mehr zum thema:
it-agile.de
software-kanban.de

KANBAN FÜR TESTER: EVOLUTIONÄRES QUALITÄTSMANAGEMENT

Kanban ist derzeit in aller Munde. Zumeist liegt jedoch der Fokus auf den Besonderheiten für die Arbeitsorganisation. In diesem Artikel begleiten wir den Tester Tobias, der gerade einen Job in einer Firma beginnt, die nach Kanban arbeitet. Dabei entdeckt er viele Dinge, die in seiner vorherigen Firma nicht funktionieren wollten.

die autoren

Markus Gärtner
(markus.gaertner@it-agile.de)
arbeitet als Berater, Coach und Tester für it-agile GmbH in Hamburg. Er ist Mitgründer des Europäischen Weekend Testings sowie Anhänger der ATDD-Pattern- und der Software-Craftsmanship-Bewegung.

Dr. Arne Roock
(arne.roock@it-agile.de)
arbeitet als Trainer und Coach bei it-agile in Hamburg. Sein besonderes Interesse gilt dem evolutionären Change-Management mit Kanban. Er ist Verfasser zahlreicher Fachartikel und Übersetzer des Standardwerks zu Kanban von David Andersson.

„Wird schon schief gehen“, denkt sich Tobias, als er die Klinke zu Flow Work Ltd. in die Hand nimmt und seinen Fuß als Tester in die Firma setzt. Beim Bewerbungsgespräch hatte er anscheinend einen guten Eindruck gemacht. Bei seinem vorherigen Arbeitgeber, der Big Enterprise AG, hatte er zuletzt seine Sachen gepackt. Immerzu traten dort die gleichen Probleme auf und nichts änderte sich.

In die Enge getrieben zwischen Entwicklern, Test- sowie Projektmanagern, hatte er sich zuletzt mehr als unwohl gefühlt. In den kurzen Auslieferungszyklen hatte er anfangs Probleme, mit den Programmierern mitzuhalten. Der Projektleiter hatte Tobias dann gefragt, ob er nicht hinter den Programmierern hinterher arbeiten könne. Natürlich führte das nur zu noch mehr Problemen, wenn Tobias dann Fehler gefunden hatte. Deshalb war er als Tester bei den Programmierern unbeliebt. Außerdem war das Zeitmanagement mit drei Programmierteams gleichzeitig alles andere als entspannt. Irgendwie hatte er auch immer das Gefühl, dass das eigentlich anders sein sollte, aber das wollte bei Big Enterprise niemand hören. Im vergangenen Herbst hatte er dann den Schritt gewagt und sich nach dem Motto „Love it, Change it, or Leave it” einen neuen Arbeitgeber gesucht.

Bisher hatte er die Erfahrung gemacht, dass Entwickler die Tester mit Arbeit überhäufen. Dabei konnte die Software oftmals gar nicht erst installiert, geschweige denn gestartet werden. Hinzu kam der Testmanager, der fast im Minutentakt nach der Anzahl durchgeführter Tests, den Testplänen und Testprotokollen fragte. Und in Krisenzeiten stand dann auch noch der Projektmanager bei Tobias am Schreibtisch und wollte ihn zur Onsite-Unterstützung nach Bangalore oder Malaysia schicken. Insgeheim hoffte Tobias, dass sich durch den Unternehmenswechsel etwas zum Positiven für ihn ändern würde.

Der erste Tag

„Willkommen Tobias“, begrüßt ihn sein neuer Chef Martin fröhlich, als dieser ihn am Empfang abholt. „Wir duzen uns alle hier. Ich bin Martin. Schön dich dabei zu haben. Ich zeige dir erst einmal deinen neuen Arbeitsplatz. Doch bevor du dich einrichtest, solltest du in zehn Minuten an unserem Stand-up teilnehmen.“

„Stand-up? Aber hast du nicht gesagt, ihr würdet Kanwan, oder wie das heißt, einsetzen?“

„Kanban, richtig. Wir stimmen uns täglich über die Arbeit des Vortags ab. Jeden Morgen treffen wir uns an unserem Kanban-Board und diskutieren die dringlichsten Aufgaben und wie wir wichtige Aufgaben voran bekommen können“, antwortet Martin.

Tobias fragt nach: „Kanban?“

„Erkläre ich dir später, lass uns erst einmal zum Team beim Stand-up dazu stoßen.“

In Tobias vorheriger Firma hatte es einen Versuch gegeben, eine teambasierte Organisationsstruktur einzuführen. Die Unternehmensführung hatte über die Erfolge mit Teams gelesen – und schließlich setzte der Vorstand die unternehmensweite Einführung einer neuen Methode durch. Hierzu wurde ein externer Berater in das Unternehmen geholt, der alle Mitarbeiter schulte. Neue Teams wurden gegründet und die komplette Unternehmensstruktur wurde über den Haufen geworfen.

Die Teams begannen nach und nach mit der Umstellung auf die neue Entwicklungsmethode. Die Teammitglieder konnten sich aber nur schwer an die neuen Teams und Rollen gewöhnen. Besonders für die Tester war dies schwierig, da sie auf einmal aus ihren bisherigen Teams herausgelöst werden sollten, um in so genannten funktionsübergreifenden Teams als Test-Einzelkämpfer zu arbeiten. Alles in allem hatte Tobias eine eher schlechte Erinnerung an diese drastische Umstellung, da es für ihn besonders schwierig war, Akzeptanz in seinem Team zu finden. Die Features wurden zwar alle innerhalb kürzester Zeit implementiert, aber für die Testaktivitäten stand kurz vor der Auslieferung stets zu wenig Zeit zur Verfügung. Außerdem war es für ihn als Tester einfach unmöglich, ohne Anforderungsdokumente seine Testfälle zu erstellen, bevor er diese ausführen konnte. Und das alles nur, weil der oberste Boss seinen Bonus durch die Einführung aufstocken wollte.

Martin und Tobias treffen den Rest des Teams vor einer riesigen Tafel mit Karten. „Das hier ist unser Kanban-Board”, erklärt Martin. Martin stellt Tobias jedem einzelnen der 25 Teammitglieder vor. Tobias hat zunächst Probleme damit, sich so viele Namen und Gesichter auf einmal zu merken.

Das Kanban-Board ähnelt einem Taskboard eines Scrum-Teams, das Tobias

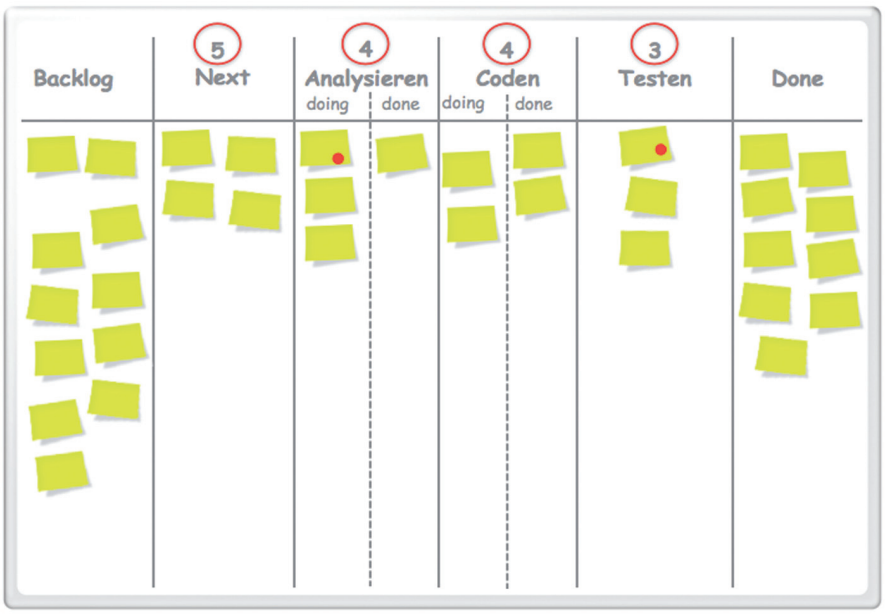

Abb. 1: Das Kanban-Board bei Flow Work Ltd.

mal im Internet auf der Suche nach besseren Ansätzen zur Softwareentwicklung gesehen hat. Nur gibt es nicht drei Spalten („ToDo“, „Doing“ und „Done“), sondern gleich acht. Ganz links befindet sich das „Backlog“. Tobias vermutet, dass sich hier anfallende Arbeit aufstaut. Ganz rechts gibt es die bekannte Spalte „Done”. Aber die „Doing“-Spalte ist jetzt aufgefächert in die Aktivitäten „Analysieren”, „Coden” und „Testen”. Außerdem gibt es nach dem Backlog noch die Spalte „Next” (siehe Abbildung 1).

Tobias erinnert sich an das, was er über Stand-up-Meetings in Scrum gelesen hat. Das Team kommt hier einmal täglich zusammen und beantwortet drei Fragen im Stehen:

- Was habe ich seit dem letzten Mal getan?
- Was mache ich bis zum nächsten Mal?
- Was behindert mich, sodass ich dabei Unterstützung benötige?

Diese Meetings findet Tobias sinnvoll, weil es ein einfacher Mechanismus ist, mit dem das Team seine tägliche Arbeit koordinieren kann. Bei der Firma Big Enterprise hatte er diese Meetings in einem Team ebenfalls eingeführt. Allerdings hatten die Stand-ups dort die Tendenz, in ausführliche Diskussionen auszuufern. Tobias kann sich noch gut daran erinnern, wie er sich einmal nach einer Stunde einen Stuhl geholt hat, weil er nicht mehr stehen konnte.

Damals waren sie nur 10 Teammitglieder gewesen. Jetzt stehen 25 Personen vor dem Board und Tobias befürchtet das Schlimmste. Aber bei Flow Work Ltd. läuft das Stand-up anders. Anstatt dass jeder Teilnehmer die drei Fragen beantwortet, schauen alle auf das Kanban-Board und gehen Karte für Karte durch, um zu besprechen, was getan werden muss, um die Karte weiter nach rechts zu bekommen. Einige Karten sind mit einem roten Sticker als Blocker markiert. Diesen Blockaden wird besondere Aufmerksamkeit geschenkt und das Team überlegt gemeinsam, wie der Fluss auch bei diesen Karten wieder hergestellt werden kann und welche Maßnahmen ergriffen werden sollen, um die tiefer liegenden Gründe für diese Blockaden zu beseitigen. Auf allen Karten ist mindestens ein Datum vermerkt (siehe Abbildung 2) – auf den Karten in der letzten Spalte sogar zwei.

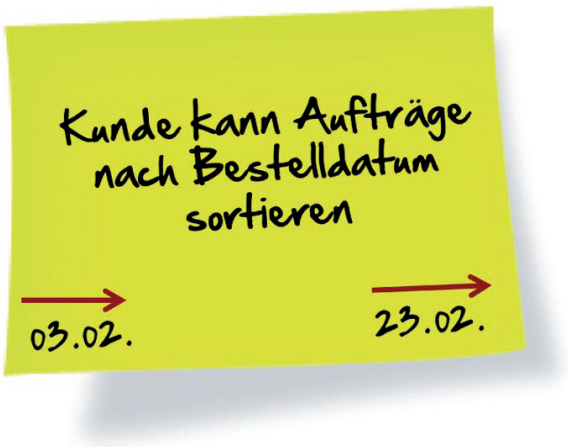

Abb. 2: Ein Ticket bei Flow Work Ltd.

„Was bedeuten die Daten?”, fragt Tobias nach dem Stand-up seinen neuen Chef.

„Sobald ein Ticket in die Next-Spalte gezogen wird, schreiben wir das aktuelle Datum darauf. Wenn das Ticket erledigt ist, notieren wir dieses Datum ebenfalls auf der Karte. Aus der Differenz zwischen Eingangs- und Ausgangsdatum lässt sich ganz einfach die Lead-Time, also die Durchlaufzeit errechnen.”

„Und wofür sind diese Daten? Bewertet ihr damit die Leistungsfähigkeit der Entwicklungsteams für die Bonuszahlungen am Jahresende?”

„Um Himmels willen, nein! Unser Ziel ist es, die durchschnittliche Lead-Time immer weiter zu reduzieren. Alles, was dieser Reduzierung im Wege steht – z. B. lange Liegezeiten oder Nacharbeiten durch Bugs oder unklare Anforderungen – wollen wir immer wieder kritisch in Frage stellen. Die Lead-Time dient also dazu, uns immer weiter zu verbessern. Darüber hinaus können wir die Lead-Time auch als Planungsinstrument verwenden, um unseren Kunden bestimmte Zusagen im Sinne von Service Level Agreements zu machen.”

„Das hört sich vernünftig an. Aber führt das nicht dazu, dass man die Aufgaben einfach immer kleiner schneidet, um so eine kürzere Lead-Time zu erreichen?”

Jetzt muss Martin lächeln: „Ja, genau das tun wir. Und das ist auch gut so! Viele kleine Tickets verkürzen die Lead-Time. So bekommt der Kunde schneller erste Features ausgeliefert, auch wenn es noch nicht das große Gesamtpaket ist. Im besten Fall kann er einzelne Features schon produktiv einsetzen. Aber auch wenn das nicht so einfach geht, ergibt sich ein riesiger Vorteil: Wir bekommen nämlich schneller Feedback vom Kunden, ob wir wirklich das entwickeln,

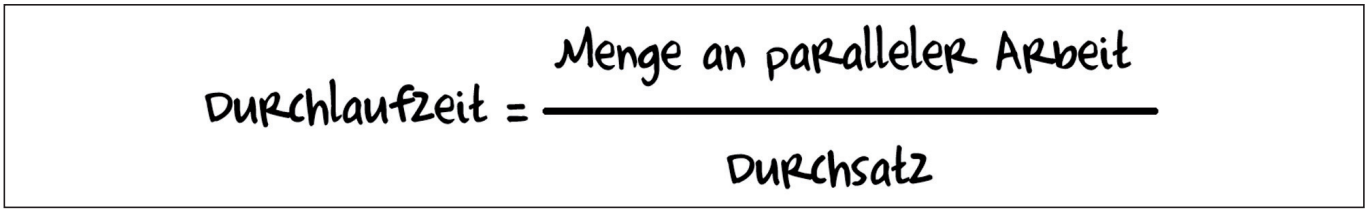

Abb. 3: Littles Gesetz.

was er sich vorgestellt hat. Voraussetzung dafür ist natürlich, dass jedes Ticket einen wirklichen Mehrwert, genannt Business Value, für den Kunden hat. Die Aufgaben müssen also immer fachlich geschnitten sein, nicht technisch!"

„Du hast die Karten auf dem Board vorhin Tickets genannt", fragt Tobias nach. „Sind damit Tickets aus dem Bug-Tracker gemeint?"

„Nein. Tickets heißen in Kanban alle Aufgaben, die auf Karteikarten, Haftnotizen oder auch Ausdrucken visualisiert und an das Board gehängt werden. Wir haben uns darauf geeinigt, dass jede Aufgabe, die voraussichtlich länger als eine Stunde für die Bearbeitung benötigt, als Ticket festgehalten und im Backlog einpriorisiert werden muss."

„Und ihr benutzt gar kein elektronisches Ticket-System?", fragt Tobias.

„In diesem Team nicht. Wir haben eine ganze Weile lang unseren Bug-Tracker parallel zum physikalischen Kanban-Board verwendet. Dann haben wir aber entschieden, dass uns das physikalische Board reicht. Die Metriken, z. B. die Auswertung der Lead-Time, erstellen wir jetzt in Excel. Andere Teams hier im Haus verwenden jedoch auch elektronische Tools."

„Was bedeuten die eingekreisten Zahlen über den Spalten Next, Analysieren, Coden und Testen? So etwas habe ich bei einem Scrum-Taskboard noch nicht gesehen."

„Das sind unsere Work-In-Progress-Limits. In jeder dieser Spalten darf sich maximal die angegebene Anzahl an Tickets befinden."

„Moment, das heißt, dass die Tester derzeit keine neuen Tätigkeiten beginnen dürfen?"

„Genau. Derzeit befinden sich drei Tickets im Test. Das Spaltenlimit für Testen ist drei. Damit darf kein neues Ticket aus der Coden-Spalte nach rechts geschoben werden. Das nennt sich das Pull-Prinzip. Anstatt immer mehr Arbeit nach hinten durchzuschieben, wird die Arbeit von hinten nach vorne gezogen", erklärt Martin.

„Aber wenn die Tester jetzt drei blockierte Tickets hätten, dann könnten sie doch an nichts mehr arbeiten. Wäre das dann nicht Verschwendung von Arbeitszeit?"

„Sieh es einmal so: Wenn wir drei blockierte Tickets im Testen haben, ist zum einen vorher etwas schief gelaufen. Und zum anderen müssen wir die drei Blockaden dann mit Hochdruck entfernen. Wenn wir nun eine weitere Aufgabe beginnen würden und diese ebenfalls blockiert wäre, hätten wir noch eine weitere unfertige Aufgabe in unserem Arbeitsfluss."

„Das leuchtet mir noch nicht ganz ein", entgegnet Tobias.

„Erinnerst du dich, wie ich vorhin meinte, dass wir regelmäßig daran arbeiten, die Lead-Time zu verkürzen? John Little hat 1961 ein Gesetz entdeckt, das bezogen auf unser Board besagt, dass die Lead-Time der Quotient aus der Menge an paralleler Arbeit und dem Durchsatz ist. Ich male dir das kurz ans Flipchart (**siehe Abbildung 3**). Anders ausgedrückt besagt Littles Gesetz, dass wir die Lead-Time verkürzen können, indem wir entweder den Durchsatz erhöhen oder aber die Menge an gleichzeitiger Arbeit reduzieren. Ersteres versuchen die meisten Unternehmen, indem sie z. B. zusätzliche Ressourcen einstellen, neue Frameworks einsetzen, Mitarbeiter anschreiben, allerdings mit mäßigem Erfolg. Letzteres ist eine sehr einfache, aber wirksame Maßnahme, die bei Kanban im Vordergrund steht."

„Das kann ich mir nur schwer vorstellen. Für mich klingt das nicht wirklich intuitiv", sagt Tobias.

„Ich weiß, so geht es den meisten am Anfang. Sei einfach offen und probiere es aus!"

Zwei Wochen später

Tobias hat sich inzwischen ganz gut eingearbeitet und die ersten Tickets auf „Done" geschoben – jedes Mal ein schönes Erfolgserlebnis. Um zu entscheiden, ob ein Ticket wirklich fertig ist, haben die Tester eine eigene „Definition of Done", die über der Test-Spalte am Board hängt. Die „Definition of Done" besagt beispielsweise, dass es für die neue Funktionalität automatisierte Tests gibt, die zukünftig als Regressionstests verwendet werden können, und dass gegebenenfalls vorhandene Lücken in der Automatisierung durch manuelles Testen in zeitlich begrenzten Testsitzungen adressiert wurden.

Die Tester bei Flow Work Ltd. arbeiten täglich mit mehreren zeitlich begrenzten Sitzungen. Manchmal schafft Tobias an einem Arbeitstag drei solcher Sitzungen von jeweils 90 Minuten, häufig werden es aber nur zwei. Während einer Sitzung geht Tobias auf Störungen von außen nur selten ein. Dadurch kann er fokussiert testen.

Zu jeder Sitzung gibt es ein Thema. Wenn Tobias eine neue Karte aus der Entwicklung nimmt, besteht seine erste Aufgabe darin, entsprechende Themen für zukünftige Testsitzungen abzuleiten. Bei neuen Funktionen bedeutet das, dass sich Tobias zunächst eine Sitzung Zeit nimmt, um einen Überblick über die Funktionen zu erlangen. Anschließend leitet er in der Nachbesprechung Themen für zukünftige Sitzungen ab. Diese setzen sich zumeist aus Bereichen zusammen, die Tobias noch nicht getestet hat oder in denen er bereits Fehler gefunden hat. Nach einigen weiteren solcher Sitzungen nehmen die möglichen neuen Themen schließlich ab, sodass er in der Nachbesprechung zusammen mit einem Kollegen die Entscheidung trifft, das Ticket in die „Done"-Spalte zu hängen. In Kombination mit der Testautomatisierung kann das Team so kontinuierlich an neuen Funktionen arbeiten, ohne Einbußen in der Qualität oder gar Regressionsfehler zu produzieren.

Nachdem Tobias das erste Ticket erledigt hat, ist er zuerst etwas verwirrt, weil ihm niemand gesagt hat, was er als nächstes tun soll. Als er dann Martin danach fragt, sagt der zu ihm: „Frag die anderen Tester, ob du ihnen helfen (oder etwas beibringen) kannst. Ansonsten zieh dir einfach das nächste Ticket aus der Spalte ‚Coden Done'". Tobias erfährt, dass sich dieses

Vorgehen „Pull-Prinzip“ nennt: Arbeit wird nicht zugewiesen, sondern selbstständig aus dem vorgelagerten Prozessschritt gezogen, sobald jemand freie Kapazitäten dafür hat. So wird eine Überlastung des Teams verhindert und es befindet sich niemals mehr Arbeit im System, als dieses verkraften kann. Allerdings ist Pull kein Wunschkonzert: Es gibt nämlich Regeln dafür, welches Ticket als nächstes gezogen werden soll. Normalerweise gilt die Regel *First in First out (FIFO)*. Davon wird nur abgewichen, wenn ein anderes Ticket als besonders dringend markiert wurde oder wenn für ein Ticket eine bestimmte Spezialisierung notwendig ist.

Besonders ungewohnt fühlt es sich an, als zuerst Tobias und dann das gesamte Testteam nicht mehr weiterarbeiten kann, weil sie alle zwar freie Kapazitäten haben, es aber keine Tickets gibt, die fertig entwickelt sind. In Tobias´ vorheriger Firma konnte so etwas nie passieren, weil jeder Tester mindestens in drei verschiedenen Projekten arbeitete und deshalb immer überlastet war. Und falls nicht, wurde er eben noch in ein viertes Projekt gesteckt. Bei Flow Work Ltd. hingegen geht man anders mit Auslastungslücken um. Das Testteam spricht sich mit den Entwicklern ab und entscheidet, dass zwei Tester bei der Entwicklung helfen sollen bzw. die Entwickler von anderen Aufgaben entlasten sollen, damit Tickets weiter fließen können. Tobias will jetzt in seiner freien Zeit ein Skript schreiben, das dabei helfen soll, in Zukunft mehr Tests zu automatisieren.

Ein paar Tage später passiert dann genau das Gegenteil: Ein Entwickler arbeitet einen Tag lang bei den Testern mit. Dabei finden sie gemeinsam heraus, warum an einer bestimmten Stelle im System immer wieder Fehler auftreten und denken sich eine Lösung für das Problem aus, die sie jetzt ausprobieren wollen. Die Spalten am Board sind also gar nicht so stark getrennt, wie man es vermuten könnte, sondern durchaus durchlässig.

Der erste Fehler

Tobias hat gerade seine letzte geplante Testsitzung an einem Ticket abgeschlossen. Als er durch seine Notizen geht, stellt er fest, dass er einen kritischen Fehler in der Software gefunden hat. Doch was soll er jetzt tun? Tobias geht zurück zum Board, und hängt sein Ticket zurück in die Spalte „Coden”.

Beim nächsten Stand-up bemerkt das Team, dass sie beim Coden derzeit ihr Work-In-Progress-Limit (WIP-Limit) nicht befolgen. Tobias meldet sich und sagt, dass er das Ticket zurückgehängt hat, weil er einen kritischen Fehler gefunden hat.

Martin erklärt Tobias, wie bei Flow Work Ltd. mit Fehlern umgegangen wird: „Ich verstehe. Das war vermutlich das naheliegendste. Aber wir behandeln Fehler leicht anders. Wenn du einen kritischen Fehler findest, dann bedeutet das, dass zuvor ein Problem entstanden ist. Damit wir nicht noch mehr Probleme auf dieser Grundlage einbauen, müssen wir das zu Grunde liegende Problem sofort beheben – und zwar mit allerhöchster Priorität. Damit wir dem Problem die höchste Priorität zuordnen, bedeutet das, dass wir die Arbeit sofort unterbrechen. In der Produktion spricht man von einem ‚Stop-The-Line‘. In den Produktionswerken, die Kanban einsetzen, hat jeder Arbeiter das Recht – ja sogar die Verpflichtung –, die Produktion völlig zu stoppen, wenn ein Fehler auftritt. Das Band darf dann erst weiter laufen, wenn der Fehler behoben ist. Bei Software-Kanban machen wir es so, dass wir ein so genanntes Expedite-Ticket für die Fehlerbehebung erstellen. Dieses spezielle Ticket bekommt die oberste Priorität. Wir haben auf unserem Board eine eigene Zeile mit einem Limit von einem Ticket dafür abgestellt.”

„Aber das bedeutet ja, dass wir genau einen kritischen Fehler haben dürfen”, wirft Tobias ein.

„Genau. Fehler stauen sich dadurch aber auch nicht an. Dadurch ist es möglich, von Anfang an die notwendige Qualität im System sicher zu gewährleisten. Das ursprüngliche Ticket bleibt dort, wo der Fehler aufgefallen ist, und wird mit einem roten Punkt als blokkiert markiert. Wenn das Expedite-Ticket das blockierte eingeholt hat, kann die Blockade wieder entfernt und das Ticket normal weitergeschoben werden.“

„Ok, also schiebe ich mein Ticket wieder zurück nach Testen. Aber wie erstelle ich jetzt ein Expedite-Ticket?“

„Lass uns das sofort adressieren.”

Das Team hat eine kurze Diskussion darüber, wo der Fehler wohl entstanden sein könnte. Wie sich später herausstellt, waren die Anforderungen missverständlich formuliert worden. Dadurch musste das Expedite-Ticket nochmal die tatsächlichen Anforderungen für das System bestimmen. Tobias ist froh, dass dieser Fehler noch im Entwicklungszyklus aufgefallen ist. Später hätte das Problem sicherlich einen großen Verlust für Flow Work Ltd. bedeuten können.

Kanban

An dieser Stelle verlassen wir unsere kleine Geschichte um Tobias, den Tester, der Kanban kennenlernt, um noch einmal die Kernelemente von Kanban zusammenzufassen. Kanban – so wie wir es in unserer fiktiven Geschichte gesehen haben – besteht im Kern aus vier Prinzipien[1]) und einem Gesetz. Zunächst einmal geht es darum, den Arbeitsprozess zu visualisieren. Wenn wir in der Lage sind, die Abläufe über die Zeit zu beobachten, entwickeln wir ein Gespür für die Probleme in unserer Arbeitsweise. In der Geschichte beispielsweise wurde ein Fehler in der Software durch ein spezielles Ticket visualisiert. Oft sind diese Expedite-Tickets rot, sodass sie dem Betrachter besonders hervorstechen. Einige Teams haben eine eigene Reihe für diese Art von Tickets, die dann ganz oben auf dem Kanban-Board erscheint. Dadurch sind sie natürlich direkt im Fokus des Betrachters. Jeder, der am Board vorbeigeht, wird auf das Problem aufmerksam gemacht. Ein Grund mehr für das Team, mit Hochdruck weiter daran zu arbeiten.

Das zweite Prinzip in unserem Beispiel ist die Limitierung von Arbeit. Ständiges Multitasking ist nicht gut für das menschliche Gehirn. Deswegen wollen wir uns um maximal zwei Aufgaben gleichzeitig kümmern: eine, an der wir arbeiten; wenn diese blokkiert ist, können wir noch eine zweite beginnen und diese hoffentlich vor dem Lösen der Blockade abschließen. Durch die Limitierung von gleichzeitiger Arbeit verringern wir aber nicht nur die Gefahr des Burnouts, sondern erlauben uns, in unseren Entwicklungsprozessen schneller zu werden.

[1]) Die hier genannten Prinzipien weichen von denen im Buch von David Andersson ab. Das Pull-Prinzip ist dabei eng verknüpft mit dem im Original zu findenden „Den Fluss managen”. Kaizen ist ein übergeordnetes Prinzip von „Verwende Modelle, um Chancen für kollaborative Verbesserungen zu erkennen”, dem fünften Prinzip im Original. Es fehlt das vierte Original-Prinzip: „Mache die Regeln für den Prozess explizit”.

Das Gesetz von Little besagt, dass wir die Durchlaufzeit – also die Zeit, die eine Arbeit von Beginn bis Ende braucht – erhöhen können, indem wir entweder den Durchsatz erhöhen oder die Menge an paralleler Arbeit reduzieren. Der Durchsatz ist durch den derzeitigen Prozess beschränkt. Wenn wir also schneller arbeiten möchten, müssen wir die Menge an gleichzeitiger Arbeit reduzieren, um noch schneller ausliefern zu können.

Das dritte Prinzip in der Geschichte ist das Pull-Prinzip. Statt in den Entwicklungsprozess stetig Arbeit hineinzuschieben, wird in Kanban die Arbeit durch das System gezogen. Ein Mitarbeiter, der freie Kapazitäten hat, nimmt sich das nächste ausstehende Ticket. Zu Beginn bedeutet das häufig, dass er nur eingeschränkt an seinen Spezialthemen arbeiten kann. Über die Zeit kann es sein, dass das Team sich gleichmäßiger aufteilt, um noch mehr Arbeit schaffen zu können.

Das letzte Prinzip lautet Kaizen. Dabei geht es um einen kontinuierlichen Verbesserungsprozess. Das Team trifft sich regelmäßig und verbessert seinen Arbeitsprozess. Das geschieht im Rahmen einer Retrospektive oder aber, wenn ein Teammitglied wegen des vorhandenen Limits gerade keine neue Tätigkeit aufnehmen kann oder darf. Dann kümmert sich dieser darum, den Arbeitsprozess zu verbessern, Altlasten aufzuräumen oder neue Wege zu erforschen, um noch besser zu werden.

Für Tester bedeutet Kanban vor allem, dass sie zunächst einmal wie gewohnt ihrer Arbeit nachgehen können. Der einzige Unterschied besteht darin, dass der Arbeitsablauf visualisiert ist und man sich täglich darüber austauscht, wie man das nächste Stück der Software einen Schritt weiter bekommen kann. Mittelfristig haben sie aber auch die Möglichkeit, sich am kontinuierlichen Verbesserungsprozess zu beteiligen. Prozessverbesserungen können dabei sukzessive durch kleine Maßnahmen aus der Retrospektive vorgenommen werden – oder eher radikal dadurch, dass ein großes Problem sofort behoben werden muss. Jeder im Team hat die Möglichkeit, dieses zusammenzurufen und ein Problem zu beseitigen. Bei Tobias war das der Fall, als ein Entwickler das Ticket mit dem Defekt entdeckt hat.

Allerdings gibt es für Kanban auch eine Grenze. Kanban macht – wie alle agilen Vorgehensweisen – zunächst ein Problem transparent. Wie eine Firma oder Firmenkultur darauf reagiert, kann dann ganz unterschiedlich sein. In einer Kultur, in der Fehler und Probleme lieber versteckt werden, wird Kanban folglich nicht ohne weitere Unterstützung funktionieren. Andererseits steht einer erfolgreichen Etablierung nichts mehr im Wege, wenn das Team selbstkritisch und offen mit dieser Transparenz umgehen kann. ■