Handreichung für Scrum Master im Praktikumseinsatz der Abteilung BIS

Simon Johanning, Hans-Gert Gräbe

Version vom 10. Januar 2016

Inhaltsverzeichnis

Dieses Dokument **ScrumMaster.pdf** setzt grundlegende Bekanntschaft mit der Scrum Entwicklungsmethodik voraus und beschreibt Modifikationen und bewährte Praxen des Einsatzes der Methodik zur Führung verschiedener Praktika an der Abteilung BIS.

Für eine Übersicht über die Scrum-Methodik siehe bspw. **Scrum.pdf**.

Als *Scrum Master* kommt in der Regel eine SHK als Tutor, als *Project Owner* eine Lehrkraft als Betreuer zum Einsatz.

1 Allgemeines

1.1 Spezifika und Zusammenhang der verschiedenen Rollen

Bei der Arbeit von *Scrum Master* und *Project Owner* ist zu berücksichtigen, dass das *Entwicklungsteam* sich selbst organisiert. Weder der *Project Owner* noch der *Scrum Master* geben vor, welches Teammitglied wann was macht und wer mit wem zusammenarbeitet. Der *Scrum Master* hat die Aufgabe, darauf zu achten, dass keiner in diesen Selbstorganisationsprozess eingreift und das Team stört oder Verantwortlichkeiten an sich zieht, die ihm nicht zustehen.

In der Implementierungsphase haben Team-Mitglieder bisweilen Schwierigkeiten, die interdisziplinären Anforderungen zu akzeptieren. So kann z.B. ein Entwickler nicht verstehen, warum er nun auch noch die Arbeit eines Testers leisten soll. Hinter diesen Anforderungen steht jedoch der Gedanke, dass ein starkes *Entwicklungsteam* den mannigfachen Unwägbarkeiten eines Projektes wesentlich besser gewachsen ist als eine Sammlung individueller Talente.

Falls beispielsweise jemand mit einer Aufgabe nicht zurechtkommt, kann ein anderer aushelfen und so die Einhaltung des *Sprint*-Ziels gewährleisten. Und fällt jemand aus privaten Gründen für einige Zeit aus, ist ein interdisziplinär aufgestelltes *Entwicklungsteam* besser in der Lage, die fehlende Expertise zu kompensieren.

1.2 Besonderheiten im Praktikumseinsatz

- Es gibt ein *Weekly Scrum* statt *Daily Scrum* von maximal einer Stunde Dauer.
- *Sprints* sind auf etwa vier Wochen ausgelegt und enden mit einem längeren *Sprint Review*, zu dem das Team die erreichten Ergebnisse in strukturierter Form (Meilenstein) präsentiert, und einer *Sprint Retrospective*, in der auf die Prozesse im letzten *Sprint* reflektiert wird.

  Daran schließt sich unmittelbar das nächste *Sprint Planning* an. Hierbei ist zu beachten, dass ausreichend Zeit für das Treffen eingeplant oder das Treffen in ein Abschlusstreffen mit *Sprint Review* und *Sprint Retrospective* und ein Auftakttreffen mit dem *Sprint Planning* geteilt wird. In der zweiten Variante ist es wichtig, dass die Treffen zeitlich nahe beieinander liegen (max. 1 – 2 Tage Abstand), sodass kein Vakuum entsteht.
- Es sollten unserer Erfahrung nach in jeder Woche *zwei* Termine für Treffen anberaumt werden, eines davon als *Weekly Scrum*. Das andere Treffen sollte für die inhaltliche Arbeit wie *Backlog Grooming*, *Story Time*, *Release Planning Meeting* oder themenorientierte Treffen zu akuten Problemen und Diskussionen innerhalb des Projektes verwendet werden.

Es sollte den Studenten vermittelt werden, warum eine organisatorische Trennung dieser Treffen notwendig ist.

- Es ist davon auszugehen, dass sich der *Scrum Master* am besten mit der Scrum Entwicklungsmethodik auskennt (auch besser als der *Project Owner*) und mit diesen Kenntnissen den *Project Owner* bei der Scrum-gerechten Aufbereitung seiner Vorgaben und Zuarbeiten unterstützt.
- Eine wichtige Besonderheit der Scrum-Methodik ist die große Bedeutung einer angemessenen *Dokumentation der Scrum-Prozesse* durch das Team. Der *Scrum Master* sollte darauf achten, dass diese teaminternen Prozesse auf strukturierte Weise und zeitnah dokumentiert werden und das Team dazu ggf. anhalten.

1.3 Zur Rolle des Scrum Masters

Der *Scrum Master* ist primär verantwortlich für die *Überwachung* der Einhaltung von Vorgaben für die Organisation der Prozesse und Erstellung der Dokumente sowie für die *Beseitigung von Hindernissen*. Dazu gehören Probleme im Entwicklungsteam (z. B. mangelnde Kommunikation und Zusammenarbeit, persönliche Konflikte) und im Scrum-Team (z. B. in der Zusammenarbeit zwischen *Project Owner* und *Entwicklungsteam*) sowie äußere Störungen. Zu seinen Aufgaben gehört ausdrücklich *nicht* das Anlegen und Warten der Dokumente. Die Verantwortlichkeit hierfür liegt beim *Entwicklungsteam*.

Ein *Scrum Master* ist gegenüber dem *Entwicklungsteam* eine Führungskraft, aber kein Vorgesetzter. Weder kann er einzelnen Team-Mitgliedern konkrete Arbeitsanweisungen geben, noch kann er diese beurteilen oder disziplinarisch belangen. Seine Rolle ist die eines Servant Leaders – der *Scrum Master* sichert sich Anerkennung und Gefolgschaft, indem er sich der Bedürfnisse der Team-Mitglieder annimmt.

Bei der Rollenaufteilung ist zu berücksichtigen, dass das Team sich selbst organisiert. Weder der *Project Owner* noch der *Scrum Master* geben vor, welches Teammitglied wann was macht und wer mit wem zusammenarbeitet. Auch bei der Zusammenstellung des *Sprint Backlog* kommt dem Scrum Master nur eine beratende Rolle zu. Er hat die Aufgabe, darauf zu achten, dass niemand in diesen Selbstorganisationsprozess eingreift und das Team stört oder Verantwortlichkeiten an sich zieht, die ihm nicht zustehen.

Für den *Scrum Master* hat sich bewährt, dass dieser das Ziel des Projektes ständig im Blick behält und dem Team seine Wahrnehmung der zu bearbeitenden Aspekte (insbesondere während des *Sprint Planning Meeting*) in Bezug auf das Projektziel mitteilt, sowie dem *Entwicklungsteam* auch inhaltlich beratend zur Seite steht.

Der Scrum Master ist verantwortlich

- für die organisatorische Planung und Durchführung von *Sprint Planning*, *Sprint Retrospective*, *Sprint Review*, *Backlog Grooming* und *Release Planning* Treffen sowie für die Planung und Moderation der *Story Time*,
- für die Leitung der *Weekly Scrum* Treffen und
- für die Führung des *Impediment Backlogs* einschließlich einer personenbezogenen Aufwandserfassung.

In enger Abstimmung mit dem *Project Owner* (Betreuer) wird weiter das *Project Backlog* geführt und dort in Zusammenarbeit mit dem Entwicklungsteam die *User Stories* eingetragen, welche Funktionalitäten aus der Sicht der Anwender beschreiben sollen. Die eingetragenen *User Stories* werden regelmäßig priorisiert. Die Festlegungen des *Project Owners* sind verbindlich und dürfen nicht überstimmt werden.

2 Scrum Artefakte

2.1 Project Backlog

Für den *Project Owner* und den *Scrum Master* ist wichtig, dass die Items des *Project Backlogs* feingranularer werden sollten, je höher sie im Backlog stehen, d.h. je akuter sind, da eine gute Ausarbeitung der weiter oben stehenden (und somit vermutlich in den neuen *Sprint* aufzunehmenden) Items das *Sprint Planning Meeting* verkürzen kann, und sich die Anforderungen für weiter unten stehende noch ändern können.

Weiterhin ist wichtig, dass das *Entwicklungsteam* ein gutes Verständnis für die Items des *Project Backlogs* entwickelt. Hat der *Project Owner* oder *Scrum Master* das Gefühl, dass im *Entwicklungsteam* ein solches Verständnis nicht ausreichend vorhanden ist, sollte so bald wie möglich eine *Story Time* anberaumt werden.

2.2 Sprint Backlog

Die Items dieser Liste sollen klein genug gehalten werden, um während eines *Sprints* realisiert zu werden.

Es ist weiterhin darauf zu achten, dass die *User Stories*, die aus dem *Project Backlog* übernommen werden, möglichst schnell in *User Tasks* heruntergebrochen werden. Ist dies zeitlich nicht während des *Sprint Planning Meetings* möglich, sollte hierzu so bald wie möglich ein themenorientiertes Treffen anberaumt werden, um dies zu erreichen, da sich im *Entwicklungsteam* sonst leicht Leerlauf einstellt, was der Produktivität des Teams abträglich ist.

Darüber hinaus ist eine gewissenhafte Dokumentation der *User Tasks* durch das Entwicklungsteam entscheidend dafür, dass das Team die Übersicht über den Erfüllungsstand der im *Sprint* zu erledigenden Aufgaben hat und keine davon in Vergessenheit gerät (siehe *Burndown Chart*). Unserer Erfahrung nach wird dies gerne vom *Entwicklungsteam* vernachlässigt, und der *Scrum Master* sollte darauf drängen, dass diese Liste gepflegt wird (inklusive geschätztem Aufwand und Verantwortlichkeiten), da ansonsten mitunter wichtige *User Tasks* nicht beachtet werden.

Im Praktikum werden normalerweise *Sprint Backlog*, *User Tasks* und *Burndown Chart* in einem Dokument geführt.

2.3 Story Map

Die *Story Map* wird unserer Erfahrung nach wenig benutzt, auch weil Prozesse, die mit der Findung und Organisation von *User Stories* zu tun haben, bisher zu wenig benutzt werden.

Hier ist darauf zu achten, dass hinreichend oft, besonders zu Beginn des Projektes die Aktivitäten, *Story Time* und *Backlog Grooming* (und auch, wenn auch in geringerem Maße, *Release Planning Meeting*) durchgeführt werden.

Da in unseren Projekten im allgemeinen wenig oder keine Einbeziehung von Stakeholdern stattfindet und die *Stories* in der Regel vom *Project Owner* und dem *Entwicklungsteam* stammen, sollte der *Project Owner* stärker in das Verfassen von *Stories* eingebunden werden als dies in anderen Anwendungskontexten der Scrum-Methodik der Fall ist.

2.4 Project Vision

Die *Project Vision* sollte vom *Project Owner* (evtl. in Zusammenarbeit mit dem *Scrum Master*) bereits vor dem Projekt geschrieben und den Studenten bei der Projektvorstellung präsentiert werden, um den nötigen Einblick in das Projekt zu geben.

Im SWT-Praktikum wird die *Project Vision* intentional zum Ende des zweiten Sprints noch einmal präzisiert und dann fixiert.

Die *Project Vision* sollte insbesondere beim Herleiten von *User Stories* und bei Unklarheiten über die Natur des Projektes während Diskussionen als Referenz benutzt werden.

Es ist bei der *Project Vision* aufgrund ihrer Wichtigkeit und Unveränderbarkeit auf die genaue Formulierung zu achten, um diese als Referenz für das gesamte Projekt verwenden zu können.

2.5 Release Plan

Da das *Sprint Planning Meeting* zeitlich sehr umfangreich werden kann, und gegen Ende das Risiko besteht, dass wegen Zeitmangel oder mangelnder Konzentration unachtsam diskutiert wird, ist es von überragender Bedeutung, dieses gut vorzubereiten.

Ein Werkzeug hierfür stellt der *Release Plan* dar, da dieser Diskussionen, die sonst im *Sprint Planning Meeting* stattfinden, auf das *Release Planning Meeting* verlagert, und so Zeit für konkretere und produktivere Diskussionen im *Sprint Planning Meeting* schaffen. Im Idealfall sollte sich bei einem gut durchdachten und diskutierten *Release Plan* das *Sprint Planning Meeting* auf das Festschreiben der Rahmen für den Sprint (Fixierung des *Sprint Goal* sowie der fertigzustellenden *Arbeitsprodukte*), eine Evaluation des *Release Plans*, das Herunterbrechen der *User Stories* in *User Tasks* sowie die Übernahme der Verantwortung für diese beschränken.

Der *Release Plan* sollte sich an den Erfahrungen im Entwicklungsprozess und dem *Backlog Grooming* orientieren.

2.6 Burndown Chart

Wie im Dokument **Scrum.pdf** angerissen, gibt es für die Y-Achse einer strukturierten *Burndown Chart* eine Vielzahl von Möglichkeiten. Es wird davon abgeraten, dafür die Anzahl der *User Stories* oder *User Tasks* zu verwenden, da hierdurch ein falscher Eindruck vom Fortschritt im *Sprint* gegeben wird, da verschiedene Items trotz unterschiedlicher Aufwandsschätzung den gleichen Wert haben und somit schnell der tatsächliche Fortschritt über- oder unterschätzt wird.

Es empfehlen sich daher Einheiten wie Stunden der *User Tasks* oder Story Points der *User Stories*.

Wichtig für eine gute Scrum-Kultur ist es, dass Items erst dann als erledigt gelten, wenn alle Aspekte behandelt sind. Für den *Scrum Master* ist es wichtig, dass dem *Entwicklungsteam* vermittelt wird, dass somit der Verlauf nicht linear ist, sondern Fortschritt zu Anfang des *Sprints* sehr langsam scheint, und dies das *Entwicklungsteam* nicht entmutigen sollte.

Im Praktikum wird im Normalfall auf die Führung einer strukturierten *Burndown Chart* verzichtet und diese zusammenm mit dem *Sprint Backlog* in einem Dokument geführt. Dabei können *User Tasks* aus dem *Sprint Backlog* nach Fertigstellung direkt in die *Burndown Chart* übernommen und mit Angaben zu Bearbeiter und Zeitverbrauch angereichert werden.

Es kann sinnvoll sein, die Items der *Burndown Chart* wieder thematisch zu gruppieren, um den Fortschritt an verschiedenen „Fronten“ des Sprints darzustellen.

2.7 Stärken und Schwächen Profil

Kenntnis der Stärken und Schwächen der einzelnen Teammitglieder spielt eine wichtige Rolle (untereinander sowie für *Scrum Master* und *Project Owner*), um Stärken zur Geltung zu bringen und Schwächen zu kompensieren. Ein *Stärken und Schwächen Profil* sollte so früh wie möglich im Projekt angefertigt werden, in jedem Fall innerhalb des ersten *Sprints*.

3 Scrum Prozesse

3.1 Sprint Planning Meeting

Unserer Erfahrung nach ist es im *Sprint Planning Meeting* neben den in **Scrum.pdf** spezifizierten Prozessen ebenfalls wichtig, eine Grobplanung für den *Sprint* festzulegen. Ist der *Release Plan* hinreichend gut ausgearbeitet, kann dieser als Vorlage dienen. Im Verlauf des *Sprints* sollte darauf geachtet werden, dass die zu erledigenden Aufgaben sich an dieser Grobplanung orientieren und bei Abweichungen zeitnah die relevanten Gründe diskutiert werden. In vergangenen Projekten hat sich gezeigt, dass aus bestimmten *User Tasks*, die für den Verlauf des Projektes wichtig waren oder wertvolle Erfahrungen für zukünftige Entscheidungen hätten liefern können, die gewünschten Impulse oft zu spät kamen. Dies kann mit einer stärkeren Berücksichtigung der Grobplanung von *Sprints* und *Release Plan* verhindert werden.

In der Literatur wird von den *Sprint Planning Meetings* gefordert, dass *User Stories* in *User Tasks* umgesetzt werden. Je nach Länge und Qualität der Vorbereitung (siehe *Release Plan*) kann es jedoch sinnvoll sein, dies auf ein späteres Treffen zu verschieben, das jedoch so zeitnah wie möglich stattfinden sollte, damit das *Entwicklungsteam* nicht in ein arbeitstechnisches Vakuum fällt.

3.2 Sprint Review Meeting

Das *Sprint Review Meeting* sollte vor allem dazu dienen, dass das *Entwicklungsteam* seine Arbeit im Kontext des *Sprints* präsentieren und „im Kopf“ abschließen kann. Zwar werden die abgeschlossenen *User Tasks* bereits in den *Weekly Scrums* präsentiert, doch bietet dieses

nicht den Rahmen, um die *User Tasks* im größeren Zusammenhang des *Sprint Goals* zu sehen und zu diskutieren.

Im Bezug auf nicht umgesetzte *User Tasks* ist es wichtig, den Grund für die Nichtumsetzung zu diskutieren. Dies sind meistens Impediments prozessualer Natur (bspw. zu viele Commitments in diesem Sprint übernommen, keine realistische Aufwandsschätzung betrieben) und somit besser in der *Sprint Retrospective* angesiedelt. Dies können jedoch auch Impediments inhaltlicher Natur sein, um die sich der *Scrum Master* im nächsten *Sprint* besonders kümmern sollte. Oftmals sind diese Impediments dem *Entwicklungsteam* nicht bewusst und werden daher auch nicht an den *Scrum Master* weitergegeben. Dieser sollte daher auch immer ein Ohr für nicht ausgesprochene Impediments haben.

3.3 Sprint Retrospective

Die *Sprint Retrospective* ist ein Treffen, das meist vom *Entwicklungsteam* stiefmütterlich behandelt wird. Fasst sich das *Entwicklungsteam* hierbei sehr kurz (bspw. „alles gut"), sollte der *Scrum Master* einzelne Prozesse oder Artefakte mit dem *Entwicklungsteam* diskutieren, um zu eruieren, welche hiervon als besonders wertvoll oder zeitverschwenderisch empfanden wurden und wie letzteres verändert werden kann.

Die *Sprint Retrospective* sollte länger als fünf Minuten dauern, was ohne Anstrengung des *Scrum Masters* leider selten der Fall ist.

3.4 Weekly Scrum

Beim *Weekly Scrum* sollte der *Scrum Master* darauf achten, dass dieser möglichst kurz gehalten wird und keine Diskussionen zu *Project Backlog*, *Sprint Backlog*, *Story Map* oder weiteren Planungsartefakten entstehen. Dazu sollen andere Treffen zu spezifischen Prozessen dienen. Spezifische Probleme können (besonders, wenn sie nur einen Teil des Teams betreffen), anschließend im informellen Rahmen geklärt werden.

Außerdem sollten im *Weekly Scrum* Präsentation und Diskussion getrennt werden, da unserer Erfahrung nach eine Diskussion oftmals auch Bereiche folgender Berichte umfasst, diese dann als besprochen verzeichnet und nicht mehr vollständig ausgeführt werden. Außerdem werden in Diskussionen oft einzelne Teammitglieder ausgeschlossen und es geht der Fokus verloren, den das Team vor allem zu Beginn des *Weekly Scrum* noch hatte.

Die Diskussionen sollten nach der Präsentation über die Tätigkeiten der letzten Woche geführt werden, aber vor der Planung für die nächste Woche, und sollten organisatorisch hiervon getrennt werden.

Unserer Erfahrung nach lassen sich die meisten derartigen Fragen außerhalb des *Weekly Scrum* klären. Eine Diskussion während des *Weekly Scrum* steht dem konzentrierten Arbeiten zum Abgleich der erreichten Ergebnisse entgegen, da sich Aufmerksamkeit und Fokus der Projektteilnehmer von einer zeiteffizienten Vorgehensweise abwenden und so eventuell wichtige Fragen bzw. Punkte nicht mehr besprochen werden.

Aufgrund des Charakters der Projekte als studentische Projekte, und da in der Regel mehrere Treffen pro Woche durchgeführt werden, bietet es sich an, spezielle Treffen nach dem *Weekly Scrum* durchzuführen, diese aber bewusst (zumindest organisatorisch) vom *Weekly Scrum* zu trennen.

3.5 Story Time

Die *Story Time* sollte vor allem zu Anfang des Projektes, vor *Release Planning Meetings* und bei Bedarf abgehalten werden. Das gilt insbesondere, wenn der *Scrum Master* das Gefühl hat, dass das *Entwicklungsteam* keinen klare oder zum *Project Owner* divergierende Vorstellungen entwickelt.

Die *Story Time* sollte „auf die Zukunft“ schauen und *User Stories* umfassen, die im folgenden *Sprint* zu bearbeiten sind (da die Items des *Sprint Backlogs* dem Team klar sein sollten), aber aus diesem Grund auch nicht zu lange dauern (maximal eine Stunde).

Weiterhin dienen diese Treffen dazu, die Items des *Project Backlogs* zu verfeinern oder zu reformulieren, um Missverständlichkeiten zu reduzieren.

3.6 Backlog Grooming

Backlog Grooming ist ein Prozess, der besonders am Anfang des Projektes von hoher Bedeutung ist. Ergibt sich während der Entwicklung, dass falsche Annahmen zu Items des *Project Backlogs* gemacht wurden, sollte zudem möglichst zeitnah ein *Backlog Grooming* durchgeführt werden.

Ein erstes *Backlog Grooming* sollte bereits beim ersten (konstruktiven) Arbeitstreffen durchgeführt werden, um aus der *Project Vision* erste *Epics* und *User Stories* abzuleiten. Es ist weiterhin empfehlenswert, den *Backlog Grooming* Treffen ein Ziel vorzugeben, um die Arbeit zu fokussieren.

3.7 Release Planning Meeting

Das *Release Planning Meeting* sollte zu Anfang des Projektes und nach Bedarf durchgeführt werden. Es kann mit einem *Backlog Grooming* kombiniert werden.

Da der *Release Plan* eine wichtige Rolle für ein erfolgreiches *Sprint Planning Meeting* ist, sollte dieses Treffen nicht unterschätzt werden.