mehr zum thema:
www.limitedwipsociety.org/
www.agilemanagement.net

„SCRUM FUNKTIONIERT NICHT ÜBERALL"

Interview mit David Anderson über Kanban, den Erfolg und Misserfolg von Veränderungsprozessen, über den Sinn und Unsinn von Aufwandsschätzungen und über Retrospektiven.

Hallo David – schön, Sie hier in Deutschland zu haben. Erzählen Sie uns, wie Sie zu Kanban gekommen sind und womit Sie sich im Moment beschäftigen?

Das ist eine lange Geschichte. Die Evolution von Kanban hat insgesamt zehn Jahre gedauert. Zuerst haben wir mit der agilen Methode *Feature Driven Development (FDD)* gearbeitet, obwohl wir damals noch gar nicht genau wussten, was agile Methoden sind. Unsere Erfahrungen haben wir durch Ideen aus dem Lean Management und einige Ansätze aus der *Theory of Constraints* erweitert. Diese Ideen habe ich in meinem Buch von 2003 veröffentlicht ([And03]). Danach habe ich weiter hinzugelernt. Ich hatte stets mit einem großen Problem zu kämpfen: Ich habe FDD zweimal sehr erfolgreich eingeführt. Nach diesem Erfolg wollte ich FDD hoch skalieren auf eine ganze Business-Unit. Beide Male, bei denen ich das versuchte, gab es große Widerstände und ich konnte es nicht institutionalisieren. Da habe ich angefangen, darüber nachzudenken, warum Leute nicht die offensichtlichen Vorteile darin sehen, die richtigen Dinge zu tun. Warum haben sie sich gewehrt? Und warum mussten wir so viel kämpfen? Ich bin zu folgendem Schluss gekommen: Wenn du mit einer vorgefertigten Definition eines Prozesses ankommst und den Leuten erzählst, dass sie diesen Prozess nun anwenden sollen und dass sie zum Beispiel ihre Jobbezeichnung ändern müssen oder die Art zu arbeiten, dann wehren sie sich. Deshalb ist es besser, evolutionär statt revolutionär vorzugehen. Durch eine ganze Reihe von Ereignissen bin ich dann zu dem Schluss gelangt, dass ein Kanban-System eine Möglichkeit darstellt, evolutionär vorzugehen, anstatt eine Methode einfach vorzuschreiben. Und das hat sich bewahrheitet. Zwischen 2004 und 2007 habe ich das mit verschiedenen Teams an unterschiedlichen Orten ausprobiert, bis ich sicher war, dass es tatsächlich funktioniert. Ich habe dann öffentlich über meine Erfahrungen gesprochen und andere haben es ebenfalls ausprobiert. Jetzt, zwei Jahre später, gibt es eine ganze Reihe von Teams auf der ganzen Welt, die Kanban anwenden, um Änderungen evolutionär zu bewirken, anstatt revolutionär, so wie es die klassischen agilen Methoden tun.

> „Es ist besser, evolutionär statt revolutionär vorzugehen."

Können Sie versuchen, Kanban in zwei Sätzen zu erklären?

Hinter Kanban steht die sehr einfache Idee, dass man die Menge an paralleler Arbeit (*Work-In-Progress*) begrenzt, dass man diese begrenzte Arbeit visualisiert, indem man z. B. ein Whiteboard mit Haftnotizen darauf verwendet, und dass man nur dann mit neuen Aufgaben beginnt, wenn man vorher bestehende Aufgaben erledigt hat. Darüber hinaus gibt es ein Signalsystem, das deutlich macht, wann neue Arbeit gezogen werden darf.

Im Moment reden alle über Scrum. Weltweit gibt es über 61.000 zertifizierte Scrum-Master. Beinahe jedes Unternehmen hat bereits Scrum eingeführt oder plant die Einführung. Warum brauchen wir da noch Kanban?

Man kann natürlich nicht abstreiten, dass Scrum außergewöhnlich erfolgreich ist. Ich bin mir ganz sicher, dass Scrum ein Rezept ist, das unter bestimmen Umständen gut funktioniert. Der Kern der Scrum-Philosophie besagt, dass Scrum funktioniert und dass man die Umstände so ändern muss, dass sie zu Scrum passen. Scrum gibt einige gute Regeln vor, die dabei helfen, erfolgreich zu sein. Es gibt Sprints von begrenzter Länge – ursprünglich waren es vier Wochen. Zu Beginn eines Sprints treffen wir dann eine Vereinbarung über den Umfang, füllen also das Backlog für den Sprint. Dann wird vier Wochen lang gearbeitet, und der Kunde darf während dieser vier Wochen nicht eingreifen. Das ist ein fantastischer Weg, um Beliebigkeit und Varianzen zu eliminieren. Zusätzlich gibt es Mechanismen wie das tägliche Standup-Meeting, in dem das Team zusammen-

DAVID J. ANDERSON

David J. Anderson gilt als Vater des Kanban-Ansatzes in der Softwareentwicklung. Zusammen mit Jeff De Luca hat er in den 90er Jahren die agile Methode Feature Driven Development entwickelt. Sein Fokus liegt heute darauf, Agilität auch in großen Unternehmen einzuführen, indem er das CMMI-Modell für organisatorische Reife mit agilen Ansätzen und Lean verbindet. David Anderson leitet eine Management-Beratung in Seattle.

Das ungekürzte Interview können Sie im Software Engineering Radio unter www.se-radio.net *hören.*

Software Engineering Radio sucht neue Mitstreiter.

Der SE Radio Podcast ist auf der Suche nach neuen Teammitgliedern. Deren Aufgabe ist die Ideenfindung, Planen, Durchführung und das Schneiden von englischsprachigen Interviews mit interessanten Menschen aus der Softwarebranche. Bei Interesse nehmen Sie bitte Kontakt auf mit Markus Völter unter voelter@acm.org.

Abb. 1: Kanban-Boards visualisieren den Workflow und machen Flaschenhälse und andere Probleme schnell sichtbar.

kommt und über seine Arbeit spricht. Dadurch entsteht ein guter Teamgeist und vielleicht auch ein leichter Druck, dass jeder Einzelne Verantwortung übernimmt. Das Team verpflichtet sich zu dem Sprint-Ziel und dann arbeiten alle zusammen, um bis zum Ende des Sprints fertig zu werden. Danach wird dem Kunden ein funktionierendes Produkt-Inkrement vorgestellt, und es gibt eine Retrospektive. Dies alles ist sehr gut! Die Herausforderung dabei liegt im Rhythmus: Vielleicht passen vier Wochen nicht und zwei Wochen sind besser?

„Die Herausforderung liegt im Rhythmus."

Ich glaube nicht daran, dass Scrum in jeder Situation funktioniert. Viele Leute besuchen Kanban-Schulungen, weil sie in Umgebungen arbeiten, für die Scrum nicht besonders gut geeignet ist, z. B. Systemadministration und Betrieb. Die Vorstellung, Arbeit für zwei oder vier Wochen in ein Backlog zu füllen, ist völlig befremdlich für sie. Da, wo Scrum anwendbar ist, ist es sehr erfolgreich, weil es klare Grenzen setzt und weil die Regeln präzise sind: Unterbrechungen werden verhindert, Menschen können ungestört ihre Arbeit tun und es wird ihnen erlaubt, sich innerhalb der Timebox selbst zu organisieren. Diese Regeln sind gut gewählt und erfolgversprechend. Und sie sind einfach und leicht zu verstehen. Wenn man sie in zwei Tagen Schulung lernen kann, dann können sie nicht so schwierig sein. Scrum ist eine große Verbesserung gegenüber anderen Ansätzen im Projektmanagement. Aber Scrum kann nicht jedes Problem lösen. Die Leute immer schärfer zu kritisieren und ihnen vorzuwerfen, dass sie Scrum nicht korrekt anwenden, ist nicht die richtige Antwort. Ich denke, man sollte aufgeschlossen gegenüber anderen Ansätzen zu sein.

Sind Scrum und Kanban konkurrierende Methoden oder ergänzen sie sich gegenseitig?

Ich sehe sie nicht als konkurrierende Methoden. Denn Kanban ist gar keine Entwicklungs- oder Projektmanagement-Methode. Im Moment kann man beobachten, wie die Scrum-Community eine Neudefinition von Scrum vornimmt. Und dabei gibt es konkurrierende Neu-Definitionen. Wenn man die frühen Bücher über Scrum liest, ist es im Wesentlichen eine Projektmanagement-Methode. Tatsächlich finden sich dort keine Entwicklungspraktiken. Vielmehr sagen viele Scrum-Leute: Man braucht zusätzlich zu Scrum XP-Techniken, damit man Entwicklung und Projektmanagement zusammen hat. Kanban ist keines von beiden: weder eine Entwicklungs-, noch eine Projektmanagement-Methode. Kanban ist eine Änderungsmanagement-Methode, eine Art, um Änderungen herbeizuführen, die zu einem schlanken Unternehmen führen, zu kontinuierlichen Verbesserungen, zu einer Kaizen-Kultur. Es ist ein Weg, um inkrementelle, evolutionäre Änderungen in einer Organisation herbeizuführen, anstatt die Art zu arbeiten zu revolutionieren.

Kanban verwendet man nicht allein. Häufig bekomme ich die Frage gestellt: „Wir stellen ein neues Team zusammen und haben uns noch nicht für eine Methode entschieden. Sollen wir Scrum oder Kanban verwenden?" Meiner Meinung nach ergibt diese Frage überhaupt keinen Sinn, denn man kann Kanban nur auf etwas bereits Bestehendes aufsetzen. Das erste Team, mit dem wir Kanban angewendet haben, war ein PSP/TSP-Team, das als ausgelagerter Lieferant in Indien für Microsoft gearbeitet hat. Dort haben wir Kanban aufgesetzt, ohne an diesem Prozess etwas zu ändern. Wir konnten die Produktivität verdreifachen und die Durchlaufzeit um 90 % reduzieren. Das nächste Team, mit dem wir Kanban ausprobiert haben, war ein gutes altes Wasserfall-Team, es hat also mit einer Methode gearbeitet, die aus den 70er Jahren stammt. Seitdem habe ich viele Teams gesehen, die in unterschiedlichen Situationen Kanban eingeführt haben. Und es gibt Teams, die Kanban auf Scrum aufgesetzt haben. Das Problem dabei ist: Wenn man Scrum um Kanban erweitert, verändert sich Scrum sehr schnell. Das Team beginnt Fragen zu stellen: „Warum machen wir zweiwöchige Iterationen?" Kanban hilft ihnen, Dinge zu entkoppeln: Der Input, also die Priorisierung und die Planung, ist unabhängig vom Output, also dem Release-Mechanismus. Ebenso die Durchlaufzeit. In Scrum sind diese drei Dinge gekoppelt. Wenn man einen Zwei-Wochen-Sprint hat, dann plant man auch alle zwei Wochen. Die Durchlaufzeit beträgt zwei Wochen und der Release-Zyklus ebenfalls. Für einige Organisationen ist das aber nicht die beste Wahl. Wenn man Kanban einführt, fragt man sich sehr bald, was der richtige Rhythmus ist, und man entkoppelt Input, Durchlaufzeit und Output. Wenn man aber aufhört, gleich lange Sprints durchzuführen, egal ob zwei oder vier Wochen, ist es dann noch Scrum?

Das ist eine große Herausforderung, denn die Scrum-Community hat hierfür Standards festgelegt. Der Nokia-Test beispielsweise überprüft, wie sehr man Scrum einsetzt. Wenn man diesen Test nicht besteht, bedeutet das dann, dass man kein Scrum einsetzt? Ursprünglich dachte ich, man führt Scrum ein und wird dadurch ermutigt, sich immer weiter zu verbessern.

Und wenn man sich verbessert, sollte man sich ändern. Ein gutes Scrum-Team sollte nach einem Jahr etwas tun, was einzigartig auf der Welt ist, aber nicht mehr unbedingt wie Scrum aussehen muss. Man kann sehen, wie die Scrum-Community mit dem Widerspruch kämpft: Wie kann man sich immer weiter verbessern, wenn man beweisen muss, dass man Scrum macht, so wie es ein Test vorschreibt? Es gibt also Teams, die Kanban auf Scrum aufgesetzt haben. Tatsächlich hat mein Freund Corey Ladas ein ganzes Buch über das Thema geschrieben ([Lad09]).

Ebenso wie Scrum ist Kanban eher ein Management-Framework, das keine Entwicklungstechniken vorgibt. Gibt es bestimmte Entwicklungstechniken, die Sie für Kanban-Systeme empfehlen?

Wenn man Kanban verwendet, um Software zu entwickeln, benötigt man die Fähigkeit, Releases zu erstellen. Das bedeutet, dass man sein Konfigurationsmanagement beherrschen muss. Man muss Anforderungen identifizieren und tracken können. Man muss die Versionskontrolle im Griff haben und verschiedene Branches managen können. Man muss Branches vernünftig in Einklang bringen können und man muss in der Lage sein, Releases so zu organisieren, dass wirklich nur die gewünschten Features ins nächste Release kommen und nicht aus Versehen auch die unfertigen Features. Konfigurationsmanagement ist also eine Kernfähigkeit, die man braucht, damit Kanban funktioniert. Man muss aber kein Meister auf diesem Gebiet sein. Ich glaube, dass die meisten agilen Teams, die XP einsetzen, diese Fähigkeit bereits besitzen. Sie wissen, wie man User-Storys trackt, wie man Versionskontrolle handhabt, wie man mit Branches umgeht, wie man kontinuierliche Integration einsetzt und wie man lauffähige Releases erstellt. Das sind also notwendige Fähigkeiten, aber sie sind nicht ungewöhnlich. Ich habe Leute sagen hören: „Wir haben ein Team, das sehr schlecht in der Softwareentwicklung war. Dann haben wir Kanban eingeführt und das Team war immer noch schlecht." Als ich fragte, warum sie so schlecht waren, kam als Antwort: „Sie konnten keine vernünftigen Releases erstellen." „Und warum nicht?" „Weil sie nicht in der Lage waren, die Builds zum Laufen zu bringen." „Und warum konnten sie die Builds nicht zum Laufen bringen?" „Weil sie kein funktionierendes Konfigurationsmanagement hatten." Das sind also die Kernfähigkeiten, die nötig sind, damit Kanban funktioniert.

Viele sagen, dass Retrospektiven das Herzstück agiler Vorgehensweisen darstellen. Gibt es auch in Kanban Retrospektiven?

Man kann nichts lernen, solange man sich nicht die Zeit nimmt, um anzuhalten und zu reflektieren. In Kanban ermutige ich die Teams dazu, auf einer mehr organisatorischen Ebene zu reflektieren. Ich nenne das *Operations Reviews*. Das ist nicht neu – ich habe dieses Konzept nicht erfunden. Worin sich *Operations Reviews* und typische agile Retrospektiven unterscheiden, ist der Grad der Objektivität. Für ein *Operations Review* benötigt man in der Regel Manager, die Daten mitbringen: Daten, die sich auf den Workflow beziehen, die Fehlerraten, die Durchlaufzeiten usw. Diese Daten werden dann bei den *Operations Reviews* vorgestellt. Hier sieht man sich beispielsweise „Cumulative Flow Diagrams" an oder die Anzahl geblockter Tasks. Dies sind alles harte Fakten, objektive Daten. Agile Retrospektiven sind viel subjektiver. Wie ging es uns beim letzten Sprint? Was gefiel uns? Was gefiel uns nicht? Was wollen wir wieder so machen? Was wollen wir anders machen? Das sind typische Fragen. Bei erfahreneren Teams werden auch die Retrospektiven immer objektiver. Aber mit Kanban-Teams versuche ich, diese Objektivität von Anfang an zu etablieren. Und hier sind wir auch wieder beim Entkopplungsthema: Agile Methoden koppeln Retrospektiven an Releases. Wenn man also zweiwöchige Iterationen hat, führt man auch alle zwei Wochen eine Retrospektive durch. Das kann in Ordnung sein. Aber was ist, wenn man sich nur einmal im Monat zusammensetzen möchte? Oder jede Woche? Oder in irgendeinem anderen Rhythmus? Ich arbeite darauf hin, dass die Teams ihren eigenen Rhythmus für ihre *Operations Reviews* suchen, dass dieser Rhythmus entkoppelt ist vom Release- und Planungsrhythmus und dass diese Meetings auf einer organisatorischen Ebene stattfinden. Es geht nicht um die Ebene einzelner Projekte. Holt die gesamte Abteilung zusammen, alle sollen gemeinsam an den Reports arbeiten! Der Grund, warum das gut ist, ist, dass sich unterschiedliche Projekte so gegenseitig befruchten können. So werden organisatorische Entwicklung und organisatorische Reife gefördert. Ich glaube, das ist der Grund, warum man bei vielen Kanban-Teams beschleunigte Entwicklung feststellen kann. Aber ich habe auch schon Kanban-Teams gesehen, die Retrospektiven auf Projekt- oder Iterationsebene durchführen. Wenn sie ein Release hinter sich haben, in dem etwas schief ging, analysieren sie, was da schief gelaufen ist und wie sie das in Zukunft verhindern können.

Im Internet schildern Leute, dass ihr Team aufgehört hat, Retrospektiven durchzuführen. Sie machen zwar noch *Operations Reviews* aber keine Retrospektiven mehr. Wenn man dann analysiert, warum das so ist, findet man heraus, dass sich viele Aktivitäten aus der Retrospektive in das tägliche Standup-Meeting verlagert haben. Weil sie jeden Tag über Prozessverbesserung reden, brauchen sie es nicht mehr am Ende jeder Iteration zu tun. Man macht es täglich und nicht alle zwei Wochen oder einmal im Monat. Sehr gute Kanban-Teams, die eine wirkliche Kaizen-Kultur entwickelt haben, also einen kontinuierlichen Verbesserungsprozess, führen jeden Tag Retrospektiven durch. Der Wert von Retrospektiven wird in Kanban also reduziert, weil sie ersetzt werden durch eine Kombination aus täglichen Standup-Meetings und *Operations Reviews*.

Ein anderer interessanter Punkt bezieht sich auf die Schätzungen. Stimmt es, dass in Kanban komplett auf Schätzungen verzichtet wird?

Das ist eine gute Frage, denn in den ersten Beispielen haben wir tatsächlich nicht geschätzt. Es hat sich herausgestellt, dass Schätzungen optional sind. Wenn du schätzen möchtest, dann tu es! Oft schätzen Leute, um ein Lieferdatum zu versprechen, auf das sich dann Leute verlassen. Aber nicht jede Tätigkeit, die man verrichtet, braucht ein hartes Lieferdatum. Immer wieder höre ich: „Wenn keine Verpflichtungen bezüglich des Lieferdatums eingegangen werden, kann man auch niemanden verantwortlich machen." Meine Einstellung dazu ist: Man setzt ein künstliches Ziel und zwingt das Team dann, es auch zu erreichen. Das führt häufig zu

„Sehr gute Kanban-Teams führen täglich Retrospektiven durch."

Heldentum. In der Prozessverbesserung und in der Qualitätssicherung gibt es viele Hinweise auf diese Konstellation: Wenn man ein Ziel setzt und erwartet, dass dieses auch erreicht wird, reagieren die Leute darauf mit Heldentum. Aber wer sich heldenhaft benimmt, hört auf, sich zu verbessern. Deshalb lautet ein Ratschlag von Goldratt (vgl. [Gol02]) und einer ganzen Reihe anderer moderner Vordenker, dass man auf solche Ziele verzichten soll. Ich habe den Eindruck, dass Schätzungen zu Deadlines führen und zu Verpflichtungen, die dysfunktional und nicht erstrebenswert sind.

„Wer sich heldenhaft benimmt, der hört auf, sich zu verbessern."

Auf der anderen Seite gibt es Situationen, in denen man einen bestimmten Liefertermin definitiv einhalten muss. Zum Beispiel muss eine Investment-Bank eine neue Anforderung umsetzen, weil die Regierung einige Regeln geändert hat und diese neuen Regeln zum 1. Oktober in Kraft treten. Wenn die Bank die neuen Regeln bis zu diesem Termin nicht einhält, darf sie bestimmte Wertpapiere nicht mehr handeln. Das sind sehr hohe Kosten, die da drohen: Wenn man die Anforderungen nicht erfüllt und das Lieferdatum nicht erreicht, darf man keine Wertpapiere mehr handeln. Dann ist ganz klar, dass man sicherstellen muss, das neue Feature pünktlich auszuliefern. Für diese spezielle Anforderung ist es sinnvoll, zu schätzen, wie lange die Umsetzung dauern wird. Wenn wir schätzen, dass die Entwicklung sechs Wochen dauert und wir die neue Funktionalität bis zum 1. Oktober benötigen, ist es sinnvoll, Anfang August mit der Entwicklung zu beginnen. Es ist nicht sinnvoll, schon im Mai zu beginnen, denn ein früher Start schafft hier absolut keinen Wert. Aber wenn man zu spät ist und erst im November liefert, fährt man große Verluste ein. Kanban sagt, dass man Schätzungen angemessen nutzen sollte. Man muss sich Gedanken darüber machen, wann Schätzungen sinnvoll sind, und diese dann intelligent nutzen. Schätzungen sind also optional. Anstatt alles zu schätzen, schätzen wir nur die wirklich wichtigen Dinge

Die Grundidee zu Kanban stammt aus der Automobilproduktion, wo wir es mit sich stets wiederholenden, gleichförmigen Abläufen zu tun haben. Softwareprojekte hingegen haben ja häufig eher den Charakter einer Produktentwicklung, also eines sehr kreativen und individuellen Prozesses. Ist Kanban überhaupt für Softwareprojekte geeignet oder liegen die wahren Stärken von Kanban in den Bereichen Betrieb und Systemadministration?

Meistens höre ich diese Frage von Leuten aus der Produktion, die skeptisch sind. Kanban ist jedoch ganz eindeutig für Softwareprojekte geeignet! Dort sehen wir viele Leute, die Kanban erfolgreich einsetzen. Aber es taucht eine Reihe interessanter Fragen auf: Warum funktioniert es, da doch die Softwareentwicklung viel mehr Variabilität aufweist als die Produktion? Das führt zu der Frage, wie Kanban an einen anderen Kontext angepasst wurde, nämlich an die Softwareentwicklung. Software-Kanban hat zwar mit denselben Prinzipien wie die Produktion angefangen, aber daraus sind unterschiedliche Praktiken entstanden. Wir haben nicht versucht, die Praktiken aus der Produktion *1:1* zu kopieren. In Wahrheit haben wir sie gar nicht kopiert, sondern wir haben uns die dahinter liegenden Prinzipien angesehen und diese auf die Softwareentwicklung übertragen, um so unsere eigenen Praktiken zu entwickeln. Deshalb kann man nicht wirklich sagen, dass Software-Kanban aus der Produktion entstanden ist.

Mein Ratschlag an alle, die an Kanban interessiert sind, lautet: Probiert es aus! Viele Aspekte von Kanban sind zunächst kontraintuitiv. Der einzige Weg, mehr darüber herauszufinden, ist es auszuprobieren!

Vielen Dank für das Gespräch, David.

Das Gespräch führte Arne Roock.

Literatur

[And03] D. Anderson, Agile Management for Software Engineering – Applying the Theory of Constraints for Business Results, Prentice Hall 2003

[Gol02] E.M. Goldratt, P. Pyka, Die kritische Kette. Das neue Konzept im Projektmanagement, Campus Verlag 2002

[Lad09] C. Ladas, Scrumban – Essays on Kanban Systems for Lean Software Development, Bertrams Print on Demand 2009