FOM Hochschule für Oekonomie & Management Essen
Standort München
Berufsbegleitender Studiengang zum B.Sc. Wirtschaftsinformatik

Seminararbeit

Scrumban - eine Kombination von Scrum und Kanban

Eingereicht von:

Oliver Kurmis

Matrikel-Nr: 328091

Betreuer: Dipl.-Ing. (Univ.) Felix Dutkowski

Abgegeben am:

28. Februar 2015

Erarbeitet im:

4. Semester

Inhaltsverzeichnis

Abkürzungsverzeichnis

CD	Continuous Delivery - Kontinuierliche Auslieferung von Software-Inkrementen
KVP	Kontinuierlicher Verbesserungsprozess
PO	Product Owner
WIP	Work In Progress - aktuell in Arbeit
WP	Work Package - Arbeitspaket
XP	Extreme Programming

Abbildungsverzeichnis

1 Einleitung

Agile Vorgehensmodelle bei der Softwareentwicklung haben sich in der Wirtschaft seit der Jahrtausendwende mehr und mehr verbreitet. Scrum ist hierbei eines der am meisten verwendeten Modelle. Mit seinem festen Regelwerk eignet sich Scrum jedoch nicht für alle Softwareentwicklungsaufgaben gleichermaßen. Teilweise ist Scrum auch eher ungeeignet, etwa bei Support-Aufgaben oder bei sehr kleinen Teams. Manche solcher Teams setzen dann das flexiblere und sehr unspezifische Kanban ein. Es ist aber auch ein Mittelweg zwischen Kanban und Scrum möglich, um die Vorteile aus beiden Vorgehensmodellen zu vereinigen und die Nachteile zu vermeiden oder zumindest zu vermindern. Dieser Mittelweg wird häufig auch *Scrumban* genannt.
Im Abschnitt zwei der Arbeit wird das Vorgehensmodell Scrum mit seinen Rollen, Artefakten und Prozessen kurz vorgestellt. Abschnitt drei ist eine Einführung in Kanban in der Produktion und in der IT und die damit eng verbundene Kaizen-Philosophie, auch bekannt als kontinuierlicher Verbesserungsprozess. Eine Gegenüberstellung von Scrum und Kanban erfolgt in Abschnitt 4 der Arbeit. Abschnitt 5 beschreibt Scrumban als eine Mischung der Methoden von Scrum und Kanban, um für die spezifischen Bedürfnisse eines Teams ein geeignetes Vorgehensmodell zu finden. Im letzten Abschnitt wird ein Fazit gezogen und ein Ausblick zu dem Thema Scrumban in der Softwareentwicklung gegeben.

2 Scrum

2.1 Einführung in Scrum

Scrum ist eine agile Alternative zu den klassischen Projektmanagement-Methoden, und wird seit den frühen 1990er Jahren eingesetzt. [1] Im Gegensatz zu den traditionellen Vorgehensmodellen wird bei Scrum das Projekt nicht von Anfang an komplett durchgeplant und dann Phase für Phase umgesetzt, sondern in kleinere Arbeitspakete zerlegt. Diese werden dann gebündelt und der Priorität nach in sog. Sprints (fixe Zeitperioden von typischerweise 2 bis 4 Wochen) abgearbeitet. Bei ständigem Kontakt mit dem Auftraggeber entstehen so in regelmäßigen Abständen lauffähige Software-Inkremente und bei Abweichungen von dem gewünschten Weg kann schnell korrigiert und gegengesteuert werden.

[1] vgl. [Ken Schwaber und Jeff Sutherland (2013)]

2.2 Rollen

Bei Scrum werden 3 Rollen unterschieden: *Product Owner* (PO) , *Entwicklungsteam* und *Scrum Master*.
Der *Product Owner* ist derjenige, der dem Entwicklungsteam gegenüber die zu lösende Implementierungsaufgabe definiert, er ist dem Team gegenüber der Auftraggeber. Er ist verantwortlich für die Eigenschaften und den wirtschaftlichen Erfolg des Produktes. Der Product Owner kann der tatsächlicher Auftraggeber der Software sein (ein externer oder interner Kunde), oder es ist jemand, dem diese Aufgabe delegiert wurde. Primäre Aufgaben des Product Owner sind die Anforderungen zu definieren, zu priorisieren und der fertig gestellten Funktionalitäten abzunehmen.
Dem *Entwicklerteam* kommt die Aufgabe zu, das Softwareprodukt gemäß den Anforderungen zu implementieren und dabei auch die geforderten Qualitätsstandards einzuhalten. Das Team organisiert sich und seine Arbeit selbst, dazu besteht es aus Mitgliedern verschiedener Fachgebiete, z. B. Architektur, Entwicklung, Testen, Dokumentation und Datenbank.
Der *Scrum Master* tritt als Schiedsrichter auf, er trägt dafür Sorge, dass die vereinbarten (Scrum-)Regeln von allen Beteiligten eingehalten werden. Bei den regelmäßigen Meetings wirkt er als Moderator. Treten Probleme auf, z.B. externe Blockaden, versucht der Scrum Master diese zu beheben um zu gewährleisten, dass das Team in seiner Arbeit nicht ausgebremst wird. Vor allem bei der Einführung von Scrum und ersten Projekten ist der Scrum-Master wichtig für das Gelingen. Später, wenn PO und Team bereits Erfahrung mit Scrum haben, kann er sich mehr und mehr zurückziehen und sich auf die Moderation der Meetings beschränken.

2.3 Artefakte

Das *Product Backlog* ist eine Auflistung aller aktuellen Anforderungen an das Produkt. Es ist dynamisch (d.h. änderbar) und wird vom Product Owner gepflegt. Es können neue Anforderungen hinzukommen und bestehende können auch wieder obsolet werden. Die Einträge im Product Backlog werden nach wirtschaftlichem Nutzen und technischer Notwendigkeit priorisiert. Je höher ein Eintrag priorisiert ist, umso genauer müssen die Anforderungen bekannt und formuliert sein. Im nächsten Sprint werden dann die am höchsten priorisierten Einträge bearbeitet.
Das *Sprint Backlog* enthält die Einträge, die im aktuellen Sprint bearbeitet werden. Hierfür müssen die Aufwände der einzelnen Aufgaben zuerest vom Team geschätzt werden, damit nur so viel Arbeit in den Sprint aufgenommen wird, wie auch bewältigt werden kann. Alle Aufgaben kommen dann auf die Aufgabentafel (Taskboard), häufig mit den Spalten *To Do*, *In Progress* und *Done*.
Im Verlauf eines Sprintes entsteht ein neues *Product Increment*, ein Software-Artefakt

als Ergebnis aus den Implementierungen aller bisherigen Sprints. Am Endes eines Sprints sollte eine lauffähige und potentiell nutzbare Software stehen, die der Product Owner bzw. der Kunde begutachten kann.

2.4 Der Scrum-Prozess

Am Anfang der Software-Erstellung stehen die Anforderungen des Kunden. Diese sind oft noch unvollständig oder vage. Der Kunde bzw. der Product Owner formuliert seine Anforderungen daher erst mal grob so weit sie bekannt sind und fasst sie als Ansammlung von *User Stories* im Product Backlog zusammen. Die wichtigsten Funktionen werden entsprechend priorisiert und müssen nun noch hinreichend genau spezifiziert werden, so dass das Entwicklungsteam den Aufwand schätzen kann.
Am Anfang eines Scrum-Entwicklungszyklus, dem sog. *Sprint*, schätzt das Team im Sprint Planning Meeting die User Stories mit der höchsten Priorität und zerlegt sie in kleinere Teilaufgaben. Es werden dann vom Team nach absteigender Priorität so viele Aufgaben für den aktuellen Sprint Backlog zur Implementierung ausgewählt, wie mit den gegebenen Ressourcen bearbeitet werden können. Das Team verpflichtet sich, diese Aufgaben bin zum Ende des Sprints umzusetzen (commitment). Teilweise wird anstatt *Verpflichtung* auch der Begriff *Prognose* verwendet, da die Erfahrung zeigt, dass die Qualität des Produktet leidet, wenn das Team verpflichtet ist, gegen Ende des Sprints noch unbedingt alle Aufgaben fertigstellen zu müssen.[2] Im Anschluss an das Sprint Planning werden die Aufgaben aus dem Sprint Backlog auf Karteikarten oder kleine Zettel geschrieben und im neuen (bereinigten) Task Board in der Spalte *ToDo* plaziert.
Innerhalb eines Sprints organisiert das Team seine Arbeit selbst, der PO und der Scrum Master dürfen keine Anweisungen geben. Jeden Tag trifft sich das Entwicklungsteam in einem kurzen *Daily Standup Meeting* in dem jedes Teammitglied über seine Arbeit (aktuelle, erledigte und nächste Aufgaben, Probleme) berichtet. Am Ende des Sprints wird im *Sprint Review Meeting* das Produkt-Inkrement dem PO bzw. dem Kunden präsentiert und die Implementierungen werden abgenommen (oder zurückgewiesen und wieder in das Product-Backlog aufgenommen).
In der den Sprint abschliessenden *Sprint-Retrospektive* wird der Verlauf des Sprint kritisch im Rückblick überprüft. Es nehmen nur das Team und der Scrum Master teil und untersuchen, was für Probleme es gab und was besser gemacht werden kann.
Abbildung 2 veranschaulicht den Ablauf des Entwicklungsprozesses mit Scrum (Sprint-Retrospektive und Aufgabentafel).

[2] vgl. [Jose Luis Soria Teruel (2011)]
[3] Quelle: [Sabine Rossbach (2013)] `http://blog.doubleslash.de/10-erfolgsfaktoren-fur-das-anforderungsmanagement-in-agilen-software-grosprojekten/`

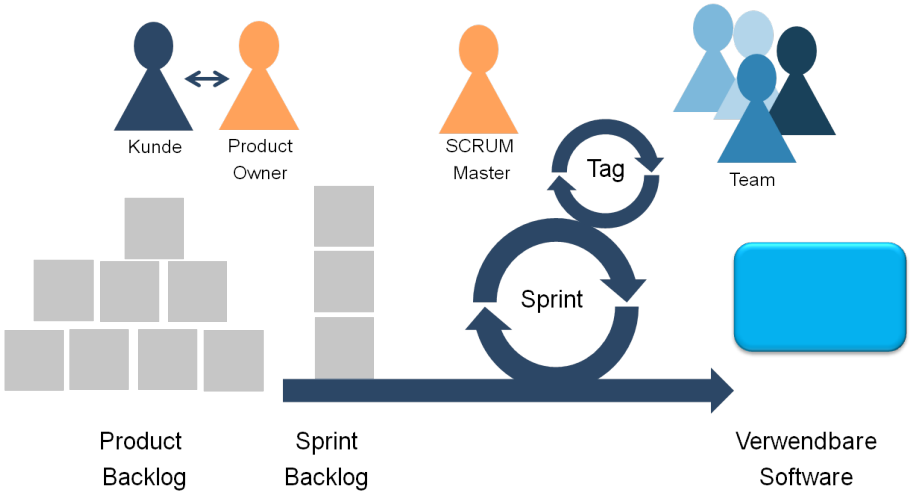

Abbildung 1: Vereinfachte Darstellung der Scrum-Vorgehensweise[3]

2.5 Vorteile und Nachteile von Scrum

Die weite Verbreitung von Scrum deutet bereits darauf hin, dass es als Werkzeug funktioniert und viele Projekte zur Zufriedenheit der Stakeholder abgeschlossen werden konnten. Bei der Einführung von Scrum in einer klassisch gemanagten Organisation sind die klaren Vorgaben und Handlungsanweisungen von Scrum ein Vorteil. Diese strenge Struktur kann sich aber auch im Verlauf der Zeit als Nachteil herausstellen.

Vorteile:

- schafft Transparenz für alle Beteiligten, vermeidet bösen Überraschungen zum Ende des Projektes
- fördert Teamgedanken
- fordert Eigenverantwortung des Teams
- Entlastung des Managements / der Projektleitung
- integrierter Verbesserungsprozess durch Retrospektiven
- geeignet für verteilte Produktentwicklung in grossen Projekten mit *Scrum of Scrums* [4].

Nachteile:

- geringere Planbarkeit: zum Projektstart kaum belastbare Aussagen zum Aufwand möglich
- Trägheit: Änderungen und neue Anforderungen erst im nächsten Sprint umsetzbar (bis zu 2 Sprint-Perioden Verzögerung)
- Arbeit innerhalb des Sprint ist wenig strukturiert
- gegen Ende des Sprints steigt die Belastung des Entwicklerteams
- Supportaufgaben können nicht parallel bearbeitet werden ohne im Sprint zu-

[4] [Jean Pierre Berchez, Uta Kapp (2010)]

rück zu fallen, oder es muss für Support eine fixer Zeitanteil reserviert werden
- lange Zeitspanne zwischen Auftreten eines Problems und dessen Bearbeitung in der Sprint-Retrospektive
- hoher Zeitbedarf für die regelmäßigen Meetings

3 Kanban

3.1 Einführung in Kanban

Kanban wurde als Methode der Produktionsplanung und -steuerung zuerst in der japanischen Automobil-Industrie der Kriegs- und Nachkriegszeit entwickelt und eingeführt. Die Firma Toyota versuchte zu der Zeit, den Abstand zu den amerikanischen Wettbewerbern zu verringern. Hierzu sollten die Kosten der Produktion verringert und die Qualität der Produkte verbessert werden. Wegen den damals sehr begrenzten Ressourcen galt es jede Art von Verschwendung zu vermeiden. Dies gilt nicht nur für Rohstoffe, sondern auch für Zeit, Personal, Kapital.
Die seit dieser Zeit entwickelten Methoden sind heute auch als *Lean Production*, *just in Time* und *Kontinuierlicher Verbesserungsprozess* (KVP)bekannt.
Das Wort *Kanban* besteht aus den zwei Zeichen 看 (*kan*=sehen) und 板 (*ban*=Tafel, Brett) und läßt sich etwa mit Sichttafel, Aushängeschild oder auch Pendelkarte übersetzen. Diese Kanban-Karten sind ein zentraler Bestandteil im Kanban-Prozess. Im industriellen Produktionsprozess steht jede Kanban-Karte für einen Behälter einer bestimmten Grösse, der eine festgelegt Anzahl von Bauteilen enthält. Die Anzahl von Kanban-Karten für ein Bauteil oder eine Bauteilgruppe ist begrenzt, auf diese Weise soll verhindert werden, dass zu viel auf Lager produziert wird, denn große Zwischenlager binden Ressourcen. Der gesamte Produktionsprozess wird betrachtet als eine Aneinanderreihung von Quellen und Senken von Produktionsgütern, mit Zwischenlagern als Puffer. Eine Senke nimmt sich einen Behälter aus dem Zwischenlager (Pull-Prinzip), verarbeitet alle Teile darin und füllt (als Quelle) das nachgelagerte Zwischenlager. Hat das nachfolgende Zwischenlager einen bestimmten Höchststand überschritten, *darf nicht* weiter produziert werden. Wird dagegen ein bestimmter Mindeststand unterschritten, so *muss* wieder Nachschub produziert werden. Auf diese Weise werden Probleme oder Engpässe schnell sichtbar und es können entsprechende Gegenmaßnahmen unternommen werden. Andererseits können durch die mehrstufigen Zwischenlager Schwankungen bei Nachfrage, Zulieferung oder Personalstärke in gewissen Grenzen ausgeglichen werden.
Die Zwischenlager der einzelnen Produktionsstufen werden auf der *Plantafel* (auch Kanban-Tafel) visualisiert, welche an zentraler Stelle für alle Beteiligten gut sichtbar platziert wird. Für jede Produktionsstufe gibt es auf der Plantafel eine Spalte oder

Zeile mit festen Plätzen für die Kanban-Karten. Die Karten der leeren Behälter werden hier für jeden sichtbar plaziert, so dass auf einen Blick der Bestand der Zwischenlager erkennbar wird.

3.2 Kanban in der IT

Mit dem Aufkommen der agilen Vorgehensmodelle bei der Softwareentwicklung wurde auch versucht, die Methoden und Prinzipien von Kanban auf den Wertschöpfungsprozess der Software-Erstellung zu übertragen. Die Erstellung von Software und die Fertigung in der Industrie unterscheiden sich jedoch wesentlich. Die Wissensarbeit in der Softwareentwicklung ist ein kreativer Prozess mit immer neuen Aufgabenstellungen, während in der industriellen Fertigung im Wesentlichen immer wieder die gleichen Arbeitsschritte möglichst gleichmäßig ausgeführt werden. Aber es gibt auch in der Softwareentwicklung gewisse "Fertigungsstufen", wie man z.B. am Wasserfallmodell oder ähnlich auch beim V-Model gut erkennen kann: Anforderungsanalyse, Entwurf, Implementierung, Tests, Auslieferung und Inbetriebnahme. Wie bei anderen agilen Methoden auch wird bei Kanban aber die Arbeit in viele Teilpakete zerlegt und für jeden Teil werden die "Fertigungsstufen" einzeln durchlaufen. Es entsteht so ein iterativer Prozess bei dem in kurzen Abständen (lauffähige) Zwischenstufen der Software ausgeliefert werden können bzw. tatsächlich auch ausgeliefert werden. Kanban ist ein sehr leichtgewichtiger Prozess und macht nur sehr wenige Vorgaben. Die beiden wichtigsten Regeln sind:

- Visualisiere den Arbeitsfluss (auf der Kanban-Tafel)
- Begrenze die Arbeit in jeder Phase des Prozesses mit WIP-Limits

Die Kanban-Tafel ist ein grosses Whiteboard, an dem Spalten für die einzelnen Prozess-Schritte eingezeichnet sind. Die einzelnen Aufgaben werden auf kleine Karteikarten oder Zettel (Kanban-Karten) geschrieben und in der entsprechenden Spalte an das Whiteboard geheftet. Die Kanban-Karten bilden den Arbeitsfluss ab und wandern von links nach rechts über die Tafel, von einer Spalte zur nächsten. Welche Spalten das konkret sind wird nicht von Kanban vorgeschrieben. Eine Minimalversion wären drei Spalten wie *ToDo*, *In Arbeit* und *Erledigt*. Es können aber auch Spalten verwendet werden wie Backlog, Epic, Next, Analyse, Design, Implementierung, Test, Integration, Abnahme, Fertig für Release, Erledigt etc. Die tatsächlich verwendeten Spalten hängen von den Gegebenheiten vor Ort ab, es gilt das Motto *so einfach wir möglich, so genau wie nötig*. Unter Umständen kann es erforderlich sein, die Kanbantafel nachträglich noch zu ändern, um sie den konkreten Bedingungen vor Ort anzupassen. Für manche Spalten ist es sinnvoll, sie noch weiter in zwei Teilspalten zu unterteilen: links *In Arbeit* und rechts *Fertig*. Die Unterspalte *Fertig* wirkt hier als eine Art Puffer, aus der die nachgelagerte Spalte eine neue Aufgabe

zur Bearbeitung ziehen kann (Pull-Prinzip). Abbildung 2 zeigt exemplarisch eine einfache Kanban-Tafel, wie sie bei einem kleinen Entwicklungsteam im Einsatz sein könnte.

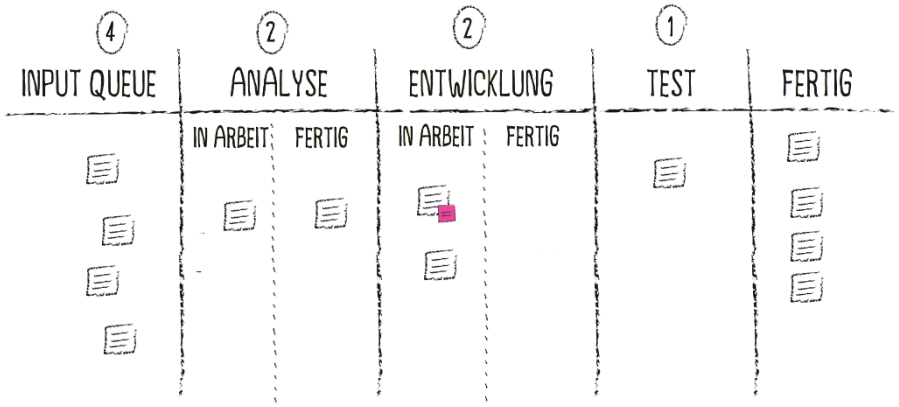

Abbildung 2: Beispiel für eine einfache Kanban-Tafel[5]

Bei der Einführung von Kanban und der Gestaltung der Kanban-Tafel muss geklärt werden, was die Grenzen des Kanban-Bereiches sind: wie kommen die Aufgaben auf die Kanban-Tafel, wie wird eine fertig gestellte Aufgabe an nachgelagerte Stellen übergeben. Bei einem kleinen Projekt könnte z.B. *Auslieferung* ein Teil der Kanban-Tafel sein, bei einem großen Projekt wird die fertige Software möglicherweise einem eigenen Release-Team übergeben (welches wiederum eine eigene Kanban-Tafel im Einsatz haben kann).
Die Begrenzung der Arbeit in den Phasen (WIP-Limit) wird durch eine Zahl im Spaltenkopf angezeigt - mehr Aufgaben, also Kanban-Karten, dürfen sich nicht in dieser Spalte befinden. Ziel dieser Begrenzung ist es, störende Unterbrechungen und häufige Kontextwechsel bei den Beteiligten zu vermeiden. Eine Aufgaben-Phase soll erst abgeschlossen werden, bevor eine neue Aufgabe begonnen wird. Wenn eine Aufgabe auf Grund einer externen Blockade im Moment nicht beendet werden kann, so kann sie mit einer Markierung als blockiert gekennzeichnet werden (siehe Spalte Entwicklung in Abbildung 2). Der Fokus der Beteiligten sollte nun sein, die Blockade zu lösen, um den Aufgabenfluss zu gewährleisten. Das richtige WIP-Limit zu finden ist eine Herausforderung, die aber durch Ausprobieren und Anpassen bewältigt werden kann. Sind die WIP-Limits zu niedrig, kommt der Arbeitsfluss oft ins Stocken, da es nicht ausreichend Puffer gibt. Ist das WIP-Limit zu hoch, kommt es nicht zur Geltung und es kommt wieder zu Multitasking, häufigen Kontextwechsel, Überlastung, Unübersichtlichkeit und vor allem zu langen Durchlaufzeiten (Lead time). Ein Ausgangspunkt für die WIP-Limits könnte z.B. sein 1,0 * Personenanzahl der Spalte, da jeder nur eine Aufgabe gleichzeitig bearbeiten kann. Sind die Prozesse

[5]Quelle: [Leopold, K., Kaltenecker, S. (2013)] S.40 Bild. 4.1

von häufigen Blockaden und hoher Variabilität geprägt, so ist ein WIP-Limit >1 * Personenanzahl sinnvoll. Wird in der Organisation Pair-Programming eingesetzt, dann wäre ein gutes WIP-Limit vielleicht 0,5 * Personenanzahl.
In Kanban sind keine bestimmten Rollen beschrieben, die Beteiligten können ihre gleiche Funktion und Rolle ausüben wie vor der Einführung von Kanban. Es kann multifunktionale Teams geben oder Spezialisten-Teams. Eine Kanban-Tafel kann zur persönlichen Organisation einer Einzelperson verwendet werden, kann sich auf ein Team, eine Abteilung oder auf die Unternehmensführung beziehen. Bei großen Projekten kann Kanban auch in einer hierarchischen Form angewendet werden: je eine Kanban-Tafel für jedes Team, für jede Abteilung, für jedes Teilprojekt und für das Gesamtprojekt.

3.3 Kaizen

Eine wesentliche Idee bei Kanban ist *Kaizen*, der kontinuierliche Verbesserungsprozess. Durch Kaizen soll der Arbeitsfluss ständig optimiert werden, Probleme sollen an ihrem Ursprung gelöst werden, Verschwendung/Vergeudung (engl. Waste) und Überlastung vermieden bzw. minimiert werden. Alle Beteiligten sind aufgefordert, sich bei diesem *Kontinuierlichen Verbesserungsprozess* einzubringen. Wenn durch eine Veränderung eine Verbesserung im Prozess erreicht wird, so wird diese Änderung zum neuen Standard erklärt.
Auch übermäßige Meeting sind im Sinne von Kaizen Vergeudung von wertvollen Ressourcen, daher sollte der Zeitbedarf für Meetings optimiert werden. Manche Teams kommen ganz ohne regelmäßige Meetings aus, andere Teams machen Daily Standup Meetings, regelmäßig oder nach Bedarf Nachschubmeetings und Release-Planungsmeetings. Meetings sollten auch abgehalten werden, wenn dafür akuter Bedarf besteht z.B. weil eine Blockade beseitigt werden muss oder weil ein Engpass besteht. Als Ergebnis ist dann eine Verbesserung des Prozesses zu erwarten. Auf diese Weise kann auch das WIP-Limit bei Bedarf angepasst werden, oder auch die Kanban-Tafel umgestaltet werden, z.B. durch mehr oder weniger Spalten, durch Einführung von Swimlanes, Ticket-Typen oder Service-Klassen.
Für eine objektive Beurteilung der Verbesserungsmaßnahmen bedarf es Kennzahlen. Wichtige Kennzahlen bei Kanban sind die Durchlaufzeit der Tickets (Lead time), der Durchsatz an Tickets und die Fehlerrate. Für aussagekräftige Kennzahlen ist es wichtig, dass die Kanbankarten von Art und Umfang vergleichbar sind. Große Aufgaben sollten daher in mittlere und kleine Aufgaben zerlegt werden, bevor sie bearbeitet werden, kleine Aufgaben sollte man Bündeln und dann an einem Stück bearbeiten. Es empfielt sich, getrennte Kennzahlen für verschiedene Aufgaben-Typen zu ermitteln, z.B. Support-Aufgaben und Implementierungs-Aufgaben zu trennen.

Die Kennzahlen sollten laufend erfasst und in Diagrammen dargestellt werden, um Probleme zu erkennen und Verbesserungen zu erfassen. Bei einem *Cumulative Flow Diagram* beispielsweise werden in regelmäßigen Abständen (z.B. wochenweise) die Anzahl der Tickets in den einzelnen Spalten der Kanban-Tafel kummulativ, also übereinander gestapelt eingetragen.

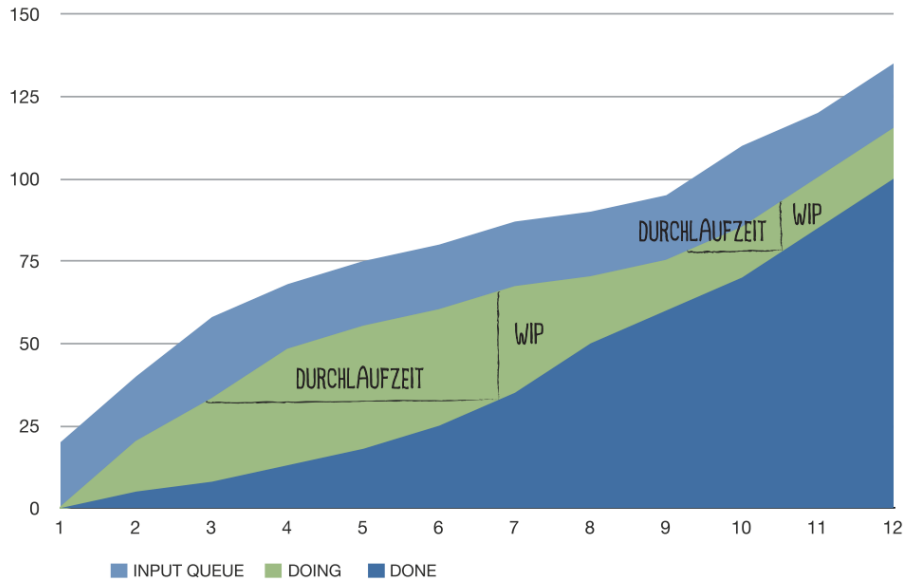

Abbildung 3: Cumulative Flow Diagram[6]

Abbildung 3 zeigt ein einfaches Cumulative Flow Diagram von einer Kanban-Tafel mit den drei Spalten *INPUT QUEUE*, *DOING* und *DONE*. An dem Diagramm lassen sich direkt die Durchlaufzeit und WIP ablesen und wie sich diese verbessert haben.

3.4 Vorteile und Nachteile von Kanban

Aufgrund der wenigen Regeln kann Kanban gut in bestehende Prozessen und Strukturen integriert werden. Es muss nicht eine plötzliche große Umstellung vorgenommen werden, sondern Kanban kann auch evolutionär Schritt für Schritt eingeführt werden.
Die wichtigsten Vorteile von Kanban sind:

- Transparenz für alle Stakeholder durch Visualisierung des Workflows
- deckt Kapazitätsengpässe im Prozess auf
- schnelle Reaktion auf Veränderung möglich
- Flexibilität: an spezifische Gegebenheiten anpassbar
- Skalierbarkeit: nutzbar für Einzelperson, kleine, mittlere und große Teams oder für eine Hierarchie

[6]Quelle: [Leopold, K., Kaltenecker, S. (2013)] S.79 Bild. 7.2

- Kontinuierliche Lieferung von Software-Inkrementen möglich (Continuous Integration)
- Vielseitigkeit: einsetzbar bei Entwicklung, Betrieb, Support, DevOps , Continuous Delivery

Auch Kanban hat natürlich Nachteile, die beachtet werden müssen:

- geringe Planbarkeit bei Umsetzung großer Projekte
- geringe Übersichtlichkeit der Kanban-Tafel bei großen Projekten und großen Teams
- längere Durchlaufzeiten bei vielen und grossen Zwischenpuffern
- externe Kunden erfordern i.d.R. eine Aufwandschätzung
- Kanban-Tafel schwierig zu implementieren bei externen Teams oder Teammitgliedern (g.g.f. elektronisch)

4 Vergleich Scrum und Kanban

Scrum und Kanban haben viele Gemeinsamkeiten und sind sich in manchen Punkten ähnlich. Beide sind agil und schlank (lean) und verwenden das Pull-Prinzip, haben eine gewisse Beschränkung der WIP und unterstützen einen kontinuierlichen Verbesserungsprozess durch Schaffung von Transparenz. Scrum und Kanban setzen beide auf sich selbst organisierende Treams und auf die eigenverantwortliche Mitarbeit und teilen die Arbeite in kleinere Arbeitspakete auf.[7] Jedoch macht Scrum sehr viel mehr Vorgaben und erlaubt kaum Variationen. Kanban ist leichtgewichtiger und anpassbar, nicht das Vorgehensmodell steht im Mittelpunkt sondern die zu erledigenden Aufgaben in ihrer angestammten Umgebung.

[7] vgl. [Kniberg, H., Skarin, M. (2010)], S. 49

	Scrum	**Kanban**
Iterationen	1-4 Wochen	optional
Zuteilung von Aufgaben	Push und Pull	Pull
Rollen	Product Owner, Scrum Master, Team	keine Vorgabe
Aufwandschätzung	vorgeschrieben	optional, grob
Kennzahlen	Team-Geschwindigkeit (Velocity)	Lead time, Zykluszeit, Durchsatz
Visualisierung	Burndown	Cumulative Flow Diagram
Verbesserungsprozess	Sprint-Retrospektive	Kaizen wenn notwendig
Cross-funktionale Teams	vorgeschrieben	optional, Spezialistenteams möglich
WIP-Limit	indirekt durch Storypoints/Sprint	direkt
Priorisierung	vorgeschrieben	optional
Tafel	1 Scrum board pro Team	1 Kanban-Tafel auch für mehrere Teams
Pflege der Tafel	mit jedem Sprint neu	kontinuierlich

5 Scrumban

Scrum und Kanban sind zwei Werkzeuge der Software-Erstellung, die je nach Situation angewendet werden müssen. So ist Scrum bei mittelgroßen Projekten mit klaren Anforderungen in reifen Organisationen sehr gut geeignet. Kanban eignet sich gut für Support-Team, junge Startups, Projekte mit häufig wechselnden Anforderungen und Prioritäten. Eine Organisation kann aber auch (bewusst oder unbewusst) beide Werkzeuge nutzen und das beste daraus kombinieren, wodurch eine Mischform aus Scrum und Kanban, also Scrumban entsteht. Der Begriff Scrumban ist nicht fest definiert, es wird in der Regel verwendet als ein Vorgehensmodell, dass zwischen Scrum und Kanban liegt. Teilweise wird auch der gleichzeitige Einsatz von Scrum und Kanban in einer Organisation (z.B. auf verscheidenen Ebenene) als Scrumban bezeichnet. Viele der oben genannten Nachteile von Scrum und Kanban versucht Scrumban zu beheben. So kann man etwa Planungs- und Releasezyklus entkoppeln oder die Aufteilung in Fachteams erlauben.

5.1 Von Scrum zu Scrumban

Scrum ist heute bei vielen Organisationen verbreitet und erfolgreich im Einsatz. Mitunter müssen Teams jedoch im Lebenszyklus der Software die Arbeitsmethoden den Gegebenheiten anpassen. Scrum macht jedoch sehr genaue und strenge Vorgaben, wird von diesen Vorgaben abgewichen, so handelt es sich nicht mehr um Scrum.[8]. Macht ein Team Anpassungen am Scrum-Vorgehen in Richtung Kanban kann man also bereits von Scrumban sprechen.
Eine Aufgabentafel eines Scrum-Teams ist einer Kanban-Tafel bereits recht ähnlich. Wenn ein Team im Laufe der Zeit bemerkt, dass es nicht effizient ist, an vielen Aufgaben gleichzeitig zu arbeiten, so kann es beschliessen (da es sich ja selbst organisiert), dass eine Aufgabe zuerst abgeschlossen werden sollte bevor eine neue begonnen wird. Dies ist bereits ein erster Schritt in Richtung Kanban, da auf diese Weise ein WIP-Limit gesetzt wird. Wird dieses WIP-Limit explizit über die Spalte *In Progress* geschrieben so hat das Team schon eine einfache Kanban-Tafel. [9]. Als weitere Verbesserung könnte die Spalte *In Progress* der Aufgabentafel noch weiter unterteilt werden, um die Arbeit besser zu strukturieren und unter den Teammitgliedern aufzuteilen. Unter den Teammitgliedern wird es immer Spezialisierungen geben, so muss ein Tester nicht unbedingt programmieren können. So bekommt jede Spalte ein eigenes WIP-Limit, entsprechend der jeweiligen personellen Ressourcen. Wenn die Organisation wächst können Spezialaufgaben wie Testen, Ausrollen oder Betrieb der Software in spezielle Teams ausgelagert werden, die dann aber immer noch auf der Aufgabentafel bzw. der Kanban-Tafel mit einer Spalte repräsentiert werden.
Wird die zu erstellende Software nicht nur in Inkrementen erstellt sondern auch ausgeliefert, z.B. bei Web-Applikationen, so fallen immer auch Aufgaben für Betrieb und Support an, die vor allem in kleinen Organisationen auch vom Entwicklungsteam mit erledigt werden müssen. Diese können nicht dem Scrum-Prozess unterworfen werden da sie in der Regel zeitnah bearbeitet werden müssen und nicht bis zum Ende des nächsten Sprints warten können. Für diese Arbeiten kann ein eigene Kanban-Tafel erstellt werden wobei jedoch für Betrieb und Support auch entsprechende Zeitaufwände einzuplanen sind, die dann für den Scrum-Prozess fehlen. Besser ist es, für diese Aufgaben eine eigene *Swim Lane* auf der Aufgabentafel vorzusehen, denn nur so können die WIP-Limits wirken. Jedoch sind nun keine Zusagen (commitments) des Teams bezüglich der Arbeitspakete (Story-Points) möglich, die Teamgeschwindigkeit (Velocity) schwankt. Somit sind auch die festen Zeiträume für einen Sprint nicht mehr zwingend. Ein Planungsmeeting kann einberufen, wenn es erforderlich

[8] vgl. [Ken Schwaber und Jeff Sutherland (2013)], S. 17
[9] vgl. [Innovel, LLC (2008)]

ist, sei es weil die Aufgaben in der *INPUT QUEUE* zur Neige gehen oder weil es wichtige Änderungen in den Anforderungen gibt.
Setzt das Team *Continuous Integration* oder auch *Continuous Delivery* ein, so kann auf Releasezyklen ganz verzichtet werden. Ein Release ist dann sehr billig (im Sinne von Aufwand) und kann theoretisch jederzeit erfolgen, sofern es vom PO bzw. Kunden abgenommen wurde.
Bei einem kleinen Team, das zusammen in einem Raum arbeitet, kann getrost auf das *Daily Standup Meeting* verzichtet werden, da man ja sowieso in ständigem Kontakt steht und weiß woran die anderen Teammitglieder arbeiten, bei Problemen kann man sich ad hoc unterstützen. Ebenso sind wahrscheinlich Sprints von zwei Wochen bei einem kleinen Team immer noch sehr lang und können auf eine Woche verkürzt werden.

5.2 Von Kanban zu Scrumban

Kanban eignet sich für eine grosse Bandbreite von Einsatzgebieten und kann mit seinem kleinen Regelwerk flexibel und inkrementell an die jeweiligen Gegebenheiten angepasst werden. So stellt sich die Frage, ab wie viel Anpassung man nicht mehr von Kanban sondern von Scrumban sprechen sollte. So sind in Kanban *Daily Standup Meetings* nicht vorgeschrieben aber durchaus möglich. Wie bei Scrum sind auch regelmäßige Nachschub-Meetings (Queue Replenishment Meeting) möglich, diese könne aber auch nur bei Bedarf einberufen werden, sobald eine bestimmte untere Grenze in der *ToDo*-Spalte unterschritten wird. Eine wichtige und typische Eigenschaft von Kanban ist die Kontinuität der Kanban-Tafel, sie bleibt immer gut gefüllt und es kommt nicht zu übermäßigen Schwankungen in der Arbeitsbelastung. Diese Kontinuität sollte auch bei eine Entwicklung in Richtung Scrumban beibehalten werden, weil die Vorteile doch überwiegen. Ansonsten können viele weiter Scrum-Techniken auch in Kanban integriert werden, je nach den konkreten Erfordernissen.

6 Fazit und Ausblick

Scrum und Kanban wurden in Grundzügen vorgestellt, miteinander verglichen sowie die jeweiligen Vorteile und Nachteile betrachtet. Bei beiden handelt es sich um Werkzeuge (aus einem ganzen Werkzeugkasten der agilen Methoden) mit ihrem jeweiligen Anwendungsgebiet. Jedoch müssen die Wekzeuge geändert werden, wenn sie nicht mehr auf das Anwendungsgebiet passen. Die Werkzeuge können auch kombiniert werden, um zusammen eine bessere Wirkung zu erzielen. Viele Organisationen machen das bereits, bewußt oder unbewußt, und können so ihre Wertschöpfung

steigern. Dabei darf das Vorgehen jedoch nicht in die Beliebigkeit und in Richtung Chaos abgleiten. Scrum und Kanban sind zwei Methoden, die sich sehr gut miteinander kombinieren lassen und so den richtigen Mittelweg zwischen Struktur und Flexibilität bieten können, der heute oft erforderlich ist. Dieser Mittelweg wird auch *Scrumban* genannt, wobei eine genaue Definition nicht gegeben ist. In der Literatur findet Scrumban seit der Veröffentlichung von Corey Ladas[10] immer mehr Beachtung und auch in der Praxis entscheiden sich immer mehr Organisationen bewusst für diesen Weg. Zukünftig bleibt zu beobachten, wie sich Scrumban in der Praxis entwickelt und im Vergleich zu anderen agilen Methoden verbreitet.

[10] [Ladas, C. (2008)]

Literatur

[Geiger, G., Hering, E., Kummer, R. (2011)] *Kanban - optimale Steuerung von Prozessen*, 3. Aufl., Carl Hanser Verlag München, 2011

[Kniberg, H., Skarin, M. (2010)] *Kanban and Scrum - making the most of both*, C4Media, Publisher of InfoQ.com, 2010

[Ladas, C. (2008)] *Scrumban - Essays on Kanban Systems for Lean Software Development* Modus Cooperandi Press, Seattle, 2008

[Leopold, K., Kaltenecker, S. (2013)] *Kanban in der IT: Eine Kultur der kontinuierlichen Verbesserung schaffen*, 2. Aufl., Carl Hanser Verlag München, 2011

[Mahnic, V. (2014)] *Improving Software Development through Combination of Scrum and Kanban* in: Recent Advances in Computer Engineering, Communications and Information Technology, Espanha 2014, S. 281-288

Internetquellen:

[Ken Schwaber und Jeff Sutherland (2013)] *Der Scrum Guide* URL: `http://www.scrumguides.org/docs/scrumguide/v1/Scrum-Guide-DE.pdf`, Abruf am 26.2.2015

[Ward Cunningham et al. (2001)] *Manifest für Agile Softwareentwicklung* URL: `http://www.agilemanifesto.org/iso/de/`, Abruf am 20.2.2015

[Savita Pahuja (2012)] *What is Scrumban?* URL: `http://www.solutionsiq.com/what-is-scrumban/`, Abruf am 15.2.2015

[Innovel, LLC (2008)] *Combining Scrum with Kanban for support and enhancement teams* URL: `http://www.innovel.net/?p=40`, Abruf am 15.2.2015

[Sabine Rossbach (2013)] *10 Erfolgsfaktoren für das Anforderungsmanagement in agilen Software-Großprojekten* URL: `http://blog.doubleslash.de/10-erfolgsfaktoren`, Abruf am 28.2.2015

[Jose Luis Soria Teruel (2011)] *Commitment vs. Forecast: A subtle but important change to Scrum* URL: `https://www.scrum.org/About/All-Articles/articleType/ArticleView/articleId/95/Commitment-vs-Forecast-A-subtle-but-important-change-to-Scrum`, Abruf am 15.2.2015

[Jean Pierre Berchez, Uta Kapp (2010)] *Kriterien für eine Entscheidung für Scrum oder Kanban* URL: `http://www.heise.de/developer/artikel/`

Kriterien-fuer-eine-Entscheidung-fuer-Scrum-oder-Kanban-1071172.html?artikelseite=2, Abruf am 25.2.2015

Ehrenwörtliche Erklärung

Hiermit versichere ich, dass die vorliegende Arbeit von mir selbstständig und ohne unerlaubte Hilfe angefertigt worden ist, insbesondere dass ich alle Stellen, die wörtlich oder annähernd wörtlich aus Veröffentlichungen entnommen sind, durch Zitate als solche gekennzeichnet habe. Weiterhin erkläre ich, dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat. Ich erkläre mich damit einverstanden, dass die Arbeit der Öffentlichkeit zugänglich gemacht wird. Ich erkläre mich damit einverstanden, dass die Digitalversion dieser Arbeit zwecks Plagiatsprüfung auf die Server externer Anbieter hoch geladen werden darf. Die Plagiatsprüfung stellt keine Zurverfügungstellung für die Öffentlichkeit dar.

München, 28. Februar 2015

Oliver Kurmis