes are High (2002). Also consider enlisting a professional facilitator who can make uncomfortable conversations more comfortable.

[6] Derby/Larson Agile Retrospectives: Making Good Teams Great (2006).

Teil III – Halten Sie sich an die Regeln der Technik?

- ❏ Hat Ihr System einen “Tests ausführen”-Knopf? Kann damit jeder - auch Teamfremde - auf einfache Weise feststellen, ob ein Fehler vorliegt und ob etwas kaputt gegangen ist, das schon einmal funktioniert hat?
  Viele Teams bauen solche Tests mit xUnit-Bibliotheken.
- ❏ Haben Sie ein gutes Verhältnis zwischen automatischen Systemtests (“funktionalen Tests”) und automatischen Unit-Tests?
- ❏ Schreibt das Team beide Arten von Tests in der gleichen Sprache wie das System selbst? Spezialsprachen behindern die Zusammenarbeit, Gleiches gilt für Aufzeichnungswerkzeuge, die nur wenige Teammitglieder verstehen.
- ❏ Kennt das Team den hilfreichen Graubereich zwischen Systemtests und Unit-Tests?[7]
- ❏ Schlägt der CI-Server Alarm, wenn ein Test bricht? Kann das Team diese Feedbackschleife auf Stunden oder sogar Minuten verkürzen? (“Daily builds are for wimps.” – Kent Beck)
- ❏ Berücksichtigt der CI-Server[8] wirklich *alle* Tests?
- ❏ Haben die Teammitglieder entdeckt, wieviel Spaß es macht die Architektur ständig zu verbessern und den Code immer wieder zu refaktorieren[9], anstatt sie von vornherein festzuschreiben (“Big Design Up Front”)? Refaktorieren ist dabei klar definiert: Man verändert die innere Struktur, aber nicht das wahrgenommene Verhalten. Entwickler sollten mehrmals stündlich refaktorieren. Gibt es Duplikation, komplexe Logik, lange Methoden, viele Einrückungsebenen, schlecht benannte Symbole, übermäßige Abhängigkeit? All das sind Hinweise darauf, dass der Code refaktoriert werden möchte. Vertrauen in die diese Änderungen können nur automatische Tests geben. Refaktoriert das Team nicht, wird es schwerer, das Produkt zukünftig zu ändern – auch, weil sie keine guten Entwickler finden werden, die sich damit herumschlagen wollen.
- ❏ Enthält Ihre “Definition of Done” automatische Tests und Refactoring? Testgetriebene Entwicklung (TDD) hilft dem Team, diese Ziele auch zu erreichen.
- ❏ Programmieren die Teammitglieder vornehmlich in Paaren? Programmieren in Paaren (*Pair Programming*) kann die Wartbarkeit des Codes drastisch erhöhen und senkt die Fehlerrate. Es bringt die Leute aus der Komfort-Zone und wirkt manchmal langsamer, besonders, wenn wir Codezeilen pro Kopf messen, nicht den geschaffenen Wert. Gehen Sie mit gutem Beispiel voran und arbeiten Sie einige Tage im Paar mit den Teammitgliedern. Einige von ihnen werden von da an das Arbeiten in Paaren bevorzugen.

Teil IV – Wie geht es der Organisation?

- ❏ Reden die Teams miteinander? Ein “Scrum of Scrums” ist nur ein Mittel, um das zu erreichen, und nicht unbedingt das Beste.

---

[7] http://blogs.collab.net/agile/michaeljames/junit_is_not_just_for_unit_testing_anymore/
[8] http://www.martinfowler.com/articles/continuousIntegration.html
[9] Martin Fowler, Refactoring: Improving the Design of Existing Code (1999).

- ❏ Können die Teams unabhängig voneinander lauffähige Features produzieren? Können sie dabei die Grenzen der Architektur ausreizen?[10]
- ❏ Treffen sich die Scrum-Master regelmäßig und bearbeiten Hindernisse innerhalb der Organisation?
- ❏ Hängen die Organisationshindernisse an der Wand neben dem Platz des Entwicklungsleiters? Können Sie die Kosten dieser Hindernisse in Euros messen, in Zeit, in Qualität oder verpassten Chancen, neue Kunden zu gewinnen? (Denken Sie dabei an Ken Schwabers Lektion: “Ein toter Scrum-Master ist ein nutzloser Scrum-Master.”[11] – Sie helfen Ihrem Team nicht mehr, wenn man Sie entlassen hat.)
- ❏ Bietet Ihre Organisationen Karrieremöglichkeiten, die sich mit den Zielen der Teams decken? Die wenigsten tun das. Die Antwort lautet auch dann “Nein”, wenn es der Karriere dient, mehr zu programmieren oder Entwurfsarbeit zu machen, anstatt zu testen, zu automatisieren oder zu dokumentieren.[12]
- ❏ Schreibt die Fachpresse oder sogar die Tageszeitung begeistert über Ihre Art zu arbeiten, schreibt, dass Sie Marktführer sind?
- ❏ Schaffen Sie eine lernende Organisation?

Zum Schluss

Wenn Sie die meisten Punkte als erledigt abgehakt haben und tagsüber trotzdem noch Zeit haben, würden wir gerne von Ihnen hören: Bitte schreiben Sie uns!

Es gibt kein Patentrezept, um Einfallsreichtum zu fördern. Wir führen hier lediglich einige Punkte auf, die Ihnen helfen können.

Manchmal werden Sie Angst bekommen, wenn Sie etwas erkennen, dass Sie verändern sollten.
Das ist ein Zeichen dafür, dass Sie auf dem richtigen Weg sind.

[10] http://featureteamprimer.org/
[11] Ken Schwaber, Agile Project Management with Scrum (2004)
[12] Alfie Kohn, Punished By Rewards: The Trouble with Gold Stars, Incentive Plans, A's, Praise, and Other Bribes (1999)

Datenblatt: Organisationshindernis

Symptom:

Ursache (“Fünf-Mal-Warum-Methode”):

Einfluss auf das Geschäft:

Einfluss auf die Moral:

Klar formulierter Handlungsvorschlag:

Anleitung

Falls Sie diese Liste als Übungsaufgabe bekommen haben und Sie bereits versuchen, Scrum zu verwenden, beschreiben Sie bitte, was Sie in Ihrer Organisation sehen.
Markieren Sie die Kästchen wie folgt:

√ “Machen wir gut”
∆ “Können wir besser machen und ich weiß, womit es losgeht”
? “Können wir besser machen – nur wie?”
nz “Nicht zutreffend” / “Bringt keinen Vorteil”

Wenn Sie nicht mit Scrum arbeiten, benutzen Sie die Zeichen so:

√ “Machen wir gut” / “Wäre einfach, gut zu machen”
∆ “Wäre eine Herausforderung, und ich weiß, wie wir begegnen können”
? “Wäre eine Herausforderung, und ich habe keine Ahnung, was wir tun können”
nz “Nicht zutreffend” / “Bringt keinen Vorteil”

Wenn Sie alle Einträge markiert haben, beschreiben Sie zwei bis sechs Organisationshindernisse auf dem Datenblatt “Organisationshindernisse” auf der vorherigen Seite. Drucken Sie es mehrmals aus!
Es spielt keine Rolle, ob Sie die Hindernisse aus dieser Liste ableiten. Wählen Sie Hindernisse, bei denen Sie zumindest ein Prozent Hoffnung haben, dass Sie etwas daran ändern können.