

# Kriterien für eine Entscheidung für Scrum oder Kanban

## Brüder im Geiste



Die Väter der agilen Softwareentwicklung hatten erkannt, dass sich Prozesse und Tools den Individuen und deren Interaktionen unterordnen müssen. Solange den Belangen der Projektbeteiligten mit Prozessen wie Kanban oder Scrum gedient ist, ist es für sie "richtig". Beide Frameworks dienen einem agilen Softwareentwicklungsprozess. Der Artikel beleuchtet Kriterien für die Entscheidung für Kanban oder Scrum.

Sowohl Kanban als auch Scrum haben ihre Wurzeln in der Lean Production. Lean-Prinzipien stammen aus der japanischen Automobilproduktion. Ihr Ziel ist es, drei Arten der Verschwendung zu unterbinden, und zwar:

- Muda – Arbeit, die dem Produkt keinen Wert hinzufügt;
- Muri – Überlastung von Mitarbeitern und Maschinen;
- Mura – Unregelmäßigkeit der Prozesse.

Die Prozesse bei Scrum und Kanban unterstützen die Lean-Prinzipien. Jedoch garantieren sie nicht ihre Umsetzung, da nur Disziplin, Kommunikation und hohe Motivation Muda, Muri und Mura vermeiden. Es sind die Individuen, die mit ihrer Persönlichkeit ein Framework füllen und die Prozesse gestalten. Hier liegt das größte Missverständnis beim Einsatz von Hilfsmitteln für die agile Softwareentwicklung.

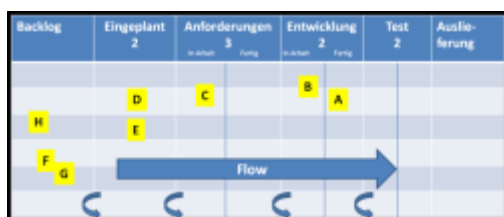
## Kanban – die Wurzeln

Wie erwähnt haben die Lean-Prinzipien ihre Wurzeln in der japanischen Automobilproduktion. Der Ausdruck "Kan" bedeutet Signal und der Ausdruck "Ban" Karte. Das Kompositum Kanban entspricht demnach einer Signalkarte. Sie kommen bei der Lean-Produktion zum Einsatz, um den Materialfluss optimal zu steuern. Ein kleines Beispiel mag das verdeutlichen: Ein Automechaniker ist in der Pkw-Produktion verantwortlich, die Frontscheiben in das Auto einzubauen. Dafür holt er sich immer neue Frontscheiben von einem Stapel. In diesem ist eine Signalkarte platziert, die anzeigt, dass die Scheiben langsam knapp werden und jetzt neue Frontscheiben nachzuliefern sind. Die Karte wandert jetzt zur Produktion der Frontscheiben und signalisiert ihr, dass neue Frontscheiben benötigt werden. Das Vorgehen wurde so entwickelt, weil man möglichst wenig auf Halbe produzieren und den Produktionsfluss weitgehend optimieren wollte.

Immer häufiger hört man in der letzten Zeit in der Softwareentwicklung den Begriff Kanban. Hier ist Kanban ein Projektmanagement-Framework, das sich an den oben genannten Lean-Prinzipien orientiert und wie alle agilen und Lean-Ansätze dem Pull-Prinzip (Hol-Prinzip) zugrunde liegt. Die anfallende Arbeit erteilt nicht ein "Supervisor", sondern die Arbeiter(-gruppen) holen sich ihre Arbeit, um den Produktionsfluss möglichst fließend zu halten.

Das Framework gibt relativ wenig Vorgaben und ist somit stark anpassbar hinsichtlich der Entwicklungsprozesse, Rollen, Abstimmungsmechanismen, Releaseplanungen et cetera. Sie sind nicht reguliert und lassen sich von den Entwicklungsteams nach bestem Wissen und Gewissen für die eigenen Bedürfnisse auswählen. Der zentrale Punkt heißt hier immer wieder Optimierung des Flusses.

## Wie funktioniert Kanban?



Beispiel für ein Kanban-Board

Im Zentrum des Kanban der Softwareentwicklung steht das Kanban-Board. Es visualisiert den aktuellen Stand des Projekts mit Kärtchen für jede Aufgabe oder jeden Work-Item. Auf ihm gibt es mehrere Spalten zur Darstellung des Workflows. Jedes Kärtchen wandert von Spalte zu Spalte, bis die Aufgabe "fertig" ist.

Ziel ist es, die "Aufgaben in Arbeit" (Work-Items in Progress; WIP) zu optimieren. Die Projektbeteiligten messen die Zeit, die eine Aufgabe braucht, bis sie "fertig" ist, identifizieren "Bottlenecks", steuern das Projekt mit der WIP-Begrenzung und passen es an einen möglichst optimierten "Flow" an. Interessanterweise verzichtet das Kanban der Softwareentwicklung auf die Signalkarten, die Begrenzung wird über die Kapazität der Prozess(zwischen)schritte geregelt. Die Kapazitätsbegrenzung wird mit der Angabe der jeweils maximalen zulässigen Work-Items im Kanban-Board dargestellt.

Konkret funktioniert das so, dass der Workflow-Status die Grenzen setzt. Ein Kanban-Board enthält zum einen die Status des gelebten Prozesses und der Begrenzung der Work-Items, die sich zur gleichen Zeit in einem gewissen Status befinden dürfen. Sollten sich nun zu einem Entwicklungszeitpunkt die Karten häufen und der nachfolgende Prozess keinen Raum mehr für Nachschub haben, zeigt sich, dass der "Flow" zu optimieren ist. Entweder indem man mehr Ressourcen für den Prozessschritt, in dem der "Stau" auftaucht, aufbringt oder gegebenenfalls das Entwicklungsteam die Prozessschritte überdenken und ändern lässt. Somit entsteht mit Kanban ein selbstregulierendes System, das den Produktionsfluss sicherstellen soll.

## Was zeichnet Scrum aus?

Bei Scrum bildet man kleine selbst organisierende, bereichsübergreifende Teams. Es bietet einen umfangreichen Satz an Spielregeln, innerhalb derer die Softwareentwicklung

stattfindet. Den Rahmen lebt jedes Team anders aus. Der Kern ist ein fester Zeitabschnitt von maximal vier Wochen, Sprint genannt. Jeder Sprint stellt eine Iteration der Produktentwicklung dar, an deren Ende ein potenziell auslieferbares Produkt "fertig" ist. Jeden Sprint planen die Beteiligten in einem festen Meeting vorab. Das fertige Produkt begutachten sie in einem Review und ziehen daraus Lernerfahrungen. Den Releaseplan optimieren sie aus den Erkenntnissen zusammen mit dem Kunden.

Auch den Prozess, mit all seinen "weichen" Faktoren, reflektieren und optimieren die Teams von Sprint zu Sprint. Dazu ist die Retrospektive vorgesehen, eine Art Meeting. Das Framework verfolgt dadurch ein empirisches Vorgehensmodell. Wie Kanban hat Scrum unterschiedliche Anforderungslisten, das Product- und den Sprint-Backlog, sowie ein Scrum-Board, das durchaus Ähnlichkeit mit dem von Kanban hat.

Welcher Prozess für ein Projekt ideal ist, hängt von den Eigenschaften ab, die es charakterisieren. Wie hoch ist die Komplexität des Projekts? Wie viele Entwickler und Teams arbeiten an dem Produkt? Befindet es sich in der Wartungsphase oder passt man hauptsächlich für Kunden an?

Viele Softwareprodukte sind klein und unkompliziert. Es sind nur drei oder vier Entwickler beteiligt. Für die Projekte kann Kanban viel bewirken. Das Kanban-Board ist der kleinste gemeinsame Nenner, über den sich das Team synchronisiert. Wenn alle im gleichen Büro sitzen, ist es nicht notwendig, eine fest vorgegebene Besprechungsabfolge einzuhalten. Man spricht sowieso miteinander. Für zusätzliche Prozessabläufe, wie Iterationen mit fest definierten Zeitboxen, kann sich jedes Team dann gemeinsam entscheiden.

Wartungsprojekte haben eine Phase erreicht, in der das Produkt nur noch bei Bedarf weiterentwickelt wird. Es ist meist nicht möglich, einen kontinuierlichen Fluss an Tasks zu erzeugen. Die Mitarbeiter arbeiten vielleicht nur teilweise an solch einem Projekt. Kanban ist in dem Fall eine gute Lösung. Geschwindigkeit spielt bei der Konstellation normalerweise keine Rolle. Scrum erscheint für die Projekte ein überdimensionierter Prozess zu sein. Das erzeugt Muri und das ist dann wie das berühmte "mit Kanonen auf Spatzen schießen". Der Prozess selbst wäre eine Verschwendung.

## **Hohe Komplexität mit Scrum bewältigen**

Für Projekte mit hoher Komplexität, bei denen mehr als vier Softwareentwickler beteiligt sind, eignet sich Scrum besonders gut. Das liegt daran, dass Scrum einen ganzen Satz an "Ritualen" definiert, der den Prozess strukturiert. Es ist einfacher, alle Teammitglieder davon zu überzeugen, regelmäßig Meetings abzuhalten und sich zu synchronisieren, wenn für sie ein festgelegter Satz Spielregeln im Raum steht. Softwareingenieure empfinden Meetings zu Recht oft als Zeitverschwendung. Die Treffen in Scrum sind jedoch so bemessen, das sie mit Zeitlimit versehen sind und deswegen nicht ausufern. Auch ist klar definiert, welche Aufgaben in welcher Sitzung zu bewältigen sind. Das schafft Struktur, und damit kann normalerweise jeder leben. Vor allem die kurzen täglichen Scrum-Meetings sorgen für regelmäßigen Informationsaustausch. Das fördert die Teamzusammengehörigkeit. So verwundert nicht, dass sich Teams auf solche regelmäßigen Treffen und starren Zeitfenster einlassen, wenn es heißt: "wir machen Scrum".

Besonders für die verteilte Produktentwicklung eignet sich Scrum. Bei ihr gibt es täglich ein "Scrum of Scrums", in denen sich die Teams untereinander synchronisieren. Es gibt ein gemeinsames Product-Backlog und ein gemeinsames Produkt. Die Verteilung von Teams optimiert die Skalierbarkeit von großen Projekten.

## Fazit

Scrum und Kanban eignen sich gut für Projekte mit unterschiedlicher Komplexität und Anzahl beteiligter Personen. Hinzu kommen der Reifegrad eines Teams und die besonderen Vorlieben aller Beteiligten. Kanban-Fans möchten sich nicht gerne auf ein so umfangreiches Regelwerk wie Scrum einlassen und mehr selbst definieren. Für eingefleischte Scrum-Anhänger sind genau diese festen Rituale das, was Scrum ausmachen. Je mehr Personen und Teams an einem Projekt beteiligt sind, desto mehr verhelfen feste Spielregeln zu einem geordneten Prozessablauf. ([ane](#))

### *Uta Kapp*

*ist freiberufliche IT-Beraterin und systemischer Coach. Mit einer Kombination aus Fach- und Prozessberatung für Softwareprojekte hilft sie Entwicklungsteams bei der Bewältigung der ständig steigenden Komplexität. Hier kommen agile Softwareentwicklungsmethoden wie Scrum und Kanban zum Einsatz.*

### *Jean Pierre Berchez*

*ist Geschäftsführer der [HLSC UG \(http://www.scrum-events.de\)](http://www.scrum-events.de) und beschäftigt sich seit mehr als 15 Jahren mit den Themen Projektmanagement, Software Engineering und objektorientierte Softwareentwicklung. In den letzten Jahren liegt sein Interesse auf den Themengebieten agile Entwicklung mit Schwerpunkt Scrum sowie "Collaborative" Software Development. Er organisiert unter anderem den Community-Event "[Scrum-Day](#)" in Deutschland.*

## Literatur

1. Henrik Kniberg, Mattias Skarin; [Kanban and Scrum – making the most of both](#); C4Media Inc., 2010
2. Marion Eickmann; Geregelt Selbstbestimmung; Ein Plädoyer für agile Softwareentwicklung mit Scrum; iX Special "Programmieren heute" 1/2010, S. 82
3. Markus Andrezak, Arne Roock, Henning Wolf; Gedämpfte Schritte; Evolutionäre Entwicklung mit Software-Kanban; iX 6/2010, S. 108
4. Bernd Oestereich: Gedanken zur Sprache in Scrum, [Blog-Eintrag](#) auf *heise Developer*

Weitere Artikel zum Thema Scrum und Softwareentwicklung finden Sie auf:

<http://www.scrum-events.de>