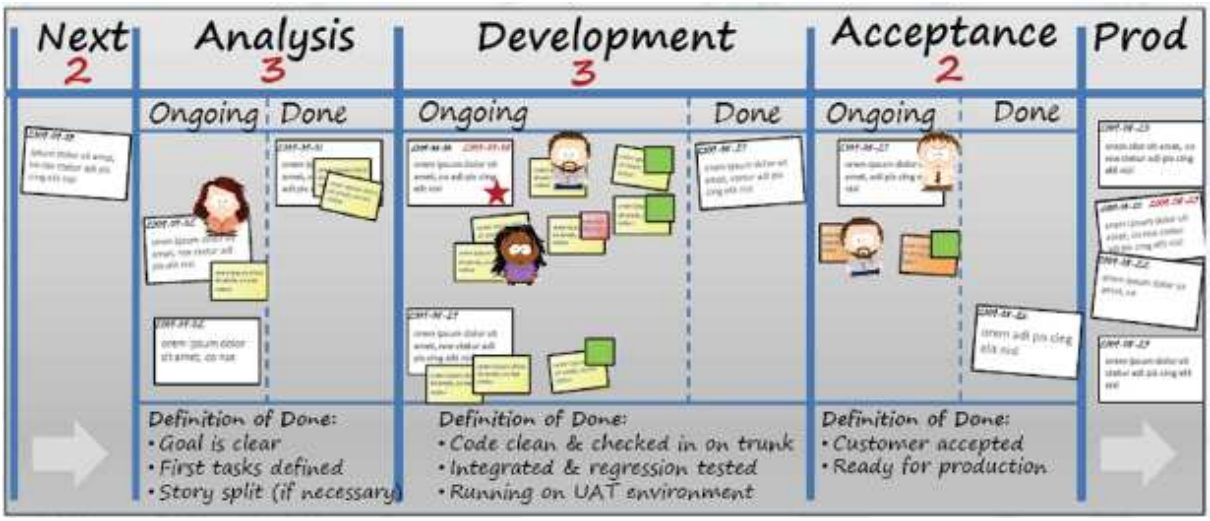
Henrik Kniberg & Mattias Skarin

Kanban und Scrum

Optimaler Einsatz beider Methoden



→ Aus dem Englischen übersetzt von Maria Oelinger

Quelle: www.infoq.com/minibooks/kanban-scrum-minibook

1 Vorwort

Vorwort von Mary Poppendieck

Henrik Kniberg ist einer dieser seltenen Menschen, die die Essenz aus einer komplizierten Sache extrahieren können, die Kernideen aus gelegentlichen Störungen herausfiltern und eine kristallklare Erklärung liefern, die unglaublich leicht zu verstehen ist. In diesem Buch erklärt Henrik auf brilliante Art und Weise den Unterschied zwischen Scrum und Kanban. Er macht klar, dass dies nur Werkzeuge sind, und dass das, was Sie wirklich möchten, ein voller Werkzeugkasten ist, Sie möchten die Stärken und Grenzen jedes Werkzeugs verstehen und wissen, wie alle diese Werkzeuge eingesetzt werden. In diesem Buch werden Sie lernen, worum es bei Kanban geht, seine Stärken und Grenzen, und wann man es einsetzt. Außerdem werden Sie eine gute Lehrstunde darüber bekommen, wie und wann Sie etwas im Hinblick auf Scrum verbessern können, oder bei irgendeinem anderen Werkzeug, dass Sie zufällig einsetzen. Henrik stellt klar, dass das Wichtige nicht das Werkzeug ist, mit dem Sie beginnen, sondern die Art, wie sie ständig Ihren Einsatz dieses Werkzeugs verbessern und wie Sie Ihre Menge an Werkzeugen über die Zeit ausbauen.
Der zweite Teil dieses Buchs von Mattias Skarin macht das Buch noch effektiver, indem es Sie durch die Anwendung von Scrum und Kanban in einer echten Praxissituation führt. Da werden Sie ein Beispiel sehen, wie die Werkzeuge einzeln und in Kombination eingesetzt wurden, um einen Softwareentwicklungsprozess zu verbessern. Sie werden sehen, dass es keinen einzeigen „besten“ Weg gibt, Dinge zu erledigen; Sie müssen selbst nachdenken und – auf der Basis Ihrer eigenen Situation – den nächsten Schritt hin zu einer besseren Softwareentwicklung herausfinden.

Mary Poppendieck

Vorwort von David Anderson

Kanban basiert auf einer einfachen Idee. Work in Progress (WIP, das sind die aktuell bearbeiteten Arbeitspakete) sollte begrenzt werden und etwas Neues sollte erst dann begonnen werden, wenn ein bestehendes Arbeitspaket ausgeliefert oder in einen nachgeschalteten Arbeitsgang gezogen wird. Die Kanbankarte (jap: Kanban = Signalkarte) stellt ein visuelles Signal dar, mit dem angezeigt wird, dass ein neues Arbeitspaket gezogen werden kann, weil aktuell weniger Arbeitspakete im Bearbeitungsprozess sind, als es die Beschränkung (das WIP-Limit) erlaubt. Das hört sich weder sehr revolutionär an noch so, als ob es einen schwerwiegenden Effekt auf die Effizienz, die Kultur, die Leistungsfähigkeit oder Reife eines Teams und seiner umgebenden Organisation hätte. Was erstaunlich ist, diesen Effekt hat es! Kanban scheint eine so kleine Änderung zu sein und doch verändert es alles rund um den laufenden Betrieb.

Uns ist nach und nach bewusst geworden, dass Kanban eine Variante des Veränderungsmanagements ist. Es ist keine bloße Softwareentwicklungsmethode oder Projektmanagement-Lifecycle und kein schlichter Prozess. Kanban ist ein Ansatz, um Veränderung bei einem bestehenden Lebenszyklus von Softwareentwicklung oder einer Projektmanagementmethode zu initiieren. Das Kanban-Prinzip besteht darin, mit dem Vorgehen zu beginnen, das man derzeit hat. Den aktuell eingesetzten Prozess macht man sich dadurch bewusst, indem man die Wertschöpfungskette visualisiert und dann für jede Phase in diesem Prozess WIP-Limits festlegt. Dann beginnt man damit, die Arbeitspakete durch diese Visualisierung je nach Stand der Arbeit hindurch laufen zu lassen, indem jedes Arbeitspaket in die nächste Phase gezogen wird, wenn Kanbankarten verfügbar sind und das WIP-Limit es zulässt.

Kanban stellte sich als nützlich für Teams heraus, die Softwareentwicklung agil betreiben, aber genauso bekommen mit Kanban Teams Zugkraft, die einen traditionelleren Entwicklungsweg einschlagen. Kanban wurde als Teil der Lean-Management-Initiative eingeführt, um Unternehmenskulturen zu verändern und zu kontinuierlichen Verbesserungen anzuregen.

Da WIP (die Anzahl der Pakete, die gerade bearbeitet werden) in Kanbansystemen begrenzt ist, verstopft alles, was durch welchen Grund auch immer blockiert wird, den Arbeitsfluss. Wenn genug Arbeitspakete blockiert sind, ist soviel Sand im Getriebe, dass der gesamte Prozess zum Halten kommt. Das hat den Effekt, dass das ganze Team sich auf die Blockaden und die umgebende Organisation sich auf die Problemlösung fokussieren, um die Blockaden aufzulösen und den Fluss der Arbeit wieder herzustellen.

Kanban nutzt einen sichtbaren Kontrollmechanismus, um den Arbeitsablauf durch den Prozess (die Wertschöpfungskette) mit seinen verschiedenen Phasen nachzuverfolgen. Typischerweise verwendet man dazu ein Whiteboard mit Haftnotizen oder ein elektronisches Kartenwandsystem. In der Praxis ist es wahrscheinlich am besten, beides zu nutzen. Die Transparenz, die damit erzeugt wird, trägt auch zur Veränderung der Organisationskultur bei. Agile Methoden liefern eine hohe Transparenz bezüglich der Arbeitspakete, die aktuell bearbeitet werden, bezüglich der erledigten Aufgaben und bezüglich Auswertungen wie Durchlaufgeschwindigkeit (die Menge der erledigten Aufgaben pro Iteration). Kanban geht jedoch noch einen Schritt weiter und stellt Transparenz des eigentlichen Arbeitsprozesses und des Arbeitsablaufes bereit. Kanban legt Engpässe, Warteschlangen, Schwankungen und Ausschuss offen – alles Dinge, die die Effizienz einer Organisation hinsichtlich der Menge wertvoller abgelieferter Arbeit sowie die Zeit, die für die Fertigstellung notwendig ist, beeinflussen.

Kanban sorgt dafür, dass für Teammitglieder und externe Stakeholder (Beteiligte, Interessengruppen) die Wirkung ihrer Aktionen (oder die Wirkung ihrer Untätigkeit) sichtbar wird. Insofern zeigen frühere Fallstudien, dass Kanban Verhalten ändert und zu besserer Zusammenarbeit am Arbeitsplatz führt. Das Sichtbarmachen von und das Einwirken auf Engpässe, Ausschuss und Schwankungen ermutigt außerdem zur Diskussion über Verbesserungen, und Teams beginnen schnell, Verbesserungen in ihrem Prozess aufzugreifen.

Als Konsequenz ergibt sich, dass Kanban die schrittweise Weiterentwicklung existierender Prozesse und die Weiterentwicklung, die in der Regel mit agilem und Leanmanagement verbunden ist, fördert. Kanban verlangt nicht nach einer pauschalen Revolution für die Arbeitsweise von Leuten, es zieht eine solche eher durch allmähliche kleinere Veränderungen nach sich. Die Veränderung durch Kanban versteht sich so, dass die Änderungen von allen verstanden und getragen werden, die im Prozess und an seinen Schnittstellen zusammenarbeiten.

Durch das Wesen des Pull-Prinzips fördert Kanban außerdem eine späte Festlegung, und zwar sowohl was die Priorisierung neuer Arbeitspakete als auch die Auslieferung angeht. Typischerweise einigen sich Teams auf einen Priorisierungsrhythmus, in dem sie sich mit Stakeholdern aus höheren Hierarchieebenen treffen und entscheiden, an was als nächstes gearbeitet werden soll. Diese Treffen können oft abgehalten werden, weil sie normalerweise sehr kurz sind. Eine einfache Frage muss beantwortet werden, z. B. „Seit unserm letzten Treffen sind zwei Slots freigeworden. Unsere aktuelle Durchlaufzeit beträgt 6 Wochen bis zur Auslieferung. Welche zwei Ergebnisse möchten Sie am liebsten in 6 Wochen zur Verfügung haben?“ Dieses Vorgehen enthält zwei Aspekte: um eine einfache, normalerweise schnell gefundene Antwort bitten und außerdem das Treffen möglichst kurz halten. Diese Form des Fragens bedeutet, dass die Festlegung, welche Arbeitsaufgaben erledigt werden sollen, verzögert wird bis zum letzten Zeitpunkt, der sich dafür verantworten lässt. Das verbessert die Agilität durch Steuerung der Erwartungshaltungen, Verkürzung der Dauer zwischen Festlegung und Auslieferung und vermeidet Nacharbeiten, da die Möglichkeit, dass sich Prioritäten ändern, auf ein Mindestmaß verkleinert.

Als abschließendes Wort zu Kanban soll genannt sein, dass die Begrenzung der gleichzeitig bearbeiteten Arbeitspakete die Vorhersagbarkeit von Durchlaufzeiten und verlässlichere Leistungen bei Auslieferung bewirkt. Der Punkt, bei Hindernissen und Fehlern die „Produktionslinie zu stoppen“ scheint außerdem hohe Qualitätslevel und eine rapide Abnahme von Überarbeitungsaufwand zu fördern.

Während all dies sich durch die wunderbaren Erklärungen in diesem Buch als offenkundig erweisen wird, ist völlig ungeklärt, wie wir dorthin kommen. Kanban wurde nicht an einem einzigen Nachmittag durch eine unglaubliche Erleuchtung konzipiert – es entwickelte sich vielmehr allmählich über Jahre hinweg. Viele der tiefgreifenden psychologischen und soziologischen Effekte, die die Kultur, Leistungsfähigkeit und Reife von Organisationen verändern, waren zu Beginn noch gar nicht abzusehen. Es war eher so, dass diese Effekte im Laufe der Zeit entdeckt wurden. Viele der Ergebnisse beim Einsatz von Kanban sind nicht intuitiv begreifbar. Was wie ein sehr mechanischer Vorgang erscheint – WIP begrenzen und nach dem Pullprinzip arbeiten – wirkt weitreichend auf die beteiligten Personen und wie sie miteinander interagieren und zusammenarbeiten. Weder ich noch irgendwer sonst, der an der frühen Verbreitung von Kanban beteiligt ist, hat das vorhergesehen.

Ich habe weiterverfolgt, was Kanban als das Vorgehensmodell wurde, das Änderungen mit minimalem Widerstand implementiert. Mir war das schon 2003 bewusst. Ich habe es außerdem weiterverfolgt wegen des unwillkürlichen Nutzens. Da ich durch die Anwendung der Leanmethoden über die Zeit entdeckt habe, dass das Steuern des WIP sinnvoll ist, erwies sich das Begrenzen von WIP als noch sinnvoller: es nahm den Managementmehraufwand aus dem Managen heraus. Also entschied ich mich 2004, ein Pullsystem auf Basis der Grundprinzipien aufzubauen. Ich bekam die Gelegenheit, als ein Microsoft-Manager an mich herantrat und mich bat, ihm dabei zu helfen, einen Veränderungsprozess in einem seiner Wartungsteams für Upgrades von internen IT-Anwendungen zu managen. Die erste Umsetzung basierte auf einer Pullprinzipvariante der „Theory of Constraints“ (Theorie der Einschränkung), bekannt als „Drum-Buffer-Rope“. Das war ein gewaltiger Erfolg: Die Durchlaufzeit senkte sich um 92 %; der Durchsatz erhöhte sich um mehr als das Dreifache; und die Vorhersagbarkeit (Termintreue) lag bei sehr akzeptablen 98 %.

2005 überredete mich Donald Reinertsen, ein komplettes Kanbansystem zu implementieren. Ich bekam 2006 die Gelegenheit, als ich die Verantwortung übernahm für die Softwareentwicklungsabteilung bei Corbis in Seattle. 2007 begann ich, über die Ergebnisse zu berichten. Die erste Präsentation fand im Mai 2007 beim Lean New Product Development Summit in Chicago statt. Ich setzte das mit einem Open-Space-Event bei der Agile 2007 in Washington DC im August desselben Jahres fort. 25 Leute nahmen teil und drei von ihnen kamen von Yahoo!: Aaron Sanders, Karl Scotland und Joe Arnold. Sie gingen zurück nach Hause nach Kalifornien, Indien und das Vereinigte Königreich und nutzten Kanban für ihre Teams, die sich bereits mit Scrum abmühten. Sie setzten außerdem ein Yahoo!-Diskussionsforum auf, das – jetzt, da ich es aufschreibe – fast 800 Mitglieder hat. Kanban begann, sich auszubreiten und frühe Anwender sprachen über ihre Erfahrungen.

Jetzt, 2009, wächst der Anteil der Kanbananwender spürbar an und es kommen mehr und mehr Erfahrungsberichte treffen ein. Wir haben viel über Kanban gelernt in den vergangenen fünf Jahren und wir lernen jeden Tag immer noch mehr dazu. Ich selbst habe mich bei meiner eigenen Arbeit auf Kanban konzentriert, schreibe über Kanban, spreche über Kanban und denke über Kanban nach, um es besser zu verstehen und es Anderen zu erklären. Ich habe ganz bewusst davon Abstand genommen, Kanban mit existierenden agilen Methoden zu vergleichen, obwohl 2008 einiger Aufwand in die Erklärung geflossen ist, warum Kanban es verdient hat, als agil-kompatible Methode betrachtet zu werden.

Ich habe es Anderen mit mehr Erfahrung überlassen, Fragen wie „Wie verhält sich Kanban zu Scrum?“ zu beantworten. Ich bin so froh, dass Henrik Kniberg und Matthias Skarin sich in dieser Beziehung als führend erwiesen haben. Sie, der Wissensarbeiter auf diesem Gebiet, brauchen Informationen, um fundierte Entscheidungen zu treffen und mit Ihrer Arbeit weiter zu kommen. Henrik und Mattias decken diesen Bedarf ab, wie ich es nie könnte. Ich bin insbesondere beeindruckt von Henriks bedachtem Ansatz des Vergleichs und davon, wie er das Thema sachlich und nicht von sich selbst überzeugt, sondern ausgewogen behandelt. Seine Cartoons und Illustrationen sind besonders aufschlussreich und ersparen oft eine Menge Seiten an Text. Mattias’ Praxisstudie ist wichtig, weil sie zeigt, dass Kanban viel mehr ist als Theorie und weil sie am Beispiel zeigt, wie nützlich Kanban für Ihre Organisation sein könnte.

Ich hoffe, dass Sie dieses Buch genießen, das Kanban und Scrum vergleicht, und dass es Ihnen mehr Einblick in die Agilität im Allgemeinen gibt, besonders in Kanban und Scrum. Wenn Sie mehr über Kanban lernen möchten, besuchen Sie bitte unsere Community-Website, The Limited WIP Society, http://www.limitedwipsociety.org/

David J. Anderson
Sequim, Washington, USA
July 8th, 2009

2 Einführung

Normalerweise schreiben wir keine Bücher. Wir ziehen es vor, unsere Zeit tief drinnen in der täglichen Praxis zu verbringen, wo wir Kunden helfen, ihre Entwicklungsprozesse und Organisation zu optimieren, Fehler darin auszumerzen und neu zu strukturieren. In letzter Zeit jedoch haben wir einen klaren Trend festgestellt, und wir möchten unsere Gedanken dazu mit den Lesern hier teilen. Folgendes ist ein typischer Fall:

• Jim: “Jetzt arbeiten wir endlich total nach Scrum!”

• Fred: “Und wie läuft’s?”

• Jim: “Naja, es ist viel besser als vorher...”

• Fred: “...aber?”

• Jim: “... aber, Du weißt, wir sind ein Support- und Wartungsteam.”

• Fred: “Ja, und?”

• Jim: “Naja, wir lieben diese ganze Priorisierung im Produktbacklog, selbstorganisierte Teams, tägliche Scrums, Retrospektiven etc...”

• Fred: “Was ist das Problem?”

• Jim: “Wir halten unsere Sprints nicht ein.”

• Fred: “Warum?”

• Jim: “Weil wir es schwierig finden, uns auf einen 2-Wochen-Plan festzulegen. Iterationen sind für uns nicht sinnvoll, wir arbeiten einfach an dem, was heute am dringendsten ist. Sollen wir vielleicht auf wöchentliche Iterationen umsteigen?”

• Fred: “Könntet Ihr Euch auf Arbeitspakete für 1 Woche festlegen? Habt Ihr die Möglichkeit, Euch eine Woche lang in Ruhe auf diese Arbeitspakete zu konzentrieren und sie abzuarbeiten?”

• Jim: “Das nun auch nicht, wir bekommen täglich neue Sachen rein. Vielleicht, wenn wir tägliche Sprints hätten…”

• Fred: “Schafft Ihr, die Sachen in weniger als einem Tag abzuarbeiten?”

• Jim: “Nein, manchmal braucht man mehrere Tage.”

• Fred: “Also würden 1-Tag-Sprints auch nicht funktionieren. Habt Ihr mal überlegt, Sprints völlig fallen zu lassen?”

• Jim: “Naja, um ehrlich zu sein, wir würden das begrüßen. Aber ist das nicht gegen die Scrum-Regeln?”

• Fred: “Scrum ist nur ein Werkzeug, eine Methode. Ihr bestimmt, wann und wie Ihr es einsetzt. Lasst Euch davon nicht versklaven!”

• Jim: “Was sollten wir denn stattdessen tun?”

• Fred: “Hast Du mal von Kanban gehört?”

• Jim: “Was ist das? Was ist der Unterschied zwischen dem und Scrum?”

• Fred: “Hier, lies das Buch!”

• Jim: “Aber ich mag den Rest von Scrum doch wirklich, muss ich jetzt alles umstellen?”

• Fred: “Nein, Ihr könnt die beiden Methoden kombinieren!”

• Jim: “Was? Wie?”

• Fred: “Lies einfach weiter...”

3 Ziel dieses Buches

Wenn Sie sich für agile Softwareentwicklung interessieren, haben Sie wahrscheinlich von Scrum gehört, und vielleicht sogar von Kanban. Eine Frage, die wir immer öfter hören ist, „also was ist Kanban, und wie verhält es sich zu Scrum?“ Wo ergänzen sie sich? Gibt es mögliche Konfliktpunkte?

Das Ziel dieses Buches ist, den Nebel zu lichten, so dass Sie selbst herausfinden können, wie Kanban und Scrum Ihnen in Ihrem Umfeld nützen können.

Lassen Sie uns wissen, ob wir Erfolg hatten!

Teil I – Vergleich

Der erste Teil des Buches ist ein Versuch, einen objektiven und praktischen Vergleich zwischen Scrum und Kanban zu ziehen. Es ist eine leicht aktualisierte Version des Originalartikels "Kanban vs. Scrum" vom April 2009. Dieser Artikel wurde so beliebt, dass ich mich entschloss, ihn als Buch herauszubringen, und ich bat meinen Kollegen Mattias, ihn ein bisschen aus der Praxis heraus mit einer Fallstudie eines unserer Kunden aufzupeppen. Tolles Material! Sie können auch direkt in Teil II einsteigen, wenn Sie lieber mit der Fallstudie einsteigen möchten, ich werde nicht beleidigt sein.

Naja, vielleicht ein bisschen.

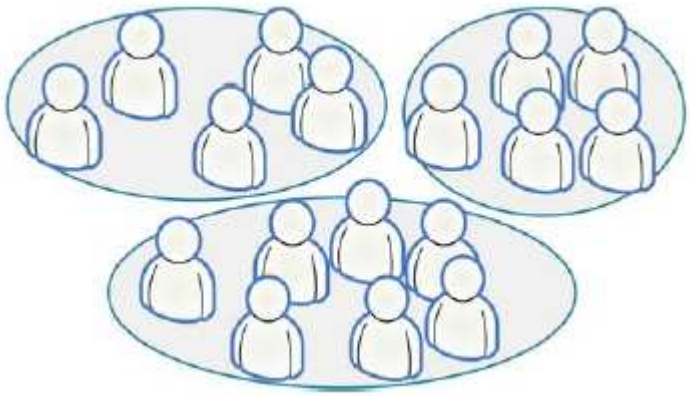
/Henrik Kniberg

4 Was sind denn jetzt Scrum und Kanban?

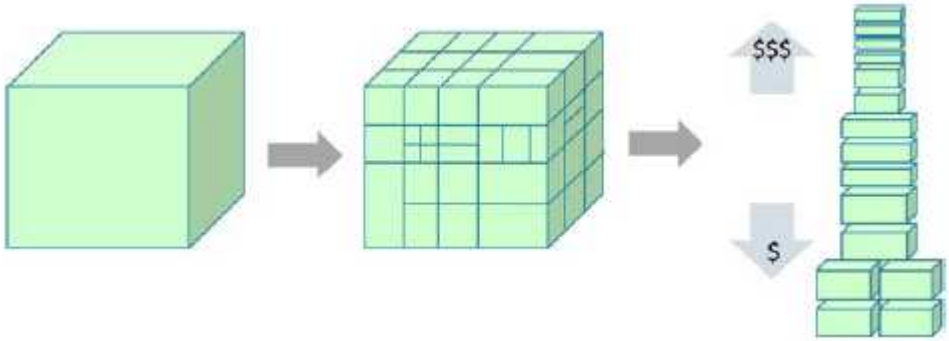
OK, lassen Sie uns versuchen, Scrum und Kanban in jeweils weniger als 100 Wörtern zusammenzufassen.

4.1 Scrum kurzgefasst

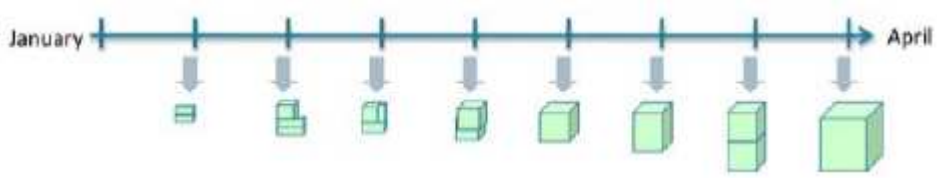
- **Teilen Sie Ihre Organisation auf** in kleine, funktionsübergreifende, selbstorganisierte Teams.



- **Teilen Sie Ihre Arbeit auf** in eine Liste aus kleinen, konkreten Auslieferungspaketen. Sortieren Sie die Liste nach Priorität und schätzen Sie den relativen Aufwand für jeden Eintrag.



- **Teilen Sie die Zeit auf** in kurze Iterationen fester Länge (normalerweise 1 – 4 Wochen), mit auslieferungsfähigem Code nach jeder Iteration.



- **Optimieren Sie den Releaseplan** und aktualisieren Sie die Prioritäten zusammen mit dem Kunden, auf der Basis der Erkenntnisse, die Sie nach der Kontrolle nach jeder Iteration gewonnen haben.

- **Optimieren Sie den Prozess**, indem Sie nach jeder Iteration eine Retrospektive durchführen.

Also statt eine **große Gruppe** eine **lange Zeit** verbringen zu lassen, um ein **großes Ding** zu bauen, haben wir ein **kleines Team**, das in einer **kurzen Zeit** ein **kleines Ding** baut. Aber mit **regelmäßiger Integration**, um das Ganze zu sehen.

168 Wörter[1]... nah genug dran.

Für mehr Details nehmen Sie sich “Scrum and XP from the Trenches” vor. Das Buch kann man kostenlos online lesen. Ich kenne den Autor, er ist ein netter Kerl :o)

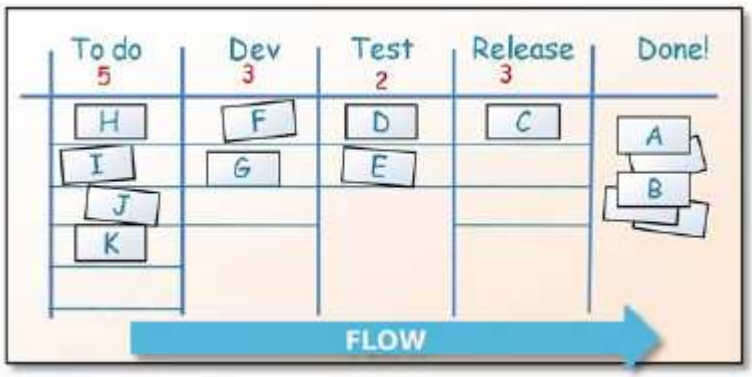
[1] im Englischen sind es 121 Wörter

http://www.crisp.se/ScrumAndXpFromTheTrenches.html

Mehr Scrumlinks unter http://www.crisp.se/scrum

4.2 Kanban kurzgefasst

- **Visualisieren Sie den Arbeitsablauf**
    - Teilen Sie die Arbeit in kleine Teile, schreiben jeden Punkt auf eine Karte und hängen sie an die Wand.
    - Setzen Sie Spaltentitel ein, um darzustellen, wo im Arbeitsablauf ein Arbeitspaket gerade steckt.
- **Limit Work In Progress (WIP) Begrenzen Sie der Arbeitspakete pro Arbeitsschritt** – ordnen Sie explizit Begrenzungen zu, wieviel Arbeitspakete in jedem Schritt des Arbeitsablaufs sein dürfen.
- **Messen Sie Durchlaufzeiten** (die mittlere Dauer, um ein Arbeitspaket abzuschließen, manchmal auch als “cycle time” bezeichnet), optimieren Sie den Prozess, um die Durchlaufzeit so kurz und vorhersehbar wie möglich zu machen.



Wir sammeln nützliche Kanbanlinks unter http://www.crisp.se/kanban

5 Wie verhalten sich Scrum und Kanban zueinander?

5.1 Scrum und Kanban sind beides Prozesswerkzeuge

Werkzeug = irgendetwas, das man als Mittel einsetzt, um eine Aufgabe zu bewerkstelligen oder einen Zweck zu erfüllen

Prozess = wie man arbeitet

Scrum und Kanban sind Prozesswerkzeuge insofern, als sie Ihnen helfen, effektiver zu arbeiten, indem sie Ihnen – in gewissem Maße – sagen, was zu tun ist. Java ist auch ein Werkzeug, es hilft Ihnen dabei, auf eine einfachere Weise einen Computer zu programmieren. Eine Zahnbürste ist auch ein Werkzeug, sie hilft Ihnen, Ihre Zähne so zu erreichen, dass Sie sie reinigen können.

5.2 Werkzeuge vergleichen, um zu verstehen, nicht um zu bewerten

Messer oder Gabel – welches Werkzeug ist besser?

Eine ziemlich unsinnige Frage, oder? Weil die Antwort vom Kontext abhängt. Um Frikadellen zu essen, ist die Gabel wahrscheinlich am besten. Um Pilze zu zerhacken, ist das Messer wahrscheinlich am besten. Um auf dem Tisch zu trommeln, funktionieren beide prima. Um ein Steak zu essen, möchten Sie wahrscheinlich beide zusammen benutzen. Um Reis zu essen… naja… Einige bevorzugen eine Gabel, während Andere Stäbchen lieber mögen.

Wir sollten also umsichtig vorgehen, wenn wir Werkzeuge vergleichen. Vergleichen, um zu verstehen, nicht um zu bewerten.

5.3 Kein Werkzeug ist komplett, kein Werkzeug ist perfekt

Wie alle Werkzeuge, sind Scrum und Kanban weder perfekt noch vollständig. Sie sagen Ihnen nicht alles, was Sie tun müssen, sie stellen nur ein bestimmtes Gerüst und bestimmte Richtlinien bereit. Beispielsweise zwingt Scrum Sie, Iterationen in festen zeitlichen Abständen einzuhalten und funktionsübergreifende Teams einzusetzen, und Kanban zwingt Sie zu visualisieren und die Größe der Warteschlangen zu begrenzen.

Interessanterweise besteht der Wert eines Werkzeugs darin, dass es Ihre *Alternativen begrenzt*. Ein Prozesswerkzeug, das Sie alles tun lässt, ist wenig brauchbar. Wir könnten diesen Prozess „Tu-was-auch-immer“ nennen oder wie wäre es mit „Tu-das-Richtige“. Der „Tu-das-Richtige“-Prozess funktioniert garantiert, es ist der Königsweg! Denn wenn es nicht funktioniert, haben Sie sich offensichtlich nicht an den Prozess gehalten :o)

Die richtigen Werkzeuge zu benutzen, wird Ihnen helfen, erfolgreich zu sein, wird den Erfolg aber nicht garantieren. Es ist leicht, *Projekt*erfolg/-misserfolg und *Werkzeug*erfolg/-misserfolg durcheinander zu bringen.

- Ein Projekt kann erfolgreich sein wegen eines großartigen Werkzeugs.
- An Projekt kann erfolgreich sein trotz eines lausigen Werkzeugs.
- An Projekt kann scheitern wegen eines lausigen Werkzeugs.
- Ein Projekt kann scheitern trotz eines großartigen Werkzeugs.

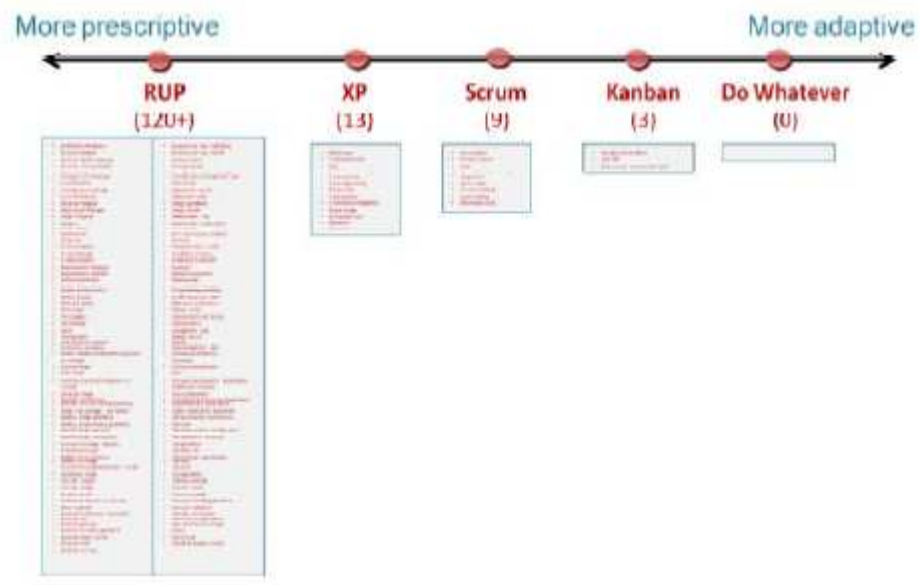
5.4 Scrum ist standardisierter als Kanban

Wir können Werkzeuge vergleichen, indem wir uns ansehen, wieviele Regeln sie mitbringen. *Standardisiert* bedeutet hier “mehr Regeln ” und *adaptiv* bedeutet “weniger Regeln”. 100 % adaptiv bedeutet: Tu was auch immer, es gibt überhaupt keine Regeln oder Einschränkungen. Wie Sie sehen, sind beide Extreme dieser Skalen auf ihre Art lächerlich.

Agile Methoden werden manchmal *leichtgewichtige* Methoden genannt, insbesondere weil sie weniger standardisiert vorschreiben, was zu tun ist, als herkömmliche Methoden. Tatsächlich steht im ersten Grundsatz des agilen Manifestes: „Individuen und Interaktionen sind wichtiger als Prozesse und Werkzeuge“.

Scrum und Kanban sind beide hochgradig adaptiv, aber *relativ gesehen* ist Scrum standardisierter als Kanban. Scrum gibt mehr Grenzen vor, und lässt damit weniger Alternativen zu. Zum Beispiel schreibt Scrum vor, zeitlich festgelegte Iterationen anzuwenden, was Kanban nicht tut.

Lassen Sie uns einige weitere Prozesswerkzeuge auf der Standardisiert-adaptiv-Skala vergleichen:



RUP ist recht standardisiert – es umfasst mehr al 30 Rollen, mehr als 20 Aktivitäten und mehr als 70 Artefakte; eine überwältigende Menge an Stoff, den man lernen muss. Obwohl niemand von Ihnen erwartet, dass Sie all das nutzen; Sie sollen eine für Ihr Projekt passende Untermenge auswählen. Leider scheint das in der Praxis schwierig zu sein. „Hmmmm… werden wir eine *Konfiguration zur Revision von Artefakten* brauchen? Werden wir eine *Veränderungskontrollmanagerrolle* brauchen? Bin nicht sicher, also nehmen wir sie vorsichtshalber mit rein.“ Das mag ein Grund dafür sein, warum der Einsatz von RUP oft ziemlich schwergewichtig daherkommt, verglichen mit agilen Methoden wie Scrum und XP.

XP (eXtreme Programming) ist recht standardisiert, verglichen mit Scrum. Es umfasst das meist von Scrum + einen Haufen ziemlich spezifischer Entwicklungsvorgaben wie testgetriebene Entwicklung und Pair-Programming.

Scrum ist weniger standardisiert als XP, denn es schreibt keine spezifischen Entwicklungsvorgaben vor. Scrum ist allerdings standardisierter als Kanban, denn es schreibt Dinge wie Iterationen und funktionsübergreifende Teams vor.

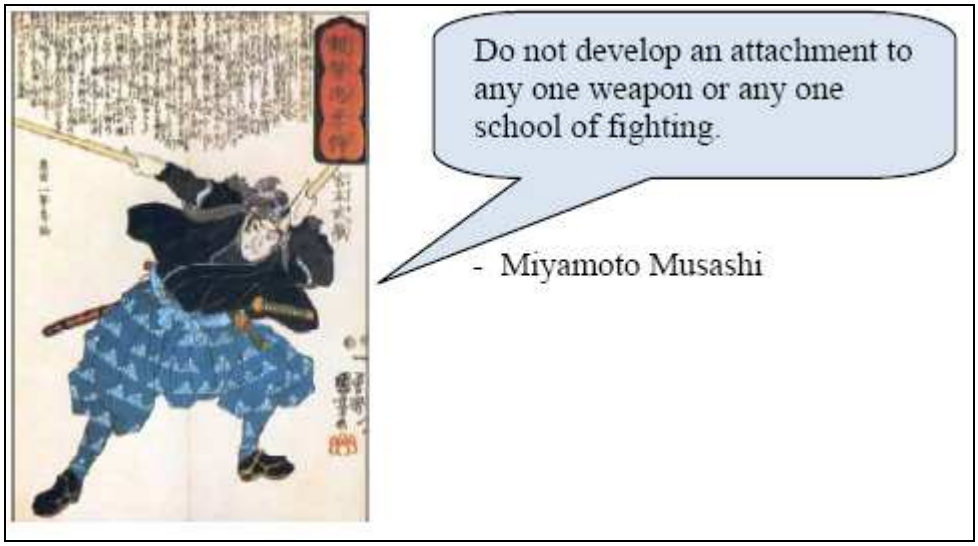
Ein Hauptunterschied zwischen Scrum und RUP ist, dass Sie in RUP zuviel bekommen, und Sie sollen das wegnehmen, was Sie nicht brauchen. In Scrum bekommen Sie zuwenig, und Sie sollen hinzufügen, was fehlt.

Kanban lässt fast alles offen. Die einzige Einschränkungen sind „Visualisieren Sie den Arbeitsablauf“ und „Limit Work In Progress – Begrenzen Sie der Arbeitspakete pro Arbeitsschritt“. Nur wenig entfernt von „Tu-was-auch-immer“, aber dennoch überraschend leistungsstark.

5.5 Begrenzen Sie sich nicht auf ein einziges Werkzeug!

Mischen und passen Sie die Werkzeuge an, wie Sie sie brauchen! Ich kann mir kaum ein erfolgreiches Scrum-Team vorstellen, das nicht die meisten Elemente von XP nutzt, zum Beispiel. Viele Kanban-Teams nutzen tägliche Stehungen (eine Scrumvorgabe). Einige Scrum-Teams schreiben einige ihrer Backlog-Items als Anwendungsfälle (Use Cases; eine RUP-Vorgabe) oder begrenzen die Anzahl ihrer Arbeitspakete (eine Kanbanvorgabe). Was auch immer für Sie funktioniert.

Miyamoto Musashi, ein Samurai aus dem 17. Jahrhundert, der für seine Kampftechnik mit dem Doppelschwert berühmt war, sagte es sehr hübsch:



Lass keine Verbundenheit mit irgendeiner Waffe oder irgendeiner Kampfschule entstehen

Achten Sie jedoch auf die Beschränkungen bei jedem Werkzeug. Wenn Sie beispielsweise Scrum nutzen und sich entscheiden, keine festen Zeitblöcke mehr für Iterationen vorzusehen (oder irgendeinen andern Aspekt von Scrum abzustellen), dann erklären Sie nicht, dass Sie Scrum einsetzen. Scrum ist minimalistisch genug, so wie es ist; wenn Sie Zeug rausnehmen und es immer noch Scrum nennen, dann wird das Wort bedeutungslos und verwirrt. Nennen Sie es etwa „Scrum-inspiriert“ oder „teilweise Scrum“ oder wie wäre es mit „Scrumisch“ :o)

6 Scrum schreibt Rollen vor

Scrum hat drei Standardrollen: Product-Owner (bestimmt Produktvision & Prioritäten), Team (implementiert das Produkt) und Scrum-Master (beseitigt Hindernisse und leitet den Prozess).

Kanban gibt überhaupt keine Rollen vor.

Das bedeutet nicht, dass Sie in Kanban keine Product-Owner-Rolle haben können oder sollen! Es bedeutet lediglich, dass Sie keine solche Rolle haben *müssen*.

Trotzdem seien Sie vorsichtig damit, Rollen hinzuzufügen; stellen Sie sicher, dass die zusätzlichen Rollen tatsächlich einen Mehrwert bringen und nicht mit anderen Elementen des Prozesses kollidieren. Sind Sie sicher, dass Sie diese Projektmanagerrolle brauchen? In einem großen Projekt ist es vielleicht eine tolle Idee, vielleicht ist das der Typ, der hilft, mehrere Teams & Product-Owner untereinander abzustimmen. In einem kleinen Projekt könnte die Rolle Überschuss sein, oder schlimmer, führt vielleicht zu suboptimalen Ergebnissen und Detailverliebtheit im Management.

Die grundlegende Denkart sowohl in Scrum als auch in Kanban ist “weniger ist mehr”. Also, im Zweifel fangen Sie mit weniger an.

Im Rest dieses Buchs werde ich den Ausdruck “Product-Owner” verwenden, um die Person zu benennen, die die Prioritäten eines Teams festlegt, unabhängig vom eingesetzten Prozess.

7 Scrum schreibt Iterationen in einem festen Zeitrahmen vor

Scrum basiert darauf, dass Iterationen in einem festgelegten Zeitrahmen erledigt werden. Sie können die Länge der Zeitfenster selbst wählen, aber die grundlegende Idee ist, dieselbe Iterationslänge über einen gewissen Zeitraum beizubehalten und so einen Rhythmus (eine Kadenz) einzuführen.

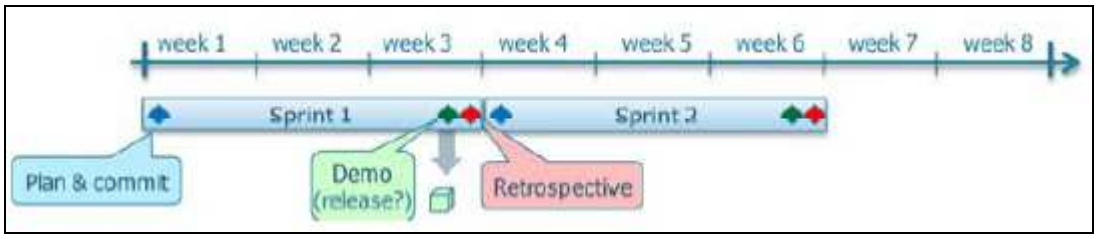
- Zu **Beginn einer Iteration:** Ein Iterationsplan wird erstellt, d. h. das Team stellt eine bestimmte Anzahl Items aus dem Produktbacklog zusammen, und zwar aufgrund der Prioritäten, die der Product-Owner festgelegt hat, und aufgrund der Einschätzung des Teams, wieviel es in einer Iteration fertigstellen kann.
- **Während einer Iteration:** Das Team konzentriert sich darauf, die Items fertig zu stellen, auf die es sich geeinigt hat. Der Umfang für diese Iteration ist unveränderlich.
- Am **Ende einer Iteration:** Das Team präsentiert den Beteiligten/Interessengruppen lauffähigen Code, idealerweise sollte dieser Code lieferfähig sein (d. h. getestet und fertig für die Auslieferung). Dann führt das Team eine Retrospektive durch, um ihr Vorgehen zu diskutieren und zu verbessern.

So gesehen ist eine Scrum-Iteration ein einzelner zeitbegrenzter Schritt, der drei verschiedenen Aktivitäten kombiniert: Planung, Vorgehensverbesserung und (idealerweise) Auslieferung der Software.

In Kanban sind zeitbegrenzte Iterationen nicht vorgeschrieben. Sie können selbst bestimmen, wann Sie planen, wann Sie das Vorgehen verbessern und wann Sie die Software ausliefern. Sie können bestimmen, diese Aktivitäten regelmäßig („Auslieferung jeden Montag“) oder bedarfsgerecht („ausliefern, wenn wir etwas Nützliches zum Ausliefern haben“) auszuführen.

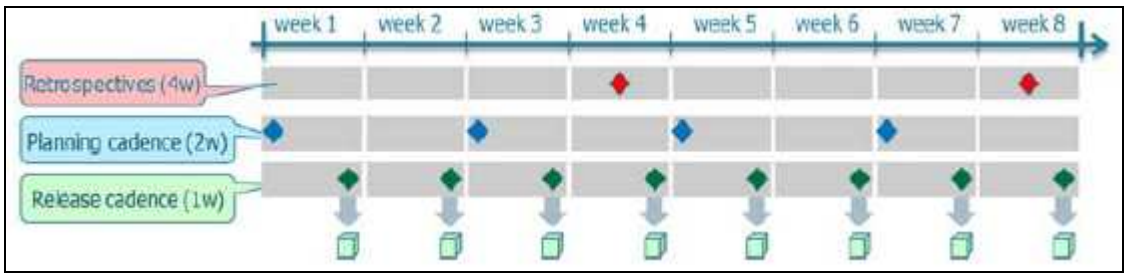
7.1 Team Nummer 1 (Einzelkadenz)

“Wir machen Scrum-Iterations”



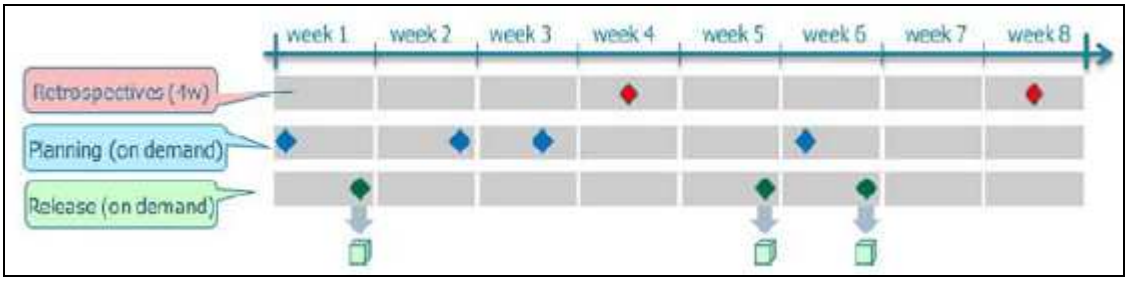
7.2 Team Nummer 2 (drei Kadenzen)

“Wir haben drei verschiedene Rhythmen. Jede Woche liefern wir aus, was gerade fertig ist für die Auslieferung. Jede zweite Woche haben wir eine Planungssitzung und aktualisieren die Prioritäten und Releasepläne. Jede vierte Woche haben wir eine Sitzung zur Retrospektive, um unser Vorgehen zu optimieren und zu verbessern.”



7.3 Team Nummer 3 (größtenteils ereignisgetrieben)

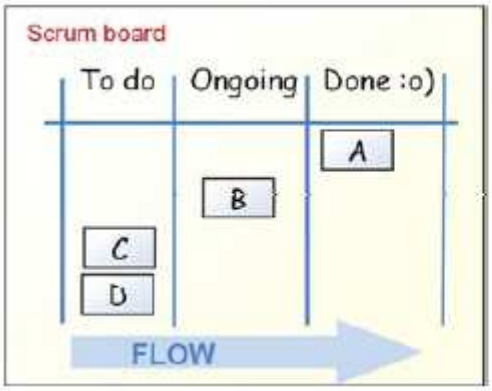
“Wir veranlassen eine Planungssitzung, wenn uns die Aufgaben ausgehen. Wir veranlassen ein Release, wenn ein Minimum an Marktreifen Komponenten (Minimum Marketable Features – MMFs) fertiggestellt ist. Wir veranlassen einen spontanen Qualitätszirkel, wenn wir zum zweiten Mal in dasselbe Problem laufen.“

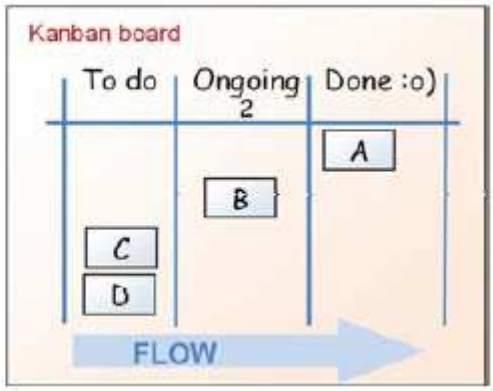


8 Kanban begrenzt WIP pro Zustand entlang des Arbeitsablaufs, Scrum begrenzt WIP pro Iteration

In Scrum zeigt das Sprint-Backlog, welche Aufgaben während einer gerade aktuell laufenden Iteration (= „Sprint“ in Scrumsprache) auszuführen sind. Im Allgemeinen wird dies durch Karten an einer Wand dargestellt, man nennt das Scrumboard oder Aufgabentafel.

Nun, was ist der Unterschied zwischen einem Scrumboard und einem Kanbanboard? Lassen Sie uns mit einem banal einfachen Projekt anfangen und die zwei vergleichen:





In beiden Fällen verfolgen wir ein Bündel einzelner Arbeitspakete, wie sie ihren Weg durch den Arbeitsablauf nehmen (flow). Wir haben hier drei Zustände ausgesucht: To do (zu tun), Ongoing (in Arbeit) und Done (fertig). Sie können die Zustände wählen, wie Sie mögen – einige Teams ergänzen das um Integrieren, Test, Release (Auslieferung) etc. Aber vergessen Sie nicht das *„Weniger ist mehr“*-Prinzip!

Also, was ist denn nun der Unterschied zwischen diesen beiden Beispielboards? Richtig – die kleine 2 in der mittleren Spalte auf dem Kanbanboard. Das ist alles. Die 2 bedeutet „es dürfen nicht mehr als 2 Arbeitspakete in dieser Spalte sein, und zwar in jedem beliebigen Augenblickt“.

In Scrum gibt es keine Regel, die das Team davor bewahrt, alle Arbeitspakete gleichzeitig in die „In Arbeit“-Spalte zu nehmen! Jedoch gibt es eine implizite Begrenzung, da die Iteration selbst einen begrenzten Rahmen hat. In diesem Fall liegt die implizite Grenze bei 4, da es auf dem ganzen Board nur 4 Arbeitspakete gibt. Also begrenzt Scrum WIP indirekt, während Kanban WIP direkt begrenzt.

Die meisten Scrumteams lernen nach und nach, dass es eine schlechte Idee ist, zu viele Arbeitspakete in der „In Arbeit“-Spalte zu haben, und entwickeln eine Kultur, bei der sie versuchen, die aktuellen Arbeitspakete fertig zu bekommen, bevor sie mit neuen Arbeitspaketen beginnen. Einige entscheiden sich sogar explizit, die Anzahl der Arbeitspakete, die in der „In Arbeit“-Spalte erlaubt sind, zu begrenzen, und dann – tadaaa! – das Scrumboard ist zum Kanbanboard geworden!

Also begrenzen beide, Scrum und Kanban, WIP (die Anzahl der Arbeitspakete, an denen gerade gearbeitet wird), aber auf unterschiedliche Weise. Scrumteams messen normalerweise die Schnelligkeit – wie viele Arbeitspakete (oder der zugehörigen Einheiten wie „Story Points“) pro Iteration fertiggestellt werden. Kennt ein Team einmal seine Schnelligkeit, wird das zu ihrem WIP-Limit (oder zumindest zu einer Richtlinie). Ein Team, das eine durchschnittliche Geschwindigkeit von 10 hat, wird normalerweise nicht mehr als 10 Arbeitspakete (oder Story Points) für einen Sprint heranziehen.

Also ist in Scrum *WIP begrenzt pro Zeiteinheit*.

In Kanban ist *WIP begrenzt pro Zustand entlang des Arbeitsablaufs*.

Im obigen Kanbanbeispiel dürfen zu jedem beliebigen Zeitpunkt maximal 2 Arbeitspakete im Arbeitsablauf im Zustand „In Arbeit“ sein, ohne Rücksicht auf die Rhythmuslänge. Sie müssen festlegen, welche Grenze in welchem Zustand des Arbeitsablaufs Anwendung finden soll, aber die grundlegende Idee ist, das WIP-Limit *aller* Zustände des Arbeitsablaufs zu setzen, beginnend so früh wie möglich und endend so spät wie möglich entlang des Nutzenstroms. Also sollten wir im obigen Beispiel darauf achten, ein WIP-Limit ebenfalls für den „In Arbeit“-Zustand hinzuzufügen (oder wie auch immer Sie Ihre Eingangsspalte nennen). Sobald wir WIP-Limits etabliert haben, können wir beginnen, die Durchlaufzeit zu messen und vorherzusagen, d. h. die durchschnittliche Zeit, die ein Arbeitspaket braucht, um sich den ganzen Weg über das Board zu bewegen. Haben wir vorhersagbare Durchlaufzeiten, erlaubt uns das, uns einem Leistungsvertrag (Service-Level-Agreement) zu verpflichten und realistische Auslieferungspläne zu machen.

Falls die Größen der Arbeitspakete dramatisch variieren, könnten Sie in Betracht ziehen, WIP-Limits stattdessen in Bezug auf Story-Points zu definieren, oder was für eine Einheit für Größe Sie auch immer nutzen. Einige Teams investieren Aufwand, um Arbeitspakete herunterzubrechen auf ungefähr gleiche Größen, um diese Arten von Abwägungen zu vermeiden und Zeit für Schätzungen von Dingen zu sparen (Sie könnten gar erwägen, dass Schätzungen überflüssig sind). Es ist einfacher, ein geschmeidig fließendes System zu erstellen, wenn Arbeitspakete annähernd gleich groß sind.

9 Beides sind Erfahrungswerte

Kapazität (Geschwindigkeit) | Durchlaufzeit | Qualität (Fehlerrate etc.) | Vorhersagbarkeit

Stellen Sie sich vor, es wären Drehknöpfe an diesen Instrumenten, und Sie könnten Ihr Vorgehen einfach durch Drehen der Knöpfe konfigurieren. „Ich will hohe Kapazität, kurze Durchlaufzeiten, hohe Qualität und hohe Vorhersagbarkeit. Also drehe ich demgemäß die Knöpfe auf 10, 1, 10, 10.“

Wäre das nicht großartig? Leider gibt es solche direkten Bedienelemente nicht. Nicht, dass ich wüsste zumindest. Lassen Sie mich wissen, wenn Sie welche finden.

Stattdessen haben wir ein Bündel indirekter Kontrollinstrumente.

Wenige/viele Leute | wenige große/viele kleine Teams | niedrige/ hohe WIP-Limits | keine/lange Iterationen | wenig/viel Planung …

Scrum und Kanban sind beide empirisch in dem Sinne, dass von Ihnen erwartet wird, mit dem Vorgehen zu experimentieren und es auf Ihre Umgebung anzupassen. In der Tat, Sie *müssen* experimentieren. Weder Scrum noch Kanban bringen alle Antworten mit – sie geben Ihnen nur eine einfache Menge von Einschränkungen, um Ihre eigenen Vorgehensverbesserungen betreiben zu können.

- Scrum sagt, Sie sollten funktionsübergreifende Teams haben. Also, wer soll in welchem Team sein? Weiß nicht, experimentieren Sie.
- Scrum sagt, das Team legt fest, wieviel Arbeit es in einen Sprint hinein nimmt. Also, wieviel sollten sie hinein nehmen? Weiß nicht, experimentieren Sie.
- Kanban sagt, Sie sollten WIP begrenzen. Also, was soll die Grenze sein? Weiß nicht, experimentieren Sie.

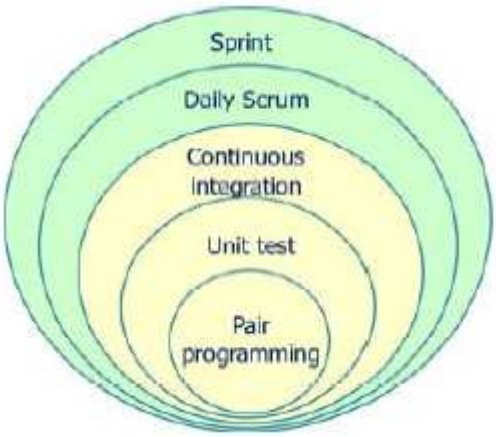
Wie ich früher schon erwähnt habe, macht Kanban weniger Einschränkungen als Scrum. Das bedeutet, Sie bekommen mehr Parameter, über die Sie nachdenken, mehr Drehknöpfe zum Drehen. Das kann sowohl ein Nachteil als auch ein Vorteil sein, abhängig von Ihrem Kontext. Wenn Sie die Konfigurationsdatei eines Softwarewerkzeugs öffnen, haben Sie zum Einstellen lieber 3 Optionen oder 100 Optionen? Wahrscheinlich irgendwas dazwischen. Hängt davon ab, wieviel Sie einstellen möchten und wie gut Sie das Werkzeug verstehen.

Also lassen Sie uns annehmen, wir reduzieren ein WIP-Limit auf Grund der Hypothese, dass das unser Vorgehen verbessern wird. Dann beobachten wir, wie Dinge wie Kapazität, Durchlaufzeit, Qualität und Vorhersagbarkeit sich verändern. Wir ziehen Schlüsse aus den Ergebnissen und ändern dann einige weitere Dinge, kontinuierlich unser Vorgehen verbessernd.

Es gibt viele Namen dafür. Kaizen (kontinuierlicher Verbesserungsprozess in Leansprache), Inspizieren & Adaptieren (Inspect & Adapt, Scrumsprache), empirische Prozesskontrolle (Empirical Process Control), oder warum nicht „Die wissenschaftliche Methodik“ (The Scientific Method).

Der kritischste Punkt dabei ist die *Rückkopplungsschleife* (feedback loop). Ändere etwas => Finde heraus, wie es lief => Lerne daraus => Ändere wieder etwas. Allgemein gesprochen, Sie möchten eine möglichst kurze Rückkopplungsschleife, so dass Sie ihr Vorgehen schnell adaptieren können.

In Scrum ist die zugrundeliegende Rückkopplungsschleife ein Sprint. Es gibt jedoch noch mehr, insbesondere wenn Sie Scrum mit XP (eXtreme programming) kombinieren:



Wenn es richtig gemacht wird, geben Scrum + XP Ihnen einen Haufen extrem wertvoller Rückkopplungsschleifen.

Die innere Rückkopplungsschleife, Pair-Programming, ist ein Rückkopplungszyklus von wenigen Sekunden. Ein Manko ist innerhalb von Sekunden gefunden und behoben („Hey, soll diese Variable nicht eine 3 sein?“). Das ist ein „Bauen wir das Zeug **richtig**?“-Rückkkopplungszyklus.

Die äußere Rückkopplungsschleife, der Sprint, ergibt einen Rückkkopplungszyklus von einigen Wochen. Das ist ein „Bauen wir das **richtige Zeugs**?“-Rückkopplungszyklus.

Was ist dann mit Kanban? Naja, zuallererst können Sie (und sollten Sie wahrscheinlich) alles aus den Rückkopplungsschleifen oben in Ihr Vorgehen aufnehmen, egal ob Sie Kanban nutzen oder nicht. Was Sie dann durch Kanban geschenkt bekommen, sind einige sehr nützliche Echtzeitmetriken:

- Durchschnittliche Durchlaufzeit. Jedes Mal aktualisiert, wenn ein Arbeitspaket die „Fertig“-Spalte erreicht (oder wie auch immer Sie Ihre Spalte am rechten Ende nennen).
- Engpässe. Ein typisches Symptom ist, dass die Spalte X vollgestopft mit Arbeitspaketen ist, während die Spalte X+1 leer ist. Achten Sie auf „Luftblasen“ auf Ihrem Board.

Das Schöne an Echtzeitmetriken ist, dass Sie sich die Länge ihrer Rückkopplungsschleife aussuchen können, basierend auf der Häufigkeit, in der Sie die Metriken analysieren und Veränderungen machen möchten. Zu lange Rückkopplungsschleifen bedeuten, Ihre Vorgehensverbesserung erfolgt langsam. Zu kurze Rückkopplungsschleifen bedeuten, Ihre Vorgehensverbesserung könnte keine Zeit haben, sich zwischen den einzelnen Änderungen zu stabilisieren, was zur Überlastung führen kann.

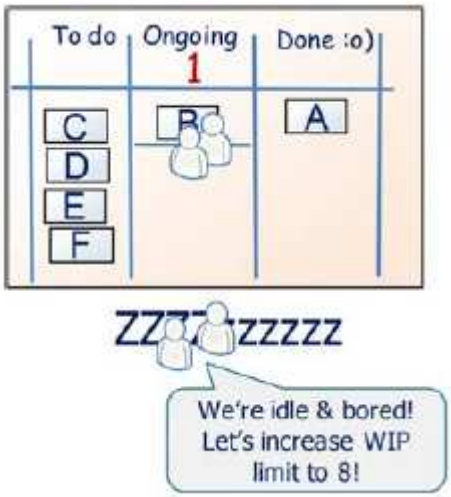
Eigentlich ist es so, dass die Länge der Rückkopplungsschleife selbst eine Sache ist, mit der Sie experimentieren können… so eine Art von Meta-Rückkopplungsschleife.

OK, Ich hör jetzt auf.

Beispiel: Mit WIP-Limits in Kanban experimentieren

Einer der typischen „Optimierungspunkte“ von Kanban ist das WIP-Limit. Und woher wissen wir, ob wir es richtig gesetzt haben?

Sagen wir mal, wir hätten ein 4-Personen-Team, und wir entscheiden uns, mit einem WIP-Limit von 1 anzufangen.

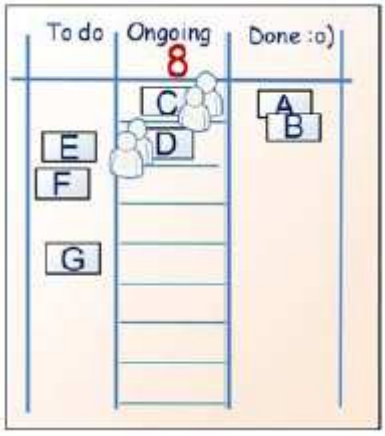


„Wir haben nichts zu tun und uns ist langweilig. Lass uns das WIP-Limit auf 8 erhöhen!“

Immer, wenn wir mit einem Arbeitspaket anfangen, können wir mit einem anderen neuen Arbeitspaket beginnen, bis das erste Arbeitspaket in der “Fertig”-Spalte ist. Also wird das wirklich schnell fertig.

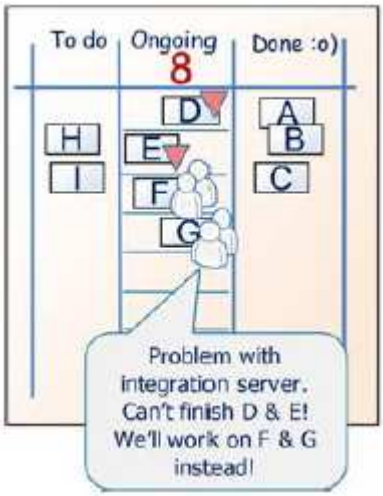
Super! Aber dann stellt sich heraus, dass es für alle 4 Leute normalerweise nicht machbar ist, am selben Arbeitspaket zu arbeiten (in diesem Beispielkontext), also sitzen Leute untätig herum. Wenn das nur ab und zu vorkommt, ist das kein Problem, aber wenn es regelmäßig passiert, ist die Konsequenz, dass die durchschnittliche Durchlaufzeit ansteigt. Grundsätzlich bedeutet ein WIP-Limit von 1, dass Arbeitspakete äußerst schnell durch die „In Arbeit“-Spalte hindurch kommen, sobald sie drin sind, aber sie bleiben länger als notwendig in der „In Arbeit“-Spalte stecken, so dass die Durchlaufzeit insgesamt über den kompletten Arbeitsablauf hinweg unnötig hoch sein wird.

Also, wenn das WIP-Limit von 1 zu niedrig war, wie sieht’s aus, wenn wir es auf 8 erhöhen?



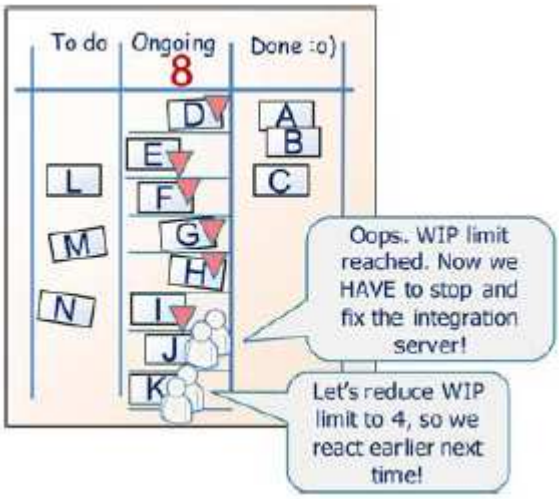
Das funktioniert für eine Weile besser. Wir entdecken, dass – im Durchschnitt – paarweises Arbeiten die Arbeit am schnellsten voranbringt. Damit haben wir zu jedem beliebigen Zeitpunkt bei einem 4-Personen-Team normalerweise 2 Arbeitspakete, an denen gerade gearbeitet wird. Das WIP-Limit von 8 ist nur eine Obergrenze, d. h. weniger Arbeitspakete im Bau zu haben passt!

Stellen Sie sich jetzt allerdings vor, dass wir mit dem Integrationsserver ein Problem bekommen, so dass wir Arbeitspakete nicht vollständig komplettieren können (unsere Definition von „Fertig“ beinhaltet die Integration). Diese Art von Dingen passiert manchmal, richtig?



„Probleme mit dem Integrationsserver. Wir können D & E nicht fertigstellen. Wir werden stattdessen an F & G arbeiten!“

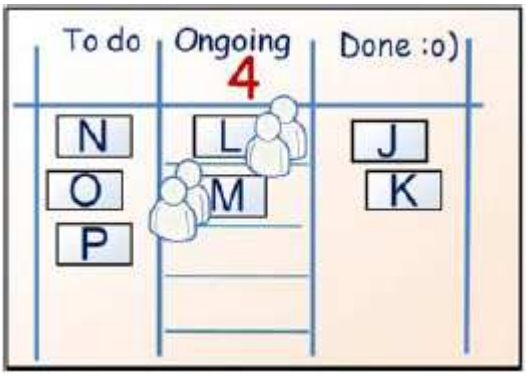
Da wir Arbeitspaket D oder E nicht fertigstellen können, beginnen wir mit der Arbeit an Arbeitspaket F. Wir können das auch nicht integrieren, also nehmen wir uns Arbeitspaket G vor. Nach einer Weile stoßen wir auf unsere Kanbangrenze – 8 Arbeitspakete in der “In Arbeit”-Spalte:



„Ups. WIP-Limit erreicht. Jetzt MÜSSEN wir anhalten und den Integrationsserver reparieren!“ – „Lass uns das WIP-Limit auf 4 reduzieren, so dass wir nächstes Mal früher reagieren!“

An diesem Punkt können wir keine Arbeitspakete mehr aufnehmen. Hey, wir sollten mal den verdammten Integrationsserver fixen! Das WIP-Limit hat uns dazu gebracht, zu reagieren und den Engpass zu beseitigen, statt einfach einen ganzen Haufen unfertiger Arbeit anzuhäufen.

Das ist gut. Aber wenn das WIP-Limit 4 wäre, hätten wir viel früher reagiert, wodurch wir eine bessere Durchlaufzeit erreicht hätten. Es ist also ein Ausbalancieren. Wir messen die durchschnittliche Durchlaufzeit und bleiben dran, unsere WIP-Limits zu optimieren, um die Durchlaufzeit zu optimieren:



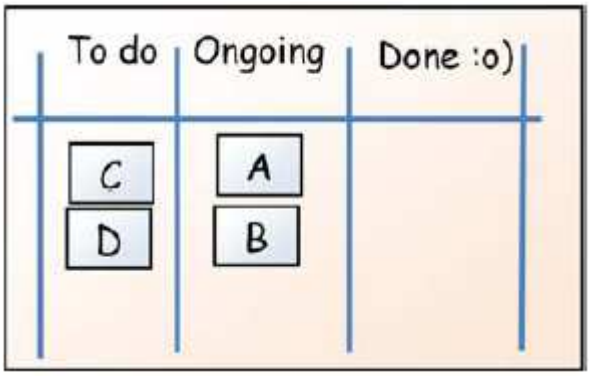
Nach einer Weile könnten wir herausfinden, dass sich Arbeitspakete in der „Zu tun“-Spalte aufstauen. Vielleicht ist es an der Zeit, dort auch ein WIP-Limit hinzuzufügen.

Warum brauchen wir überhaupt eine „Zu tun“-Spalte? Naja, wenn der Kunde immer verfügbar ist, um dem Team zu sagen, was als nächstes zu tun ist, wann immer es auch fragt, dann würde die „Zu tun“-Spalte nicht gebraucht. Aber in diesem Fall ist der Kunde manchmal nicht verfügbar, also gibt die „Zu tun“-Spalte dem Team einen kleinen Puffer, um in der Zwischenzeit von dort Arbeit zu holen.

Experimentieren Sie! Oder, wie der Scrumologe sagt, inspizieren & adaptieren Sie!

10 Scrum widersteht Änderungen innerhalb einer Iteration

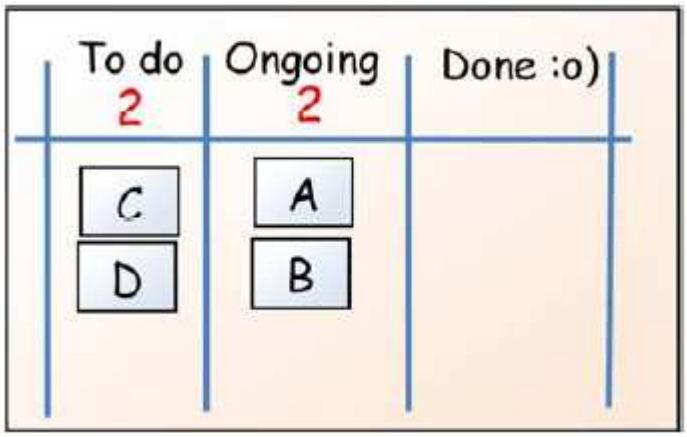
Sagen wir mal, unser Scrumboard sieht so aus:



Was, wenn jemand auftaucht und ein E ans Board hängen will?

Ein Scrumteam wird typischerweise etwas sagen wie „Nein, tut uns leid, wir haben uns darauf verständigt, dass wir A+B+C+D in diesem Sprint abarbeiten. Aber packen Sie das E ruhig ins Produktbacklog. Wenn der Product-Owner es mit einer hohen Priorität versieht, werden wir es in den nächsten Sprint hinein nehmen.“ Sprints der richtigen Länge geben dem Team gerade genug Zeitfenster, um etwas fertig zu stellen, während es dem Product-Owner trotzdem erlaubt ist, zu managen und Prioritäten regelmäßig zu aktualisieren.

Und was würde das Kanbanteam sagen?



Ein Kanbanteam könnte sagen „Packen Sie E in die „Zu tun“-Spalte. Aber die Grenze liegt für diese Spalte bei 2, also werden Sie in dem Fall C oder D herausnehmen müssen. Wir arbeiten gerade an A und B, aber sobald wir die Kapazität frei haben, werden wir uns das oberste Arbeitspaket aus der „Zu tun“-Spalte vornehmen.“

Somit folgt die Antwortzeit (wie lange es braucht, um auf eine Änderungen in den Prioritäten zu reagieren) eines Kanbanteams dem allgemeinen Prinzip „ein Arbeitspaket raus = ein Arbeitpaket rein“ (getrieben von den WIP-Limits).

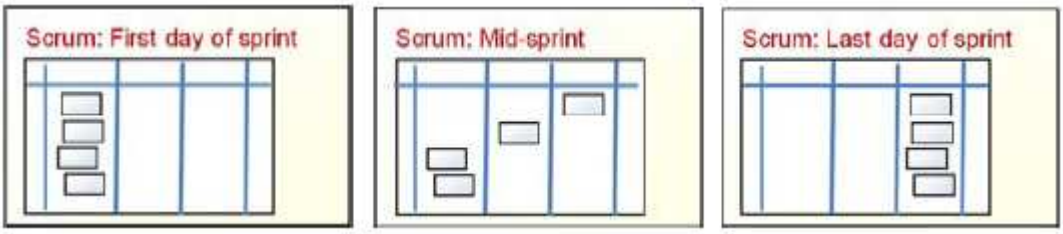
In Scrum beträgt die Antwortzeit im Durchschnitt eine halbe Sprintlänge.

In Scrum kann der Product-Owner das Scrumboard nicht anfassen, da das Team sich auf eine spezielle Menge von Arbeitspaketen für die Iteration festgelegt hat. In Kanban müssen Sie Ihre eigenen Grundregeln dafür einrichten, wer was auf dem Board ändern darf. Typischerweise bekommt der Product-Owner eine Art „Zu tun“- oder „Fertig“- oder „Backlog“- oder „Vorgeschlagen“-Spalte ganz links, wo er Änderungen machen kann, wann immer er mag.

Diese zwei Ansätze schließen sich jedoch nicht aus. Ein Scrumteam könnte entscheiden, einem Product-Owner zu erlauben, Prioritäten mitten im Sprint zu ändern (obwohl das normalerweise als Ausnahme angesehen wird). Und ein Kanbanteam könnte entscheiden, Einschränkungen darüber zu machen, wann Prioritäten geändert werden dürfen. Ein Kanbanteam könnte sogar entscheiden, zeitbegrenzte, streng selbstverpflichtete Iterationen zu nutzen, ganz wie in Scrum.

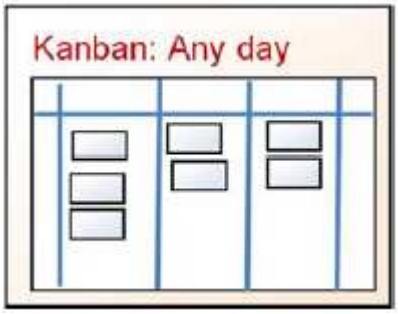
11 Ein Scrumboard wird zwischen zwei Iterationen geleert

Ein Scrumboard sieht während verschiedener Phasen eines Sprints typischerweise so aus.



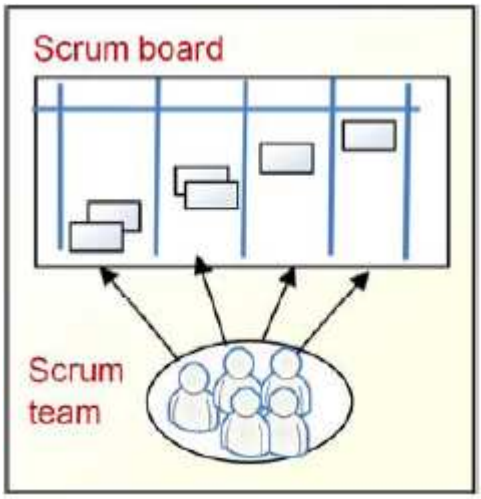
Wenn der Sprint vorbei ist, wird das Board leer geräumt – alle Arbeitspakete werden herunter genommen. Ein neuer Sprint beginnt und nach der Sprintplanungssitzung haben wir ein neues Scrumboard, mit neuen Arbeitspaketen in der Spalte ganz links. Technisch gesehen ist das Verschwendung, aber für erfahrene Scrumteams dauert das üblicherweise nicht zu lang, und das Durchführen des Tafelwischens kann einem ein angenehmes Gefühl von Vollendung und Abschluss geben. So wie das Geschirrspülen nach dem Abendessen – das zu tun ist mühsam, aber danach fühlt man sich gut.

In Kanban bleibt das Board normalerweise bestehen – Sie müssen es nicht zurückstellen auf Anfang und neu beginnen.



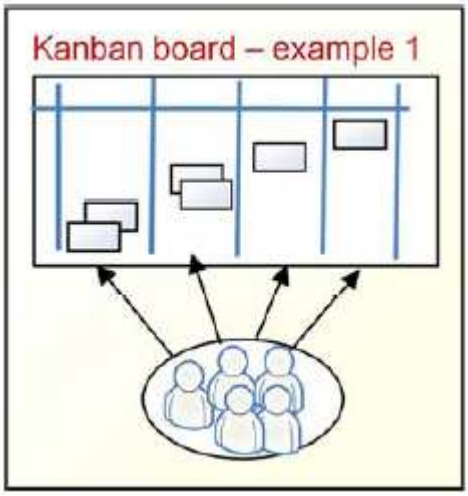
12 Scrum schreibt funktionsübergreifende Teams vor

Ein Scrumboard gehört genau einem Scrumteam. Ein Scrumteam ist funktionsübergreifend, es beinhaltet alle Fähigkeiten, die benötigt werden, um alle Arbeitspakete einer Iteration fertig zu stellen. Ein Scrumboard ist üblicherweise für jeden sichtbar, der sich dafür interessiert, aber nur das Scrumteam darf es editieren – es ist ihr Werkzeug, um ihre Selbstverpflichtung für diese Iteration zu managen.

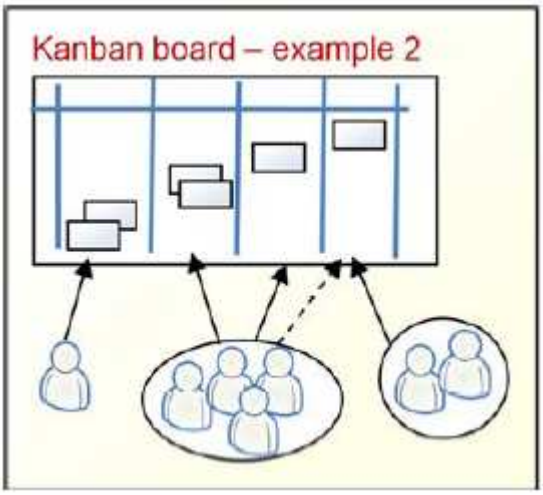


In Kanban sind funktionsübergreifende Teams optional, und ein Board muss nicht im Besitz eines speziellen Teams sein. Ein Board gehört zu einem Arbeitsablauf, nicht notwendigerweise zu einem Team.

Hier zwei Beispiele:



Beispiel 1: Das ganze Board ist einem funktionsübergreifendes Team zugestellt. Genau wie in Scrum.



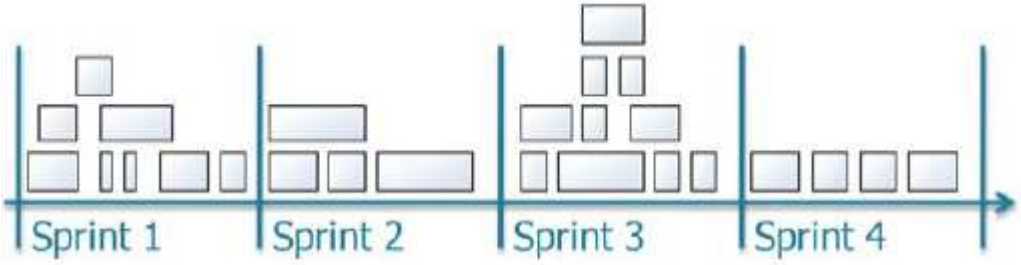
Beispiel 2: Der Product-Owner setzt Prioritäten in Spalte 1 fest. Ein funktionsübergreifendes Entwicklungsteam erledigt die Entwicklung (Spalte 2) und testet (Spalte 3). Auslieferung (Spalte 4) wird durch ein Spezialistenteam erledigt. Die Kompetenzen überlappen sich leicht, so dass einer der Entwickler helfen kann, wenn das Auslieferungsteam zum Engpass wird.

Also müssen Sie in Kanban einige Grundregeln dafür erstellen, wer das Board nutzt und wie, dann experimentieren Sie mit den Regeln, um den Arbeitsablauf zu optimieren.

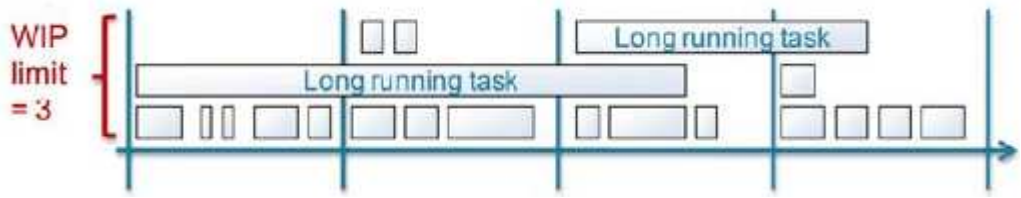
13 Scrum Arbeitspakete aus dem Backlog müssen in einen Sprint passen

Beide, Scrum und Kanban, basieren auf ständiger Entwicklung, d. h. die Arbeit wird in kleinere Stücke herunter gebrochen.

Ein Scrumteam wird sich nur auf Arbeitspakete verpflichten, von denen es denkt, dass es sie innerhalb einer Iteration (gemäß der Definition von „Fertig“) fertigstellen kann. Wenn ein Arbeitspaket zu groß ist, um in einen Sprint zu passen, werden das Team und der Product-Owner versuchen, Wege zu finden, es in kleinere Stücke aufzuteilen, bis es passt. Wenn Arbeitspakte dazu tendieren, groß zu sein, werden Iterationen länger (allerdings üblicherweise nicht länger als 4 Wochen).

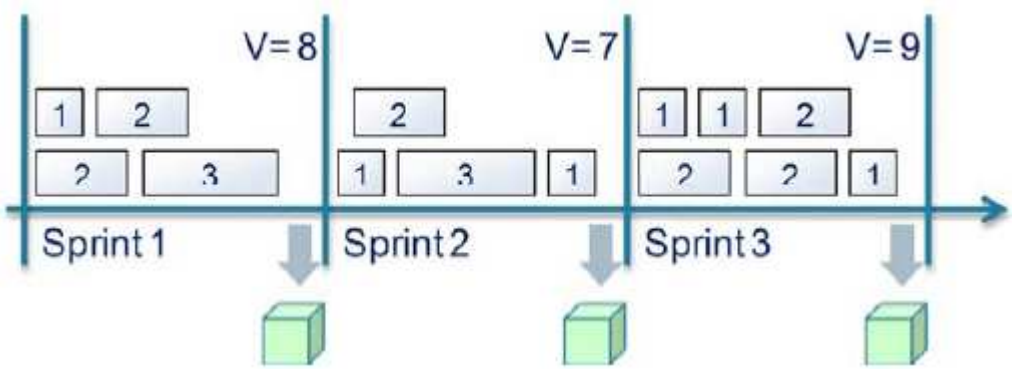


Kanbanteams versuchen, Durchlaufzeiten zu minimieren und den Arbeitsverlauf gleichmäßig zu halten, und das erschafft indirekt einen Anreiz, Arbeitspakete in relativ kleine Teile zu zerlegen. Aber es gibt keine explizite Regel, die besagt, dass Arbeitspakete klein genug sein müssen, um in einen speziellen Zeitrahmen zu passen. Auf demselben Board können wir ein Arbeitspaket haben, das einen Monat braucht, um es fertig zu stellen, und ein andere Arbeitspaket, das 1 Tag erfordert.



14 Scrum schreibt Schätzen und Geschwindigkeit vor

In Scrum wird von Teams erwartet, die relative Größe (= Arbeitsaufwand) für jedes Arbeitspaket, zu dessen Abarbeitung sie sich verpflichten, zu schätzen. Indem wir die Größe jedes am Ende eines Sprints fertig gestellten Arbeitspakets addieren, bekommen wir die Geschwindigkeit. Geschwindigkeit ist ein Maß für die Kapazität – wie viel Dinge wir pro Sprint liefern können. Hier ein Beispiel eines Teams mit einer Durchschnittsgeschwindigkeit von 8.



Zu wissen, dass die Durchschnittsgeschwindigkeit 8 ist, ist nett, denn dann können wir realistische Vorhersagen darüber machen, welche Arbeitspakete wir in den kommenden Sprints fertig stellen können, und können so realistische Auslieferungspläne machen.

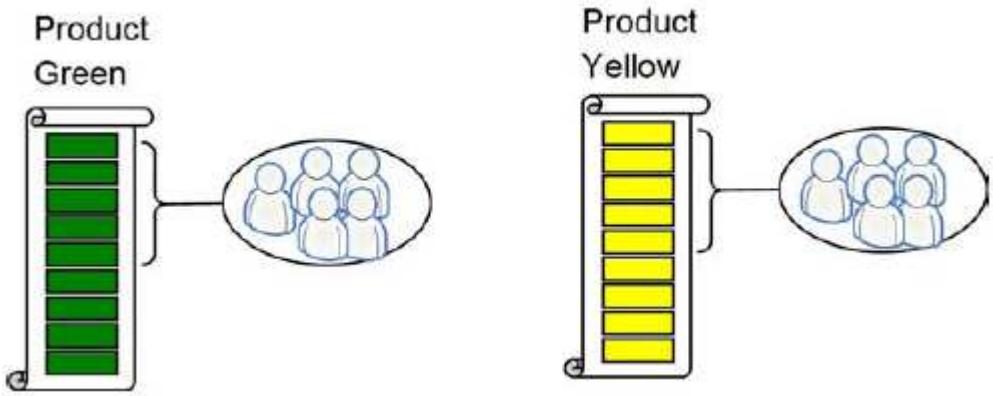
In Kanban ist Schätzen nicht vorgeschrieben. Wenn Sie also Verpflichtungen eingehen müssen, müssen Sie entscheiden, wie Vorhersagbarkeit sicherzustellen ist.

Einige Teams entscheiden sich dafür, Geschwindigkeit zu schätzen und zu messen, genau wie in Scrum. Andere Teams ziehen vor, das Schätzen wegzulassen, aber versuchen, jedes Arbeitspaket in etwa gleich große Teile zu zerlegen – dann können sie die Geschwindigkeit dahingehend messen, wieviele Arbeitspakete pro Zeiteinheit fertig gestellt waren (zum Beispiel Features pro Woche). Einige Teams gruppieren ihre Arbeitspakete in marktreifen Komponenten (Minimum Marketable Features – MMFs), messen die durchschnittliche Durchlaufzeit pro MMF und nutzen das, um den Leistungsvertrag (Service-Level-Agreements – SLAs) festzulegen – zum Beispiel „wenn wir uns einem MMF verpflichten, wird das immer innerhalb von 15 Tagen geliefert“.

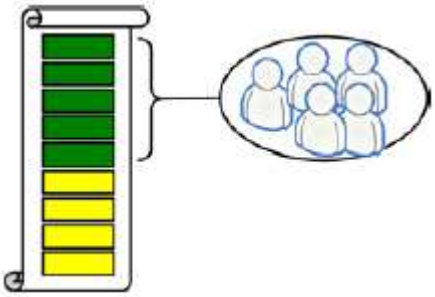
Es gibt alle Arten interessanter Techniken für Auslieferungsplanung und Verpflichtungsmanagement im Kanbanstil – aber keine bestimmte Technik ist vorgeschrieben, also los und weg mit Google und versuchen Sie andere Techniken, bis Sie die eine gefunden haben, die am besten zu Ihrem Kontext passt. Wir werden wahrscheinlich in der nächsten Zeit einige „Best Practices“ auftauchen sehen.

15 Beide erlauben, an mehreren Produkten gleichzeitig zu arbeiten

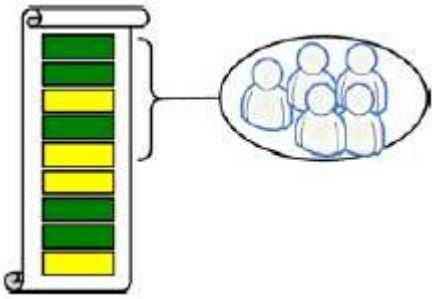
In Scrum ist das Produktbacklog ein eher unglücklich gewählter Name, da er impliziert, dass alle Arbeitspakete zum selben Produkt gehören müssen. Hier sind zwei Produkte, grün und gelb, jedes mit seinem eigenen Produktbacklog und seinem eigenen Team:



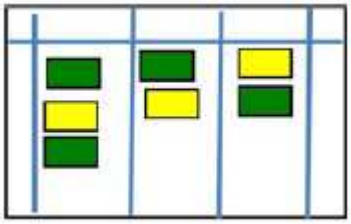
Was denn dann, wenn Sie nur ein Team haben? Nun ja, denken Sie sich das Produktbacklog eher als ein Teambacklog. Es listet die Prioritäten für künftige Iterationen für ein bestimmtes Team (oder eine Menge von Teams) auf. Wenn dieses Team also mehrere Produkte wartet, verschmelzen beide Produkte in einer Liste. Das zwingt uns dazu, zwischen den Produkten zu priorisieren, was in einigen Fällen nützlich ist. Es gibt einige Arten, das in der Praxis umzusetzen. Eine Strategie ist, sich das Team auf ein Produkt pro Sprint konzentrieren zu lassen:



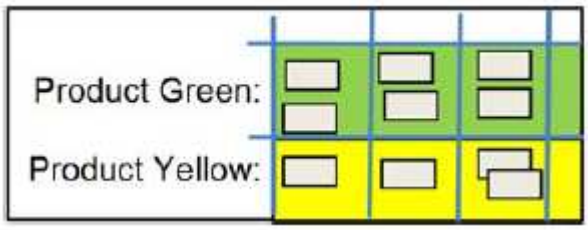
Eine andere Strategie ist, das Team in jedem Sprint an Komponenten für beide Produkte arbeiten zu lassen:



In Kanban ist es dasselbe. Wir können einige Produkte haben, die über dasselbe Board fließen. Wir könnten sie unterscheiden, indem wir verschieden farbige Karten verwenden:



... oder durch „Schwimmbahnen”:



Beide sind Lean und Agil

Ich werde hier nicht Lean Thinking und das agile Manifest durchnehmen, aber allgemein gesprochen stehen sowohl Scrum als auch Kanban in einer Linie mit jenen Werten und Prinzipien. Zum Beispiel:

- Scrum und Kanban sind beides Systeme zur Ablaufplanung, die sich auf das "Just-in-Time"-Prinzip aus der Lagerverwaltung des Leankonzepts beziehen. Das bedeutet, dass das Team sagt, wann und zu wieviel Arbeit es sich verpflichtet, die Teammitglieder "ziehen" Arbeit heran, wenn sie bereit sind, und bekommen sie nicht von außen „aufgedrückt". Wie ein Drucker das nächste Blatt erst zieht, wenn er bereit ist, es zu bedrucken (obwohl es einen kleinen & begrenzten Stapel von Papier gibt, von dem er es ziehen kann).
- Scrum und Kanban basieren auf einem kontinuierlichen und erfahrungsgestützten Verbesserungsprozess, was sich auf das Kaizen-Prinzip aus dem Leankonzept bezieht.
- Scrum und Kanban betonen das Eingehen auf Änderungen gegenüber dem bloßen Verfolgen eines Plans (obwohl Kanban typischerweise schnelleres Eingreifen erlaubt als Scrum), einer der vier Werte des agilen Manifests.

… und mehr.

Auf eine gewisse Weise kann Scrum als „nicht-so-lean" verstanden werden, da es vorschreibt, Arbeitspakete in zeitbegrenzte Iterationen hinein zu packen. Aber das hängt von der Länge Ihrer Iteration ab, und was Sie damit vergleichen. Verglichen mit einem traditionelleren Prozess, wo wir vielleicht 2 – 4 Mal pro Jahr etwas integrieren und ausliefern, ist das Produzieren von auslieferungsreifem Code alle 2 Wochen durch ein Scrumteam extrem schlank.

Aber dann, wenn Sie die Iterationen kürzer und kürzer machen, nähern Sie sich tatsächlich Kanban. Wenn Sie anfangen, darüber zu reden, die Iterationen kürzer als 1 Woche zu machen, könnten Sie in Erwägung ziehen, zeitbegrenzte Iterationen komplett zu verwerfen.

Ich habe es früher schon gesagt und ich höre nicht auf, es zu sagen: Experimentieren Sie, bis Sie etwas finden, das für Sie hinhaut! Und dann experimentieren Sie weiter :o)

16 Unwesentliche Unterschiede

Hier sind einige Unterschiede, die weniger relevant zu sein scheinen, verglichen mit den andern, die vorher erwähnt wurden. Trotzdem ist es gut, sich ihrer bewusst zu sein.

16.1 Scrum schreibt ein priorisiertes Produktbacklog vor

In Scrum wird Priorisierung immer durch Sortierung im Produktbacklog umgesetzt, und Änderungen der Priorität wirken sich im nächsten Sprint aus (nicht im aktuellen Sprint). In Kanban können Sie sich irgendein Priorisierungsschema aussuchen (oder sogar keins), und Änderungen wirken sich aus, sobald Kapazitäten frei werden (eher als zu festen Zeitpunkten). Es mag ein Produktbacklog geben oder nicht, und es mag oder mag nicht priorisiert sortiert sein.

In der Praxis macht das nur einen kleinen Unterschied. Auf einem Kanbanboard erfüllt die Spalte am linken Ende typischerweise denselben Zweck wie ein Scrum-Produktbacklog. Ob oder ob nicht die Liste nach Priorität sortiert ist, das Team braucht eine Art Entscheidungsregel, nach der es Arbeitspakete heranzieht. Beispiele für Entscheidungsregeln:

- Nimm immer das oberste Arbeitspaket.
- Nimm immer das älteste Arbeitspaket (dann hat jedes Arbeitspaket ein Erstelldatum).
- Nimm irgendein Arbeitspaket.
- Nimm ungefähr 20 % Wartungspakete und 80 % neue Features (bzgl. Aufwand).
- Teil die Teamkapazität grob gleichmäßig zwischen Produkt A und Produkt B auf.
- Nimm immer die roten Arbeitspakete zuerst, wenn es welche gibt.

In Scrum kann ein Produktbacklog also auf eine Kanban-ische Art eingesetzt werden. Wir können seine Größe begrenzen, und Entscheidungsregeln dafür aufstellen, wie es priorisiert werden soll.

16.2 In Scrum sind tägliche Stehungen vorgeschrieben

Ein Scrumteam hat jeden Tag zur selben Zeit & am selben Ort ein kurzes Treffen (maximal 15 Minuten). Der Zweck des Treffens ist, Informationen auszutauschen, was gerade ansteht, die Arbeit des Tages zu planen und signifikante Probleme zu identifizieren. Das nennt sich manchmal Daily-Standup (tägliche Stehungen), da es normalerweise im Stehen abläuft (um es kurz zu halten & einen hohen Energielevel zu erreichen).

Tägliche Stehungen sind in Kanban nicht vorgeschrieben, aber die meisten Kanbanteams scheinen es trotzdem zu tun. Es ist eine tolle Methode, ungeachtet dessen, welches Vorgehen sie einsetzen.

In Scrum ist der Ablauf eines Treffens personenbezogen – jede Person, eine nach der anderen, berichtet. Viele Kanbanteams setzen einen boardbezogenen Ablauf ein, konzentrieren sich auf Engpässe und andere sichtbaren Probleme. Dieser Ansatz ist besser skalierbar. Wenn Sie 4 Teams haben, die dasselbe Board benutzen und ihre täglichen Stehungen gemeinsam abhalten, müssen wir vielleicht nicht notwendigerweise Jeden sprechen lassen, solange wir uns auf die Engpässe am Board konzentrieren.

16.3 In Scrum sind Burndown-Diagramme vorgeschrieben

Geschätzte verbleibende Arbeit | Datum

Ein Burndown-Diagramm für einen Sprint zeigt, tagesaktuell, wieviel Arbeit in der aktuellen Iteration noch übrig bleibt.

Die Einheit der Y-Achse ist dieselbe Einheit, die für die Aufgaben in den Sprints eingesetzt wird. Typischerweise Stunden oder Tage (wenn das Team Arbeitspakete in Aufgaben aufteilt) oder Story-Points (wenn das Team das nicht tut). Es gibt allerdings einen Haufen Variationsmöglichkeiten hierfür.

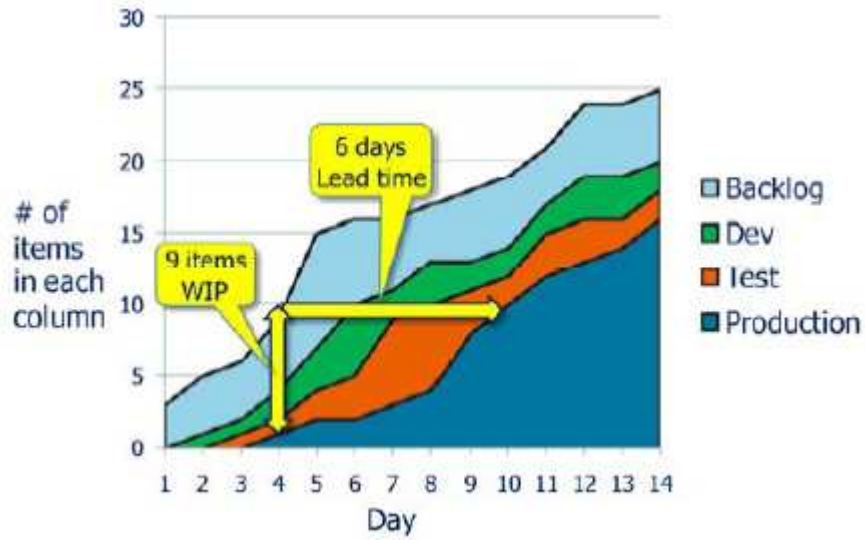
In Scrum werden Sprint-Burndown-Diagramme als eins der grundlegenden Werkzeuge eingesetzt, um den Fortschritt einer Iteration nachzuverfolgen.

Einige Teams setzen auch Release-Burndown-Diagramme ein, die denselben Aufbau haben, allerdings auf Ebene eines Releases – es zeigt typischerweise, wie viele Story-Points nach jedem Sprint im Produktbacklog noch übrig sind.

Der Hauptzweck eines Burndown-Diagramms ist, so früh wie möglich leicht herauszufinden, ob wir hinter dem Plan liegen, so dass wir ihn anpassen können.

In Kanban sind Burndown-Diagramme nicht vorgeschrieben. Vielmehr ist gar kein bestimmter Diagrammtyp vorgeschrieben. Aber Sie dürfen natürlich jede Art von Diagramm einsetzen, die Sie mögen (inkl. Burndown).

Hier ist ein Beispiel eines gestapelten Flächendiagramms. Diese Art von Diagramm illustriert ganz nett, wie geschmeidig Ihr Arbeitsablauf ist und wie WIP Ihre Durchlaufzeit beeinflusst.



Anzahl der Arbeitspakete pro Spalte pro Tag
Backlog | Entwicklung | Test | Produktiv

Jetzt kommt, wie's funktioniert. Jeden Tag zählen Sie die Gesamtzahl der Arbeitspakete in jeder Spalte des Kanbanboards zusammen und stapeln diese auf der Y-Achse. Am 4. Tag waren 9 Arbeitspakete am Board. Von der Spalte ganz rechts aus gab es 1 Arbeitspaket in "Production", 1 Arbeitspaket in "Test", 2 Arbeitspakete in "Dev" (Entwicklung) und 5 Arbeitspakete im Backlog. Wenn wir diese Zahlen jeden Tag aufzeichnen und die Punkte verbinden, bekommen wir ein hübsches Diagramm wie das oben. Die vertikalen und horizontalen Pfeile illustrieren die Beziehung zwischen WIP und Durchlaufzeit.

Der horizontale Pfeil zeigt uns, dass Arbeitspakete, die am 4. Tag zum Backlog hinzugefügt wurden, durchschnittlich 6 Tage brauchten, um Produktionsreife zu erreichen. Ungefähr die halbe Zeit davon dauerte der Test. Wir können sehen, dass wir – würden wir WIP im Test und Backlog begrenzen – unsere Gesamtdurchlaufzeit signifikant reduzieren könnten.

Die dunkelblaue Fläche zeigt uns die Schnelligkeit (d. h. die Anzahl von Arbeitspaketen, die pro Tag eingespielt werden). Mit der Zeit können wir sehen, wie höhere Geschwindigkeit die Durchlaufzeit reduziert, während höhere WIP-Limits die Durchlaufzeit erhöhen.

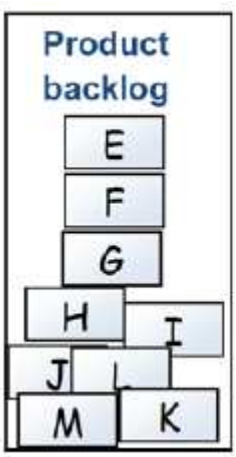
Die meisten Organisationen möchten Dinge schneller erledigen (= Durchlaufzeit reduzieren). Leider laufen viele in die Falle anzunehmen, dass das bedeutet, mehr Leute einzubeziehen oder Überstunden zu fahren. Normalerweise ist die effektivste Art, um Dinge schneller zu erledigen, den Arbeitsablauf gleichmäßiger zu machen und die Arbeit auf die Kapazität zu begrenzen, *nicht aber* mehr Leute hinzuzuziehen oder härter zu arbeiten. Dieser Diagrammtyp zeigt, warum, und erhöht so die Wahrscheinlichkeit, dass das Team und das Management effektiv zusammenarbeiten werden.

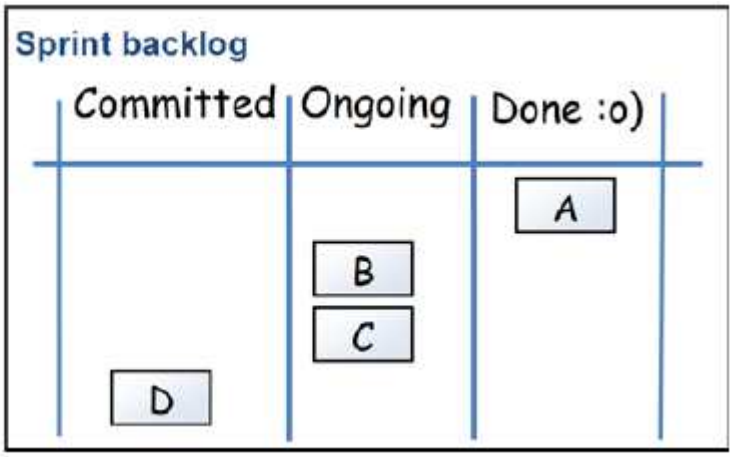
Es wird noch klarer, wenn wir zwischen den Warteschlangenzuständen (wie „wartet auf Test") und den Arbeitszuständen (wie „im Test") unterscheiden. Wir möchten die Anzahl der Arbeitspakete, die in den Warteschlangen liegen, absolut minimieren, und ein gestapeltes Flächendiagramm hilft, die richtigen Anreize dafür zu schaffen.

17 Scrumboard vs. Kanbanboard – ein weniger triviales Beispiel

In Scrum ist das Sprintbacklog nur ein Teil des Bildes – der Teil, der zeigt, was das Team während eines laufenden Sprints tut. Der andere Teil ist das Produktbacklog – die Liste von Sachen, die der Product-Owner in künftigen Sprints erledigt haben möchte.

Der Product-Owner kann das Sprintbacklog sehen, aber nicht anfassen. Er kann jederzeit das Produktbacklog ändern, wie er will, aber die Änderungen haben bis zum nächsten Sprint keinen Effekt (d. h. ändern nicht, was gerade abgearbeitet wird).

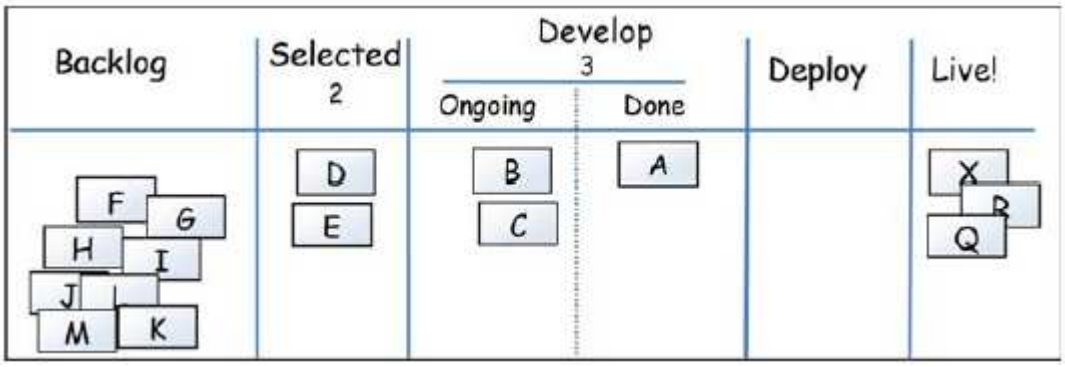




Produktbacklog Sprintbacklog: Abgestimmt | In Arbeit | Fertig

Sobald der Sprint erledigt ist, “liefert” das Team dem Product-Owner “potenziell auslieferbaren Code”. Also beendet das Team den Sprint, macht eine Sprintbesprechung und demonstriert dem Product-Owner stolz die Features A, B, C und D. Der Product-Owner kann jetzt entscheiden, ob er das ausliefern möchte oder nicht. Der letzte Teil – tatsächlich das Produkt auszuliefern – ist normalerweise nicht im Sprint selbst enthalten und deswegen nicht im Sprintbacklog sichtbar.

In diesem Szenario könnte ein Kanbanboard stattdessen irgendwie wie dies aussehen:



Backlog | Ausgewählt (2) | Entwicklung mit „In Arbeit“ und „Fertig“ (3) | Ausliefern | Produktiv!

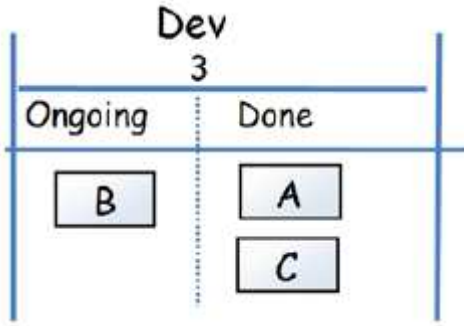
Jetzt ist der gesamte Arbeitsablauf auf demselben Board – wir betrachten nicht nur das, was ein Scrumteam in einer Iteration gerade tut.

Im Beispiel oben ist die „Backlog“-Spalte einfach eine allgemeine Wunschliste, ohne bestimmte Reihenfolge. Die „Ausgewählt“-Spalte („Selected“) enthält die Arbeitspakete von hoher Priorität mit einem Kanbanlimit von 2. Also dürfen nur 2 hoch priorisierte Arbeitspakete zu jedem gegebenen Zeitpunkt dort drin sein. Wann immer das Team bereit ist, an einem neuen Arbeitspaket zu arbeiten, nimmt es sich das oberste Arbeitspaket aus „Ausgewählt“. Der Product-Owner kann Änderungen in den Spalten „Backlog“ und „Ausgewählt“ machen, wann immer er möchte, aber nicht in anderen Spalten.

Die „Entwicklung“-Spalte („Develop“), die in zwei Unterspalten aufgesplittet ist, zeigt, was aktuell in der Entwicklung ist, mit einem Kanbanlimit von 3. In Netzwerktechnik ausgedrückt, entspricht das Kanbanlimit der „Bandbreite“ und die Durchlaufzeit entspricht dem „ping“ (oder Antwortzeit).

Warum haben wir die „Entwicklung“-Spalte in zwei Unterspalten „In Arbeit“ und „Fertig“ gesplittet? Das ist so, um dem Produktionsteam die Chance zu geben zu wissen, welche Arbeitspakete sie für das Produktivrelease heranziehen können.

Das „Entwicklung“-Limit von 3 gilt für die beiden Unterspalten zusammen. Warum? Sagen wir mal, es gibt 2 Arbeitspakete in „Fertig“:

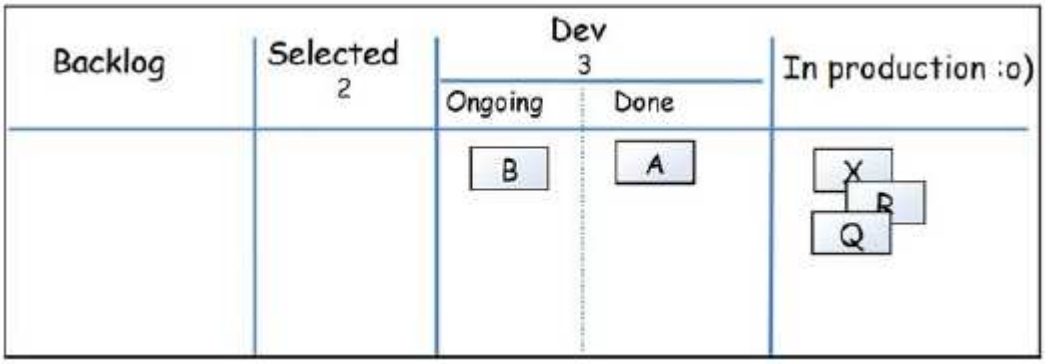


Entwicklung mit „In Arbeit“ und „Fertig“ (3)

Das bedeutet, es kann nur ein Arbeitspaket in „In Arbeit“ geben. Das bedeutet, es wird eine Überkapazität geben, Entwickler, die ein neues Arbeitspaket beginnen *könnten*, die es aber nicht dürfen wegen des Kanbanlimits. Das gibt ihnen einen starken Anreiz, ihren Aufwand zu fokussieren und zu helfen, Dinge in die Produktion („Deploy“) zu bringen, um die „Fertig“-Spalte zu leeren und den Fluss zu maximieren. Dieser Effekt ist nett und geht Schritt für Schritt voran – je mehr Zeugs in „Fertig“ ist, desto weniger Zeugs darf in „In Arbeit“ sein – was dem Team hilft, sich auf die richtigen Dinge zu konzentrieren.

17.1 Einzelstückmaterialfluss

Ein „Einzelstückmaterialfluss“ ist eine Art Szenario des „perfekten Durchflusses“, wobei ein Arbeitspaket über das Board wandert ohne jemals in einer Warteschlange zu stecken. Das bedeutet, dass zu jedem Zeitpunkt jemand an diesem Arbeitspaket arbeitet. Hier sieht man, wie das Board in diesem Fall aussehen könnte:



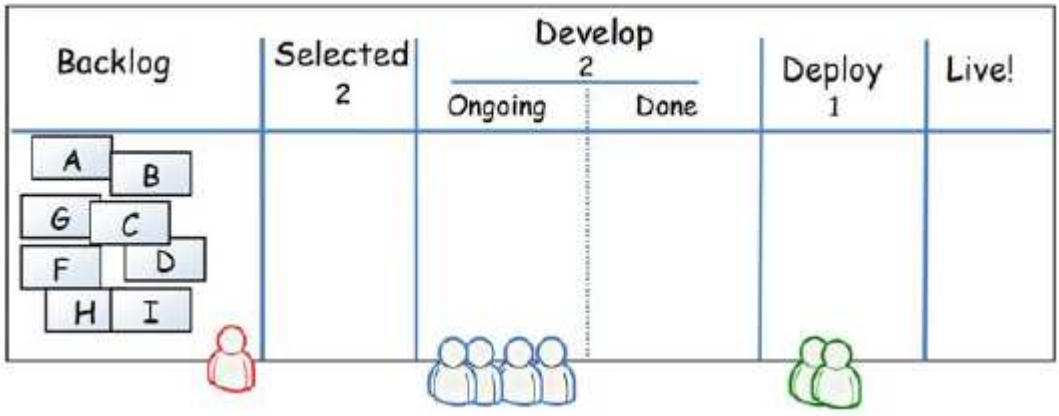
Backlog | Ausgewählt (2) | Entwicklung mit „In Arbeit“ und „Fertig“ (3) | Im Produktivsystem :o)

B wird im Moment implementiert, A wird im Moment in das Produktivsystem eingespielt. Wann immer das Team für das nächste Arbeitspaket bereit ist, fragen sie den Product-Owner, was das nächstwichtige ist und bekommen sofort eine Antwort. Wenn dieses ideale Szenario fortbesteht, können wir die beiden Warteschlangen „Backlog“ und „Ausgewählt“ wegschmeißen und bekommen eine *richtig* kurze Durchlaufzeit!

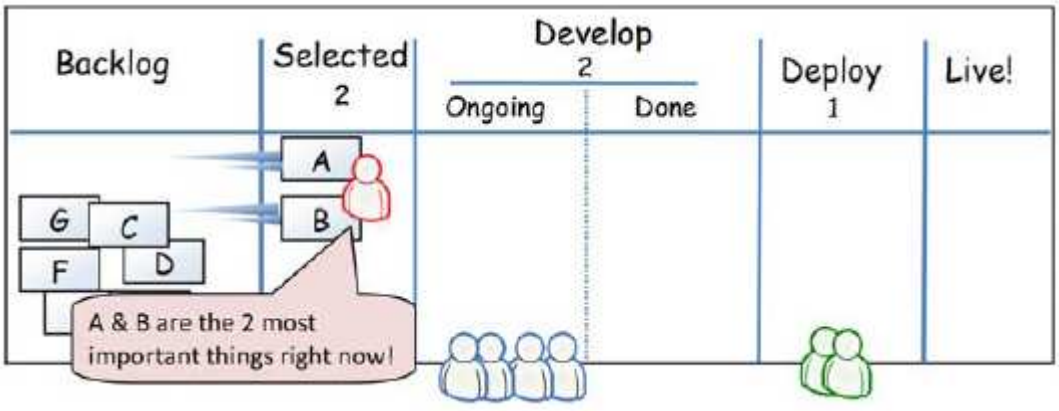
Cory Ladas drückt das nett aus: „Der ideale Arbeitsplanprozess sollte dem Entwicklungsteam immer das am besten als nächstes zu bearbeitende Ding liefern, nicht mehr und nicht weniger.“

Die WIP-Limits sind dafür da, bei Problemen davor zu bewahren, außer Kontrolle zu geraten, so dass wenn Dinge reibungslos ablaufen, die WIP-Limits gar nicht zum Einsatz kommen.

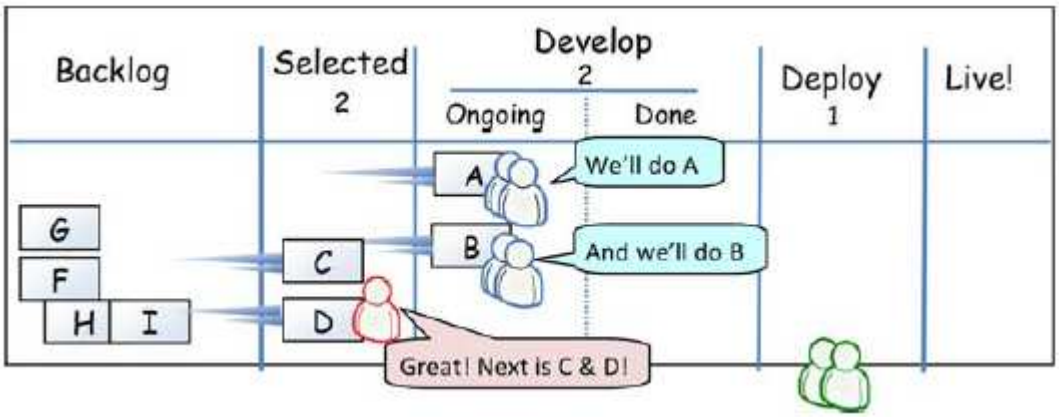
17.2 Ein Tag im Kanbanland



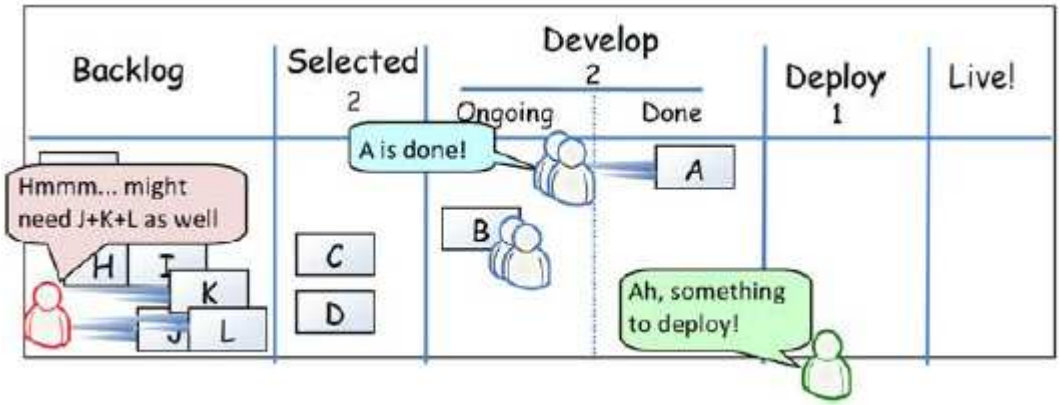
Backlog | Ausgewählt (2) | Entwickeln (2) mit „In Arbeit“ und „Fertig“ | Bereitstellen (1) | Produktivrelease



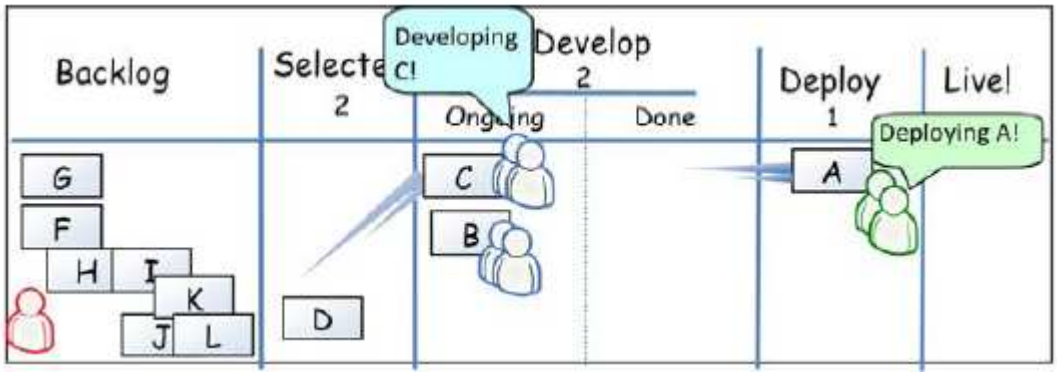
„A & B sind gerade die zwei wichtigsten Dinge“



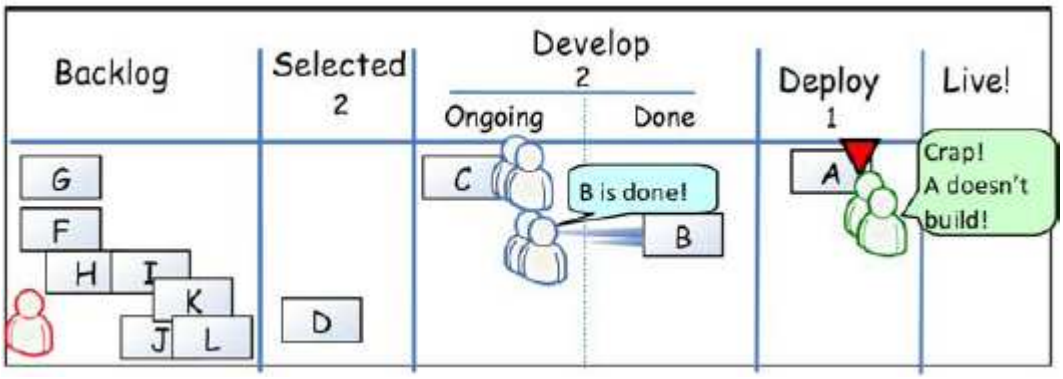
„Klasse! Als nächstes C und D!“ – „Wir machen A“ – „Und wir machen B“



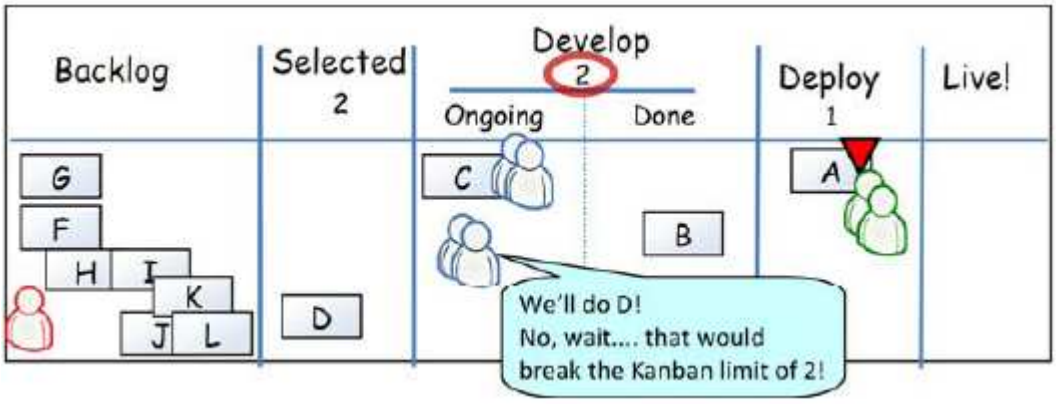
„Hmmm… ich könnte J+K+L auch brauchen“ – „A ist fertig“ – „Ah, etwas zum einspielen!“



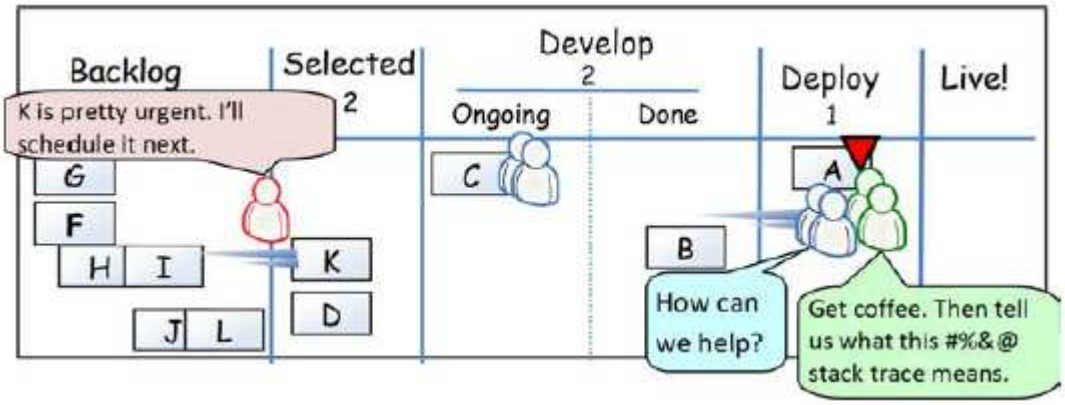
„Entwickeln C!“ – „Spiele A ein!“



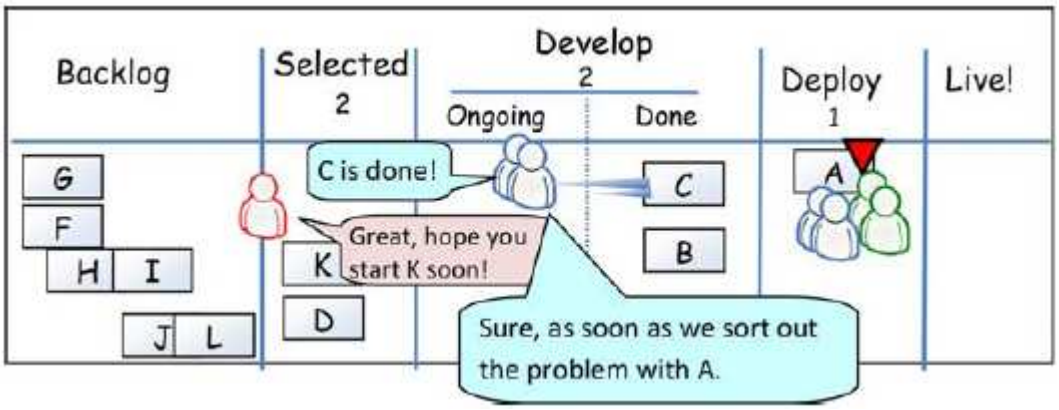
„B ist fertig!“ – „Mist! A lässt sich nicht einspielen!“



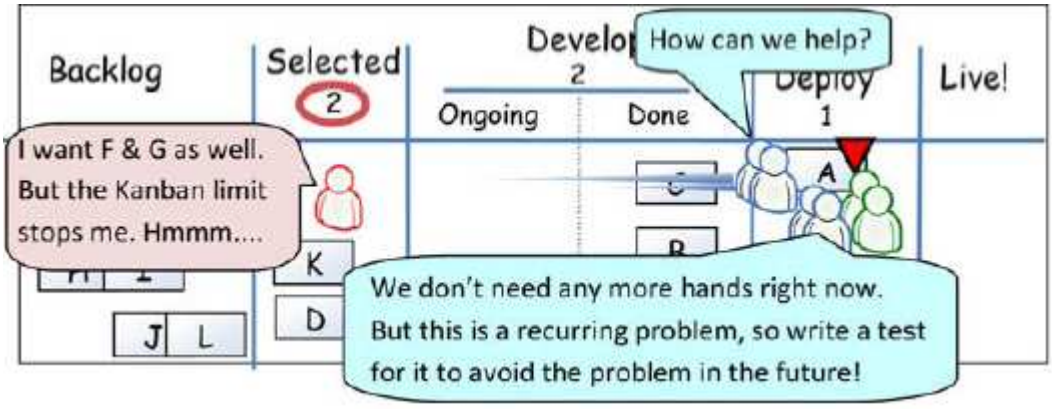
„Wir werden D machen! Nein, warte… das würde das Kanbanlimit von 2 verletzen!“



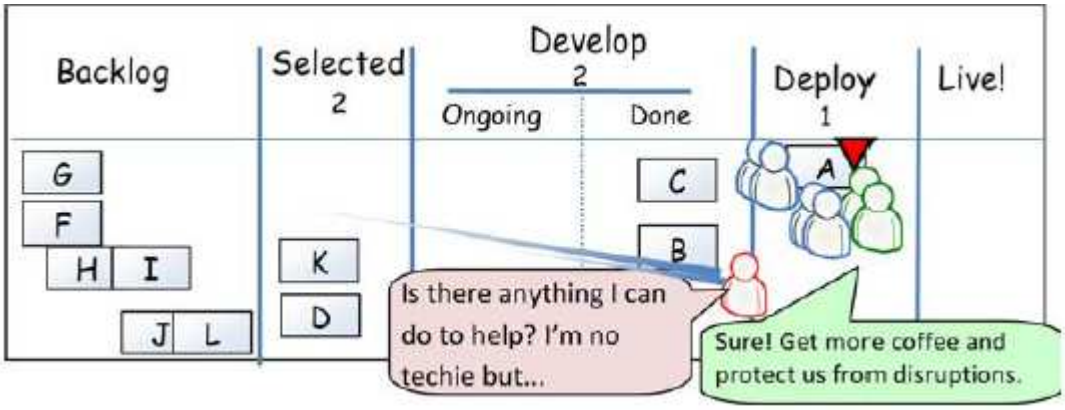
„K ist ziemlich dringend. Ich werde es als nächstes einordnen." – „Wie können wir helfen?" – „Holt Kaffee. Dann erzählt uns, was diese #%&@ Stapelverfolgung bedeutet."



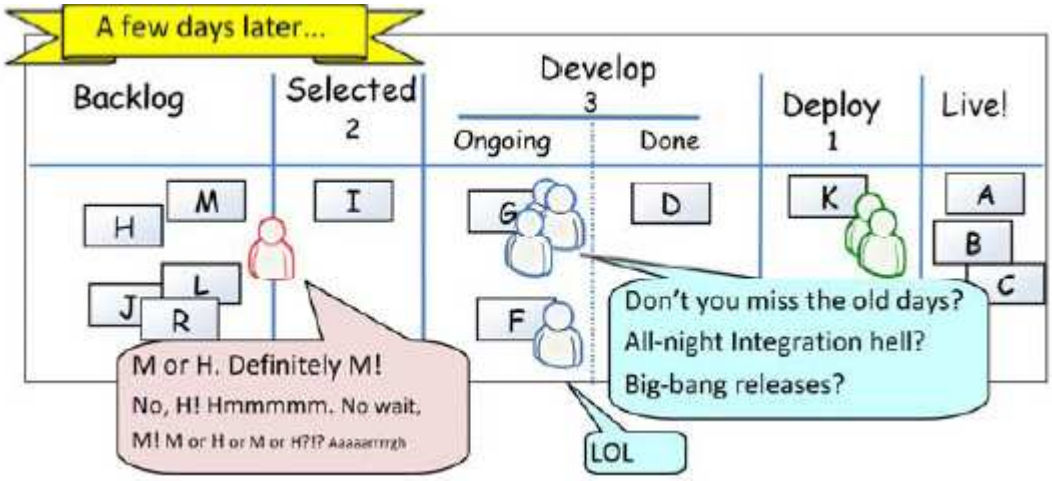
„C ist fertig" – „Toll, ich hoffe, Ihr fangt bald mit K an!" – „Sicher, sobald wir das Problem mit A ausbaldowert haben."



„Ich möchte F & G auch haben. Aber das Kanbanlimit stoppt mich. Hmmm…" – „Wie können wir helfen?" – „Wir brauchen jetzt gerade nicht mehr mehr Hände. Aber das ist ein wiederkehrendes Problem, also schreibt einen Test, um das Problem in Zukunft zu vermeiden!"



„Gibt es irgendwas, das ich tun kann, um zu helfen? Ich bin kein Tekkie, aber…“ – „Sicher! Bring mehr Kaffee und halte uns den Rücken frei.“



Einige Tage später…

„M oder H. Definitiv M! Nein, H! Hmmmmm. Nein, warte, M! M oder H oder M oder H?!? Aaaaaaahhh“ – „Vermisst Ihr nicht die alten Zeiten? Nächtelange Integrationshölle? Urknallsoftwareversionen?“ – „LOL“ (laughing out loud)

17.3 Muss das Kanbanboard wie das hier aussehen?

Nein, das Board oben war einfach ein Beispiel!

Das Einzige, was Kanban vorschreibt ist, dass der Arbeitsablauf visualisiert und dass die Arbeit, die gleichzeitig erledigt wird, begrenzt werden soll. Der Zweck ist, einen gleichmäßigen Fluss durch das System zu erzeugen und die Durchlaufzeit zu minimieren. Also müssen Sie regelmäßig Fragen wie die folgenden auf den Tisch bringen:

Welche Spalten sollten wir haben?

Jede Spalte repräsentiert einen Zustand im Arbeitsablauf oder eine Warteschlange (Puffer) zwischen zwei Arbeitsablaufzuständen. Fangen Sie simpel an und fügen Sie Spalten hinzu, wenn es erforderlich ist.

Wie sollten die Kanbanlimits aussehen?

Wenn das Kanbanlimit für „Ihre“ Spalte erreicht ist und Sie nichts zu tun haben, fangen Sie an, sich auf die Suche nach einem Engpass im weiteren Ablauf zu machen (d. h. Arbeitspakete stapeln sich rechts von ihrer Spalte am Board) und helfen Sie, den Engpass zu beheben. Falls es keinen Engpass gibt, ist das ein Zeichen dafür, dass das Kanbanlimit zu niedrig sein könnte, da der Grund für das Limit war, das Risiko zu senken, Engpässe im weiteren Verlauf mit Arbeitspaketen zuzuschütten.

Falls Sie bemerken, dass viele Arbeitspakete lange Zeit stillstehen, ohne dass an ihnen gearbeitet wird, ist das ein Anzeichen dafür, dass das Kanbanlimit zu hoch sein könnte.

- Zu niedriges Kanbanlimit => untätige Leute => schlechte Arbeitsergebnisse
- Zu hohes Kanbanlimit => ruhende Aufgaben => schlechte Durchlaufzeit

Wie streng sind die Kanbanlimits?

Einige Teams behandeln sie wie strikte Regeln (d. h. das Team darf ein Limit nicht überschreiten), einige Teams behandeln sie wie Richtlinien oder Diskussionsauslöser (d. h. das Brechen eines Kanbanlimits ist erlaubt, es sollte aber eine bewusste Entscheidung mit einem konkreten Grund geben). Also nochmal, Sie entscheiden selbst. Ich habe Ihnen gesagt, Kanban schreibt nicht sehr viel vor, richtig?

18 Zusammenfassung von Scrum vs. Kanban

18.1 Ähnlichkeiten

- Beide sind Lean und agil.
- Beide setzen das Pull-Prinzip ein.
- Beide begrenzen WIP.
- Beide nutzen Transparenz, um die Prozessverbesserung anzustoßen.
- Beide konzentrieren sich darauf, schnell und oft ausführbare Software auszuliefern.
- Beide basieren auf selbstorganisierten Teams.
- Beide erfordern es, die Arbeit aufzuteilen.
- Bei beiden wird der Releaseplan auf Basis empirischer Daten (Geschwindigkeit und Durchlaufzeit) kontinuierlich verbessert.

18.2 Unterschiede

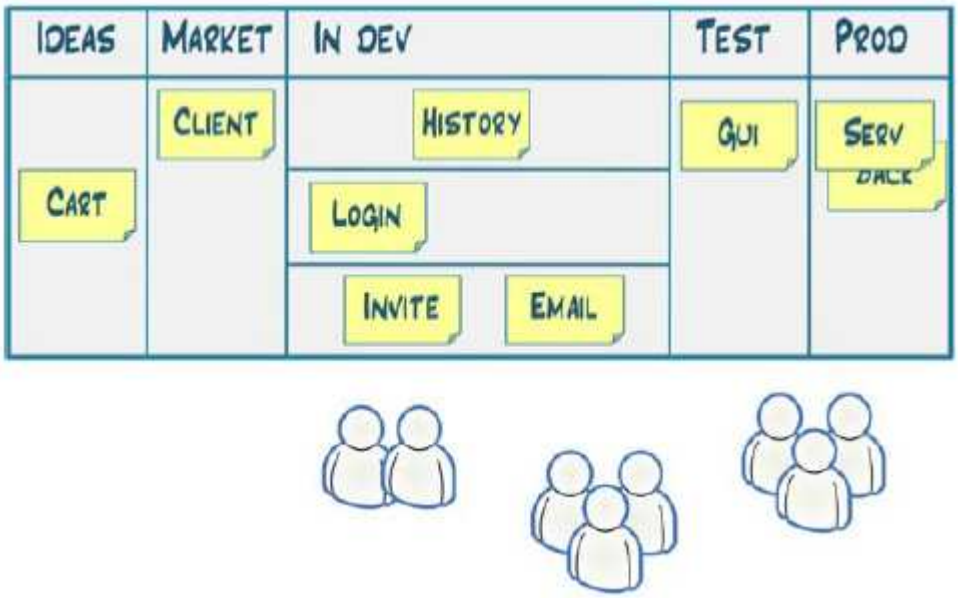
Scrum	Kanban
Iterationen mit festem Zeitschema vorgeschrieben.	**Iterationen mit festem Zeitschema sind optional.** Kann verschiedene Rhythmen (Kadenzen) für Planung, Release und Prozessverbesserung haben Kann ereignisgetrieben sein statt zeitgetrieben
Team verpflichtet sich zu einer bestimmten Menge an Abarbeitung für diese Iteration	**Verpflichtung optional**
Setzt **Geschwindigkeit** als Standardmetrik für Planung und Prozessverbesserung ein	Setzt **Durchlaufzeit** als Standardmetrik für Planung und Prozessverbesserung ein
Funktionsübergreifende Teams vorgeschrieben	Funktionsübergreifende Teams optional. **Spezialisierte Teams erlaubt.**
Arbeitspakete müssen aufgeteilt werden, so dass sie innerhalb eines Sprints abgearbeitet werden können.	Keine bestimmte Arbeitspaketgröße vorgeschrieben.
Burndown-Diagramm vorgeschrieben	Kein spezieller Diagrammtyp vorgeschrieben
WIP-Limit indirekt (durch Sprints)	**WIP-Limit direkt** (pro Zustand im Arbeitsablauf)
Schätzen vorgeschrieben	**Schätzen optional**
Kann keine Arbeitspakete in laufenden Iterationen hinzufügen	**Kann neue Arbeitspakete hinzufügen, wann immer Kapazität verfügbar ist**
Ein **Sprintbacklog gehört einem bestimmten Team**	Ein **Kanbanboard kann von mehreren Teams** oder Personen **geteilt werden**
Schreibt 3 Rollen vor (Product-Owner/ Scrummaster/Team)	**Schreibt gar keine Rollen vor**
Ein **Scrumboard wird** für jeden Sprint **neu eingesetzt**	Ein **Kanbanboard bleibt durchgehend bestehen**
Schreibt ein priorisiertes Produktbacklog vor	**Priorisierung ist optional**

Da. Das isses. Jetzt kennen Sie die Unterschiede.

Aber es ist noch nicht vorbei, jetzt kommt das Beste! Ziehen Sie sich die Stiefel an, es ist Zeit, mit Mattias in die Baugrube zu springen und zu sehen, wie das in der Praxis aussieht!

Teil II – Fallstudie

Kanban im echten Leben



Ideen | Markt | In Entwicklung | Test | Produktiv

Dies ist die Geschichte davon, wie wir durch Kanban gelernt haben, unsere Arbeit zu verbessern. Als wir begonnen haben, gab es nicht viele Informationen und Dr. Google ließ uns ausnahmsweise mit leeren Händen zurück. Heute entwickelt sich Kanban erfolgreich weiter und es gibt immer mehr gesammeltes Wissen. Ich empfehle sehr, sich David Andersons Arbeit anzusehen, z. B. „Serviceklassen“ (classes of service). Hier kommt also der erste (und letzte) Disclaimer (versprochen!). Welche Lösung Sie auch immer einspielen, vergewissern Sie sich, dass sie sich an Ihren spezifischen Problemen ausrichtet. Na also, geschafft. Lassen Sie uns einsteigen. Also, dies ist unsere Geschichte.

/Mattias Skarin

19 Die Natur technischer Abläufe

Wenn Sie jemals 24 Stunden und 7 Tage auf Abruf waren, haben Sie eine ungefähre Ahnung von der Verantwortung, die man fühlt, wenn man eine Produktionsumgebung managt. Es wird von Ihnen erwartet, die Lage mitten in der Nacht auseinander zu dröseln, egal ob Sie das Problem verursacht haben oder nicht. Keiner weiß weiter, deswegen werden Sie angerufen. Es ist eine ganz schöne Herausforderung, weil Sie nicht die Hardware gebaut haben, die Treiber, das Betriebssystem oder die kundenspezifische Software. Ihre Möglichkeiten sind oft begrenzt darauf, das Problem einzugrenzen, die Wirkung abzuschwächen, die Anhaltspunkte festzuhalten, die zum Nachvollziehen des Problems benötigt werden, und auf die verantwortliche Person zu warten, die den ganzen Ärger verursacht hat, um das zu reproduzieren und das Problem zu lösen, das Sie gerade miterlebt haben.

Für technische Abläufe sind die Schnelligkeit und Genauigkeit sowohl für die Ansprechbarkeit als auch für die Problemlösung Schlüsselfaktoren.

20 Warum um alles in der Welt ändern?

2008 durchlief einer unserer Kunden, ein skandinavisches Spieleentwicklungsunternehmen, eine ganze Reihe von Prozessverbesserungen. Eine davon beinhaltete, Scrum auf die Ebene der Entwicklungsorganisation zu skalieren und stückweise Hindernisse abzubauen, die die Entwicklungsteams an der Softwareauslieferung hinderten. Als die Softwareentwicklung flüssiger ablief und die Performanz stieg, vergrößerte das den Druck auf die Technikteams in den späteren Phasen im Ablauf. Früher schaute die Technik meistens nur zu, jetzt wurden sie mehr und mehr als aktive Truppe in den Entwicklungsprozess involviert.

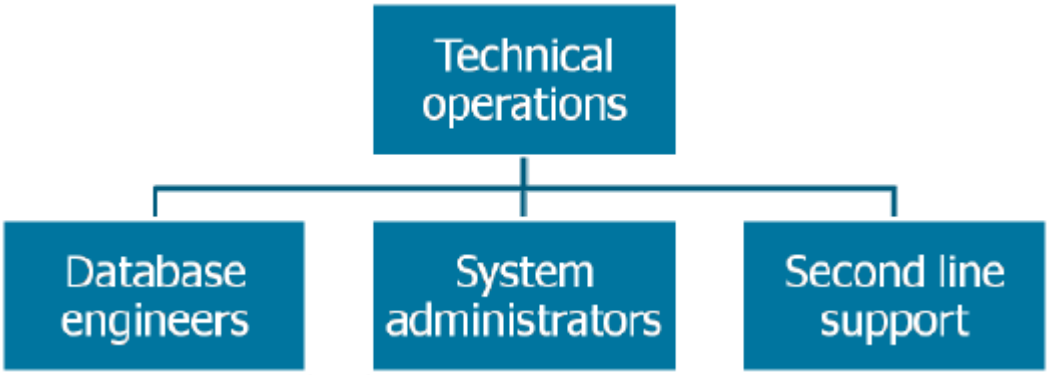


Abbildung 1. Die Organisation der Technik beinhaltet drei Teams: Die Datenbankadministratoren, die Systemadministratoren und den weiterführenden Support

Als Ergebnis kam heraus, dass die Unterstützung der Entwicklungsteams nicht ausreichte. Wenn wir uns nur auf die Entwicklungsteams konzentrieren, würde dies Verzögerungen in der Verbesserung der kritischen Infrastruktur hervorrufen, die durch die Technikteams in Gang gehalten wird. Verbesserungen waren in beiden Bereichen erforderlich.

Außerdem bedeutete ein Fortschritt bei den Entwicklungsteams, dass Manager mehr und mehr angefragt wurden, um bei der Analyse und bei Feedback zu Ideen einzuspringen. Das bedeutete, dass sie weniger Zeit für zeitnahe Aufgabenpriorisierung und zum Problemlösen hatten. Dem Managementteam wurde bewusst, dass sie handeln mussten, bevor die Situation unbeherrschbar wurde.

21 Wo fangen wir an?

Ein guter Startpunkt war, die Entwicklungsteams zu fragen, die die Kunden der Technik waren.

21.1 Die Entwicklersicht auf die Technik

Ich fragte, „Was sind die drei Hauptdinge, die Ihnen in den Kopf kommen, wenn Sie an die ‚Technik' denken?" Die häufigsten Antworten waren:

"Variables Wissen"

"Sehr kompetent, wenn es um Infrastruktur geht"

„Sie möchten uns helfen, aber tatsächlich Hilfe zu bekommen ist schwierig"

„Projekte dauern zu lang"

"Ihr Workflowsystem ist beschissen"

„Was machen die Jungs?"

„Brauchen viele Emails, um simples Zeug zu tun"

„Schwer zu erreichen"

Das war in Kürze die Entwicklersicht auf die Technik. Jetzt lassen Sie uns das mit der Techniksicht auf die Entwicklung vergleichen.

21.2 Technikersicht auf die Entwicklung

Wir (tekkies)

„Warum benutzt Ihr nicht die Vorteile der existierenden Plattform?"

„Lasst uns Releases als weniger schwergewichtigen Akt machen!"

„Eure schlechte Qualität tut uns weh!"

„Sie sollten etwas ändern" – war das gemeinsame Thema in den Argumenten beider Seiten. Offensichtlich, dass diese geistige Haltung eine Veränderung erforderte, wenn wir beim Beheben des gemeinsamen Problems Boden gewinnen wollten. Auf der Plusseite: „sehr kompetent, wenn es um Infrastruktur geht" (deutet auf Vertrauen in die Kernkompetenz hin); das machte mich zuversichtlich, dass die „wir-gegen-die"-Mentalität aufgehoben werden könnte, wenn wir die richtigen Arbeitsbedingungen schaffen. Überstunden zu eliminieren und sich auf Qualität zu konzentrieren, war eine brauchbare Option.

22 Auf die Beine stellen

Also mussten wir in Schwung kommen, aber wo sollten wir anfangen? Das einzige, was wir sicher wussten: Womit wir anfangen, wird nicht das sein, womit wir enden.

Mein Hintergrund ist der eines Entwicklers, so dass ich natürlich wenig über das Wesen von Arbeitsprozessen wusste. Ich wollte nicht „reinstürmen und anfangen, Dinge zu ändern". Ich brauchte einen Ansatz mit weniger Konfrontationspotenzial, der uns die relevanten Dinge lehren, irrelevante Dinge ausschalten und leicht zu lernen sein würde.

Die Kandidaten waren:

1. Scrum – das lief mit Entwicklungsteams gut
2. Kanban – neu und ungetestet, aber gut passend zu den Leanprinzipien, die uns fehlten

In der Diskussion mit Managern schienen Kanban und Leanprinzipien die Probleme anzusprechen, die wir versuchten anzugehen. Aus ihrer Sicht würden Sprints nicht besonders gut passen, da sie eine tägliche Re-Priorisierung vornahmen. Also war Kanban der logische Ausgangspunkt, auch wenn es für uns alle ein unbekanntes Wesen war.

23 Teams in Gang setzen

Wie setzen wir die Teams in Gang? Da draußen gab es kein Handbuch darüber, wie man das auf die Beine stellt. Es falsch zu machen wäre sehr riskant. Abgesehen davon, uns die Verbesserungen entgehen zu lassen, hatten wir es mit einer Produktionsplattform mit hoch spezialisierten und qualifizierten Leuten zu tun, die schwer zu ersetzen waren. Sie abzuschrecken, wäre eine schleeechte Idee.

- Sollten wir einfach anfangen und uns erst mit den Konsequenzen beschäftigen, wenn sie erscheinen?
- Oder sollten wir zuerst einen Workshop durchführen?

Es war für uns offensichtlich – wir sollten zuerst einen Workshop durchführen. Aber wie? Es war eine Herausforderung, das gesamte technische Arbeitsteam zusammen zu bekommen, um an einem Workshop teilzunehmen (wer sollte abnehmen, wenn jemand anrief?). Zuletzt entschieden wir, einen Halbtagsworkshop zu machen und ihn einfach zu halten, auf Grundlage von praktischen Übungen.

23.1 Der Workshop

Einer der Vorteile des Workshops war, dass er helfen würde, unsere Probleme früh zum Vorschein zu bringen. Er sorgte außerdem für eine sehr vertrauensvolle Atmosphäre, wo Auswirkungen direkt mit Teammitgliedern diskutiert werden konnten. Ich meine, seien wir ehrlich – nicht Jeder war übermäßig enthusiastisch darüber, die derzeitige Arbeitsweise zu ändern. Aber die meisten Teammitglieder waren offen dafür, es zu probieren. Also führten wir den Workshop durch, indem wir die wichtigsten Prinzipien demonstrierten, und führten eine Kanbansimulation in kleinem Maßstab durch.

Einige Grundprinzipien lernen

- Begrenze Arbeit auf Kapazität
- Batchgröße vs. Durchlaufzeit
- Arbeit „in Arbeit“ vs Durchfluss
- Theory of Constraints

Kanbandemo

- 3 „Arbeitstypen“;
  beantworte Fragen,
  baue ein Legoauto,
  entwirf und baue ein Haus.
- 3 Iterationen
  Messe Geschwindigkeit pro Arbeitstyp.
  Experimentiere, passe WIP an.
- Nachbesprechung

Am Ende des Workshops ließen wir per „Fist-of-Five“ abstimmen, um zu überprüfen, ob die Teams gewillt waren, das in der Praxis auszuprobieren. An diesem Punkt gab es keine Einwände, so dass wir einvernehmlich weiter gehen konnten. („Fist of Five ist eine konsensbildende Methode, bei der jeder Teilnehmer seine Zustimmung auf einer Skala von 1 – 5 Fingern zeigt. 5 bedeutet starke Zustimmung, 1 bedeutet starke Ablehnung, 3 bedeutet Zustimmung mit Bedenken. Konsens ist erreicht, wenn jede Person der Gruppe mindestens 3 Finger hoch hält.)

24 Stakeholder ansprechen

Es war wahrscheinlich, dass die Stakeholder (Beteiligten) durch die Kanbanumsetzung ebenso betroffen sein würden. Die Änderungen würden jedoch Verbesserungen sein – es bedeutete, dass Team würde beginnen, „nein“ zu Arbeiten zu sagen, die sie nicht fertigstellen könnten, es würde beginnen, sich für Qualität stark zu machen und niedrig priorisierte Punkte aus dem Backlog des Teams zu entfernen. Trotzdem, vorher miteinander zu diskutieren ist immer eine gute Idee.

Die am meisten betroffenen Stakeholder waren die Hotline-Mitarbeiter und die Abteilungsleiter. Da sie am Workshop teilnahmen, waren sie bereits positiv dazu eingestellt, weiter zu machen. Dasselbe galt für die Entwicklungsteams (die mehr oder weniger sowieso Verbesserungen erwarteten). Aber für ein Team, das Supportteam, war die Sache anders. Ihr größtes Problem war, dass sie mit Arbeit überfrachtet waren. Außerdem beschäftigen sie sich mit Kundenproblemen und die Firma hatte sich verpflichtet, auf alle diese Probleme einzugehen. Jetzt war es ziemlich wahrscheinlich, das zu ändern, wenn wir Kanban einsetzen und WIP-Limits zwingend einführen.

Also trafen wir uns mit den Beteiligten und stellten unsere Absichten vor, den erwarteten Nutzen und die möglichen Konsequenzen. Zu meiner Erleichterung wurden unsere Ideen grundsätzlich begrüßt, begleitet durch eine „klasse, wenn wir am Ende diese Punkte beerdigen können“-Bemerkung.

25 Das erste Board einrichten

Eine gute Möglichkeit zu beginnen ist, ein Kanbanboard einzurichten, indem man eine Wertstromkarte macht. Es ist im Grunde eine Visualisierung der Wertkette und bietet Einsicht in die Arbeitszustände, den Ablauf und die Dauer für den Lauf durchs System hindurch (Durchlaufzeit).

Aber wir begannen viel einfacher; als Beispiel-Kanbanboard, gemalt auf Papier, zusammen mit dem Manager. Sahen uns ein paar Zeiten an und damit fingen wir an. Fragen, die in dieser Phase angesprochen wurden, waren unter anderem:

- Welche Arbeitstypen haben wir?
- Wer kümmert sich darum?
- Sollten wir die Verantwortung über verschiedene Arbeitstypen teilen?
- Wie gehen wir mit geteilter Verantwortung angesichts von Spezialkenntnissen um?

Da die unterschiedlichen Arbeitstypen verschiedene Service-Level-Agreements hatten, schien es naheliegend, jedes Team ihr eigenes Board entwerfen zu lassen. Sie suchten sich ihre Spalten und Zeilen selbst zusammen.

Die nächste große Entscheidung war, ob geteilte Verantwortung für die verschiedenen Arbeitstypen eingeführt werden sollte oder nicht. „Sollten wir einen festen Teil des Teams sich mit den direkten Fragen (reaktive Arbeit) befassen lasen und den Rest des Teams sich auf die Projekte (Arbeit in Eigeninitiative) konzentrieren lassen?“ Wir entschieden uns, zuerst die geteilte Verantwortung auszuprobieren. Ein Hauptgrund war, dass wir erkannt hatten, dass Selbstorganisation und stetiges Lernen und Wissenstransfer seitens der Teammitglieder entscheidend waren, um ständiges Wachstum zu ermöglichen. Der Nachteil dieser Entscheidung waren mögliche Unterbrechungen für Jeden, aber das war die beste Lösung, die wir uns für den Anfang vorstellen konnten. Eine kurze Randnotiz: als wir den Workshop durchführten, organisierten sich die Teams rund um dieses Problem tatsächlich selbst. Sie ließen eine Person die dringenden Anfragen bearbeiten und den Rest mit den größeren Dingen.

25.1 Das erste Kanbanmodell

Unten ist das Grundmodell, das wir für Kanban genutzt haben. Beachten Sie, dass das Team entschieden hat, die Arbeitspakete hochsteigen zu lassen (wie Wasserblasen), entgegen dem typischeren Modell, fließend von links nach rechts.

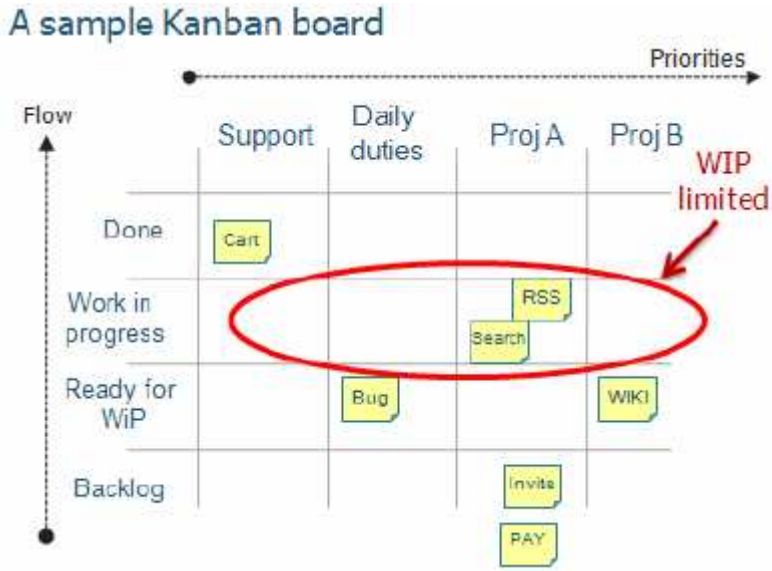


Abbildung 2. Dies ist das erste Modell eines Kanbanboards. Prioritäten laufen von links nach rechts, der Arbeitsablauf läuft aufwärts. WIP wird als Gesamtanzahl von Aufgaben bei den Arbeitspaketen, die gerade in der Spalte „In Arbeit“ (Work in progress) sind, gezählt (eingekringelt). Das Modell ist beeinflusst durch die Erfahrungen, die Linda Cook berichtet hat.

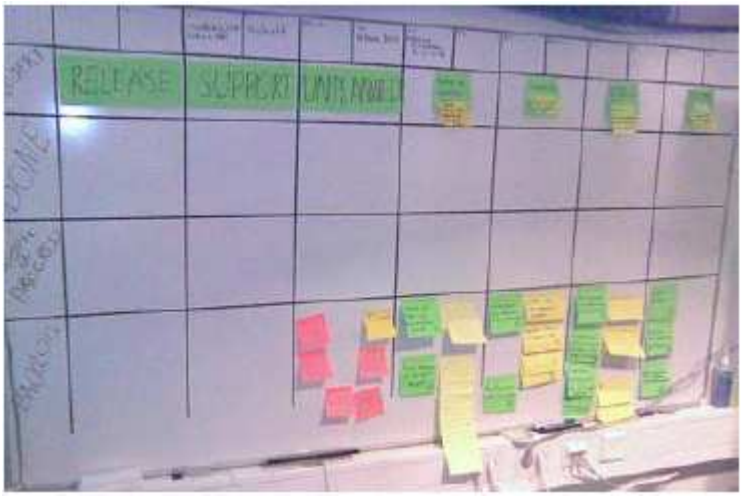


Abbildung 3. Erstes Kanbanboard für das Systemadministrationsteam.

Eingesetzte Zeilen

Ablaufphase (Zeile)	Wie wir es definiert haben
Backlog	Storys, für die sich der Manager als notwendig entschieden hat.
Fertig für WIP (Ready for WIP)	Storys, die geschätzt und auf Aufgaben mit einer maximalen Größe von 8 Stunden heruntergebrochen sind.
In Arbeit (Work in progress)	Die Zeile, die das WIP-Limit hatte. Wir starteten mit einem Limit von 2 x Teamgröße - 1 (die -1 steht für Kollaboration). Damit hat ein 4-Personen-Team ein WIP-Limit von 7.
Fertig (Done)	lauffähig für Anwender.

Eingesetzte Spalten

Arbeitstyp	Wie wir es definiert haben
Release	Hilfreich für Entwicklungsteam bei Softwareauslieferung.
Support	Kleinere Anfragen anderer Teams.
Ungeplant	Unerwartete Arbeit, die erledigt werden muss, aber keinen Eigentümer hat, z. B. kleinere Infrastrukturverbesserungen.
Projekt A	Größeres IT-Projekt, z. B. die Hardware einer Softwareumgebung ändern.
Projekt B	Ein anderes größeres Projekt.

Nicht alle Kanbanboards sahen gleich aus. Alle fingen mit einem einfachen Entwurf an und entwickelten sich im Laufe der Zeit.

26 Das erste WIP-Limit setzen

Unser erstes WIP-Limit war ziemlich großzügig. Wir überlegten uns, dass wir durch Visualisierung den Arbeitsablauf sehen und erfahren würden, was vor sich ging, und es war unwahrscheinlich, dass wir von

Anfang an das beste Limit erraten könnten. Mit der Zeit würden wir die WIP-Limits anpassen, jedes Mal, wenn wir einen guten Grund dafür fanden (alles, was wir tun mussten war, auf das Board zu zeigen).

Das erste WIP-Limit, das wir einsetzten, war 2n - 1 (n = Anzahl der Teammitglieder, -1 um Kooperation zu animieren). Warum? Ganz einfach, wir hatten keine bessere Idee ☺. Also schien es unverfänglich, damit zu starten. Die Formel bot eine einfache und logische Erklärung für Jeden, der versuchte, Arbeit für das Team einzubringen: „…vorausgesetzt, dass jedes Teammitglied maximal an zwei Dingen zur gleichen Zeit arbeiten kann, eins aktiv und eins in Warteposition, warum würden Sie erwarten, dass sie *noch eins* übernehmen?“ Rückblickend hätte jedes großzügige Limit erst einmal funktioniert. Durch Kontrolle des Boards ist es einfach, die richtigen Limits im weiteren Verlauf herauszufinden.

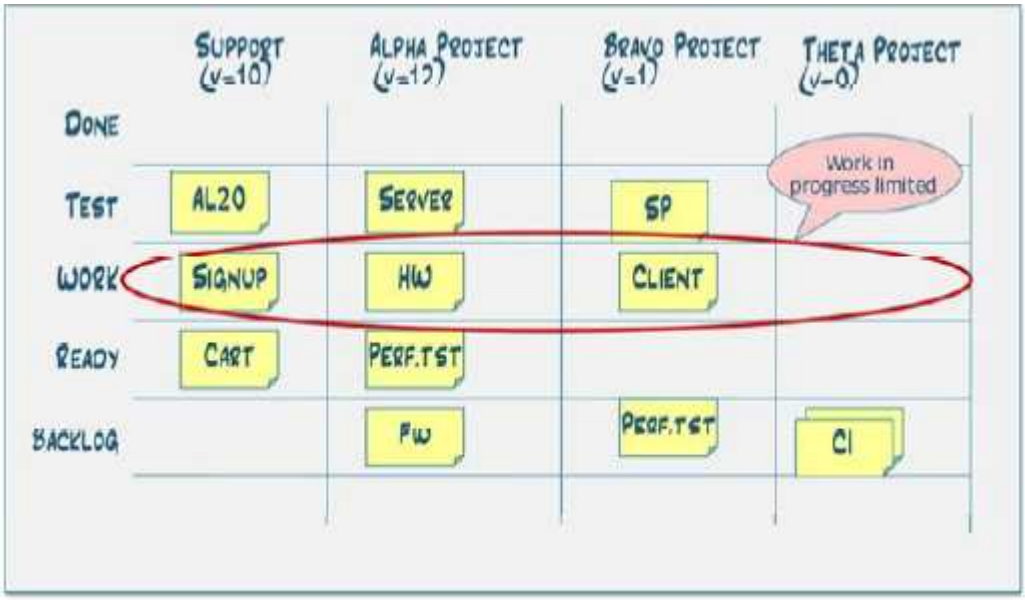


Abbildung 4. Wie wir das WIP-Limit für DB-Administrations- und Systemadministrationsteam ansetzten: ein Limit über alle Arbeitstypen.

Eine Beobachtung, die wir machten, war, dass es nutzlos war, das WIP-Limit in Story-Points zu definieren. Das war einfach zu schwierig nachzuhalten. Das einzige Limit, das einfach genug nachzuhalten war, war einfach die Anzahl der Arbeitspakete (= parallele Aufgaben).

Für das Supportteam setzten wir WIP-Limits pro Spalte ein. Und zwar, weil wir kürzere Reaktionszeiten brauchten, wenn das Limit überschritten wurde.

27 Das WIP-Limit akzeptieren

Während das Berücksichtigen eines WIP-Limits sich in der Theorie einfach anhört, ist es ein raues Geschäft, es in der Praxis zu akzeptieren. Es bedeutet, an einem gewissen Punkt „nein“ zu sagen. Wir probierten verschiedene Ansätze aus, um damit umzugehen.

27.1 Am Board diskutieren

Wenn eine Verletzung entdeckt wurde, brachten wir die Stakeholder ans Board und fragten, was sie erreichen wollten. Zuerst war der häufigste Grund für die Verletzung schlicht Unerfahrenheit. In einigen Fällen fanden wir verschiedene Ansichten zur Priorisierung, ein typischer Fall war, dass ein Spezialistenteammitglied in einem spezifischen Bereich arbeitete. Das waren die einzigen Punkte, an denen wir Unstimmigkeiten erfuhren, die meiste Zeit wurden die Arbeitspakete durch Diskussionen vor dem Board sofort geklärt.

27.2 Überlauf kennzeichnen

Wenn „Nein“-Sagen zu einer Konfrontation geführt hätte, und das Weglassen der Arbeitspakete schwierig gewesen wäre, schoben wir die Arbeitspakete mit geringer Priorität in einen „Überlaufbereich“ am Board, wenn WIP-Limits überschritten wurden. Zwei Regeln finden bei Überlaufaufgaben Anwendung:

1. Sie werden nicht vergessen; wann immer wir Zeit haben, werden wir uns mit ihnen beschäftigen
2. Wenn wir sie aussortieren, werden Sie informiert.

Nach nur zwei Wochen war es offensichtlich, dass die Überlaufarbeitspakete gar nicht behandelt würden, also konnten sie mit Unterstützung des Teammanagers endgültig entfernt werden.

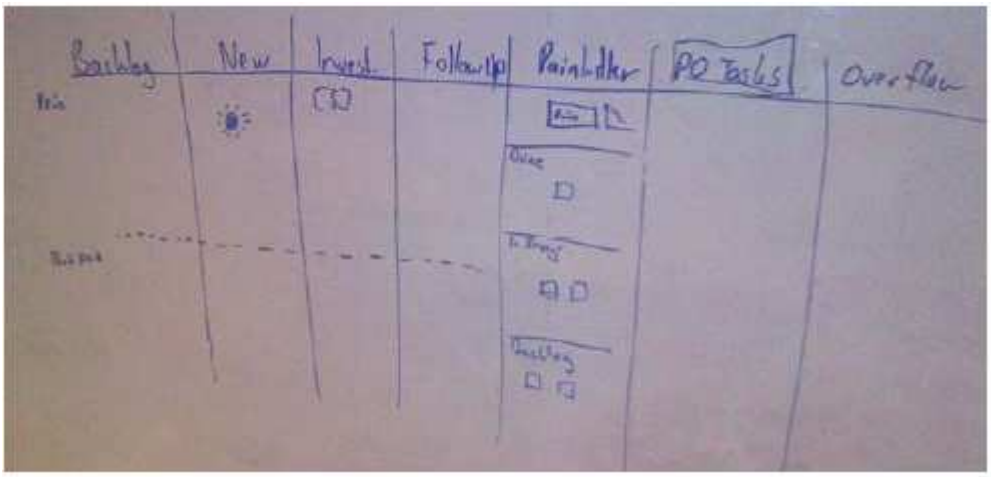


Abbildung 5. Ein Entwurf des Kanbanboards für das Supportteam; der Überlaufbereich ist ganz rechts.

28 Welche Aufgaben kommen an das Board?

Wir entschieden früh, nicht *alle* Arbeit, die das Team tat, dem Board hinzuzufügen. Dinge wie einen Anruf zu überwachen oder Kaffee zu besorgen, würden Kanban in ein Verwaltungsmonster verwandeln. Wir waren da, um Probleme zu *lösen*, nicht zu erschaffen ☺. Also entschieden wir, nur die Aufgaben mit einer Größe > 1 Stunde ans Board zu hängen, alles Kleinere wurde als „weißes Rauschen“ betrachtet. Das 1h-Limit funktionierte tatsächlich recht gut und war eines der wenigen Dinge, die unverändert blieben. (Wir mussten die Annahme darüber korrigieren, was einen Einfluss auf das Hintergrundrauschen hatte, aber mehr davon später.)

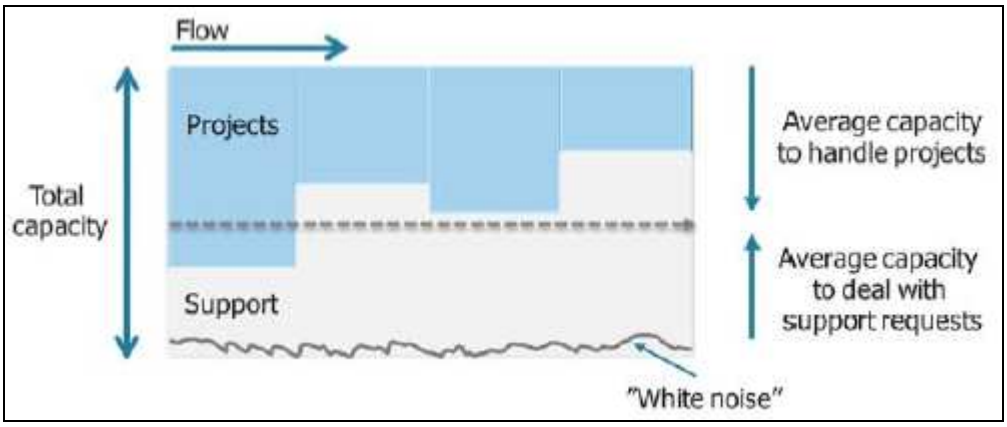


Abbildung 6. Wir begannen mit der Annahme, dass die Gesamtkapazität hauptsächlich durch zwei Arbeitstypen gebunden wurde, größere (Projekte) und kleinere (Support). Die Verfolgung der Entwicklungsgeschwindigkeit konnte uns einen Hinweis zum Auslieferungstermin geben, falls das erforderlich war. „Weißes Rauschen“ (kleiner Support < 1h, Besprechungen, Kaffee holen, Kolleg/innen helfen) wurde als immer vorhanden erwartet.

29 Wie schätzen?

Das ist ein laufender Posten; und dazu gibt es sicher mehr als eine Antwort:

- Regelmäßig schätzen.
- Schätzen, wenn es erforderlich ist.
- Ideale Tage/Story-Points für Schätzungen einsetzen.
- Schätzungen sind unsicher, setzen Sie T-Shirt-Größen ein (Small, Medium, Large).
- Schätzen Sie nicht, oder schätzen Sie nur, wenn Kosten für Verzögerungen es rechtfertigen.

Etwas beeinflusst durch Scrum (da es das war, wo wir herkamen, letzten Endes), entschieden wir, mit Story-Points zu beginnen. Aber in der Praxis behandelten die Teams Story-Points äquivalent zu Personenstunden (das war für sie naheliegend). Am Anfang wurden alle Storys geschätzt. Mit der Zeit lernten die Manager, dass sie ihre Stakeholder nicht warten lassen mussten, wenn sie die Anzahl der gleichzeitigen Projekte niedrig hielten. Sie lernten außerdem, dass sie im Fall plötzlicher Änderungen neu priorisieren und das Problem angehen konnten.

Der Bedarf, einen Liefertermin zu schätzen, war nicht mehr so eine große Sache. Das veranlasste die Manager dazu, nicht mehr nach Schätzungen im Voraus zu fragen. Sie taten das nur noch, wenn sie fürchteten, dass sie Leute warten lassen müssten.

Zu einem frühen Zeitpunkt versprach ein Manager, gestresst durch einen Telefonanruf, die Lieferung eines Projekts „zum Ende dieser Woche“. Da es ein Projekt am Kanbanboard war, war es einfach, den Fortschritt abzuschätzen (zähle fertiggestellte Storys) und zu folgern, dass es nach einer Woche zu etwa 25 % bearbeitet sein würde. Damit waren weitere drei Wochen notwendig. Konfrontiert mit dieser Tatsache änderte der Manager die Priorität des Projekts, stoppte die laufenden Arbeiten und machte die Auslieferung möglich. Überprüfen Sie immer das Board ☺

29.1 Was bedeutet geschätzte Größe? Bearbeitungszeit oder Arbeitszeit?

Unsere Story-Points spiegelten die Arbeitszeit wieder, d. h. wie viele Stunden ununterbrochenen Aufwand wir für diese Story erwarteten; keine Bearbeitungszeit (oder Kalenderzeit, oder wieviele Stunden Wartezeit). Durch Messen der Anzahl von Story-Points, was jede Woche den Zustand „bearbeitet“ erreichte (Entwicklungsgeschwindigkeit), konnten wir auf die Bearbeitungszeit rückschließen.

Wir schätzten jede neue Story nur einmal, wir revidierten Story-Schätzungen nicht während der Ausführung. Das erlaubte uns, die Zeit zu reduzieren, die das Team für Schätzungen aufbringen musste.

30 Also, wie arbeiteten wir denn nun wirklich?

Kanban ist wirklich sehr zwanglos, Sie können in jeder Art und Weise arbeiten. Sie können Ihr Team mit zeitbasierten Aktivitäten arbeiten lassen, oder Sie können sich aussuchen, Aktivitäten anzugehen, wenn genug Eigendynamik aufgebaut ist, um sie zu rechtfertigen.

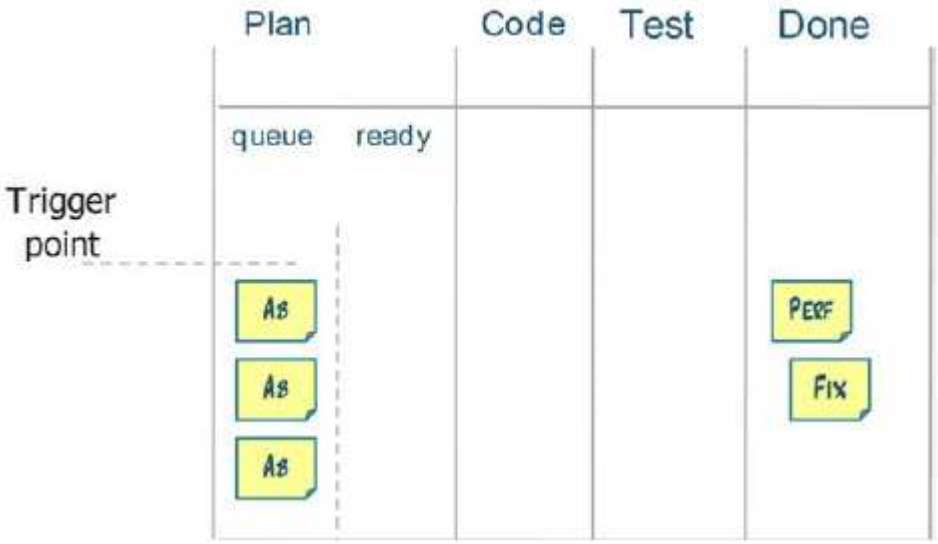
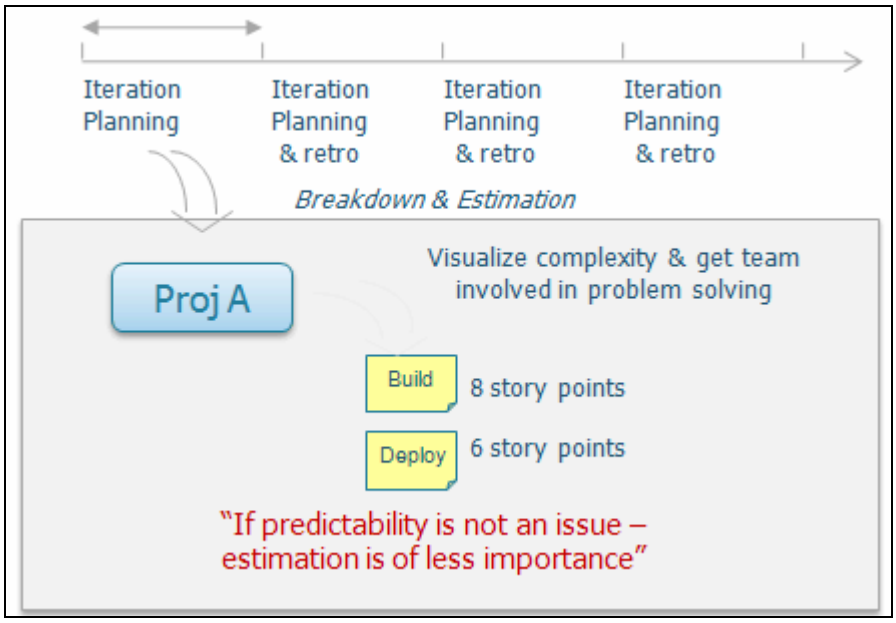


Abbildung 7. Wenn drei Aufgaben im Backlog eingetroffen sind, löst das ein Planungs-/Schätzungsereignis aus.

Wir suchten uns aus, zwei wiederkehrende Ereignisse einzuplanen:

- Tägliche Standups – mit dem Team vor dem Board, um Probleme zu Tage zu bringen und dabei zu helfen, uns ein gemeinsames Bild anderer Teammitgliedsaufgaben zu machen.
- Wöchentliche Iterationsplanungen, zum Planen und für ständige Verbesserungszwecke.



Visualisiere Komplexität & und beziehe das Team ins Problemlösen ein

„Wenn Vorhersagbarkeit keine Rolle spielt –
ist Schätzen von geringerer Wichtigkeit“

Das klappte bei uns gut.

30.1 Tägliches Standup

Die tägliche Standup-Besprechung war einem Daily-Scrum ähnlich. Sie wurde abgehalten nach dem täglichen „Scrum-of-Scrums“-Treffen mit Teilnehmern aus allen Teams (Entwicklung, Test, Betrieb). Das Scrum-of-Scrums lieferte den Kanbanteams wichtige Beiträge, z. B. welche Probleme zuerst behandelt werden sollen oder welches Entwicklungsteam gerade die meisten Schmerzen hatte. Am Anfang nahmen Manager regelmäßig an diesen täglichen Besprechungen im Stehen ein, schlugen Lösungen vor und priorisierten Entscheidungen. Mit der Zeit, als die Teams in der Selbstorganisation besser wurden, nahmen die Manager seltener teil (waren aber nicht weit weg, falls sie gebraucht würden).

30.2 Iterationsplanung

Einmal pro Woche trafen wir uns zur Iterationsplanung. Wir machten sie wöchentlich immer zur selben Zeit, weil wir die Erfahrung machten, dass die Zeit durch andere Prioritäten ausgefüllt wurde ☺, wenn wir das nicht so einplanten. Wir benötigten außerdem mehr Teamkommunikation. Der typische Ablauf war:

- Diagramme und das Board aktualisieren. (Erledigte Projekte auf eine „Wand des Erledigten“ verschieben.)
- Rückblick auf die vergangene Woche. Was war passiert? Warum war das so? Was könnte getan werden, um das zu verbessern?
- Neueinstellung der WIP-Limits (falls nötig).
- Herunterbrechen der Aufgaben und Schätzen neuer Projekte (falls benötigt).

Grundsätzlich war die Iterationsplanung eine Kombination aus Schätzung und kontinuierlicher Verbesserung. Kleine bis mittlere Angelegenheiten wurden sofort erledigt, mit Unterstützung der jeweiligen Vorgesetzten. Aber genug Zug beizubehalten, was komplexe Angelegenheiten und Infrastrukturaufgaben anging, war eine heftigere Tortur. Um damit umzugehen, führten wir für die Teams die Möglichkeit ein, ihren Vorgesetzten bis zu zwei „Teamhindernisse“ aufzutragen.

Die Regeln waren:

1. Vorgesetzte können an zwei Slots arbeiten, zu jedem einzelnen Zeitpunkt.
2. Wenn beide gefüllt sind, kann man einen neuen hinzufügen, so lange man einen weniger wichtigen wegnimmt.
3. Das Team entscheidet, wann eine Angelegenheit erledigt ist.

Das war eine eindeutige Wendung. Plötzlich konnten Teams sehen, dass Vorgesetzte arbeiteten, um ihnen sogar bei schwierigen Angelegenheiten auszuhelfen. Sie konnten auf ihre Hindernisse zeigen und fragen „wie geht’s da voran?“ Sie wurden nicht mehr vergessen oder durch neue hoch prioritäre Strategien umgestoßen.

Ein Beispiel eines ernsthaften Hindernisses war, dass das Technikteam[2] nicht die Hilfe bekam, die es von den Entwicklern brauchte, als das Technikteam einen Bug vermutete. Sie brauchten einen Entwickler, um dabei zu helfen herauszufinden, welcher Teil des Systems das Problem verursachte, aber da die Entwickler mit ihren Sprints beschäftigt waren, um neue Sachen zu entwickeln, stapelten sich die Probleme immer mehr. Nicht überraschend, empfand das das Technikteam so, dass die Entwickler sich nicht um Qualität kümmerten.

Als dieses Hindernis an die Oberfläche kam, wurde es zunächst zum direkten Vorgesetzten eskaliert und dann weiter zum Abteilungsleiter. Er setzte eine Besprechung zusammen mit dem Leiter der Entwicklung an. In den folgenden Diskussionen einigten sich die Vorgesetzten darauf, Qualität an erste Stelle zu setzen. Sie gestalteten einen Ringversuch als Lösung für den Support – in jedem Sprint sollte ein Entwicklungsteam „auf Abruf“ und sofort verfügbar sein, um dem Technikteam zu helfen. Nach Sicherstellung des Supports seitens der Manager übergab der Leiter der Entwicklung eine Liste von Ansprechpartnern an die Supportteams. Sofort setzten sie einen Test für diese Lösung an, mutmaßend, dass der Plan nicht funktionieren würde. Aber diesmal war es effektiv; die notwendigen Vorbereitungen waren diesmal getroffen worden und das Hindernis wurde als gelöst betrachtet. Das brachte den Technikteams große Erleichterung.

31 Ein Planungskonzept finden, das funktioniert

31.1 Eine Geschichte

Ich erinnere mich an einen Wendepunkt für eins der Teams. Ich saß mit ihnen in ihrer zweiten Schätzungssitzung. Das Team steckte an einem Projekt fest, bei dem sie nicht wussten, wie sie schätzen sollten. Es gab zu viele Unbekannte und die ganze Schätzungssitzung kam zum Stillstand. Statt einzugreifen und die Führung zu übernehmen, bat ich sie darum, den Prozess weiter zu entwickeln, um eine bessere Lösung zu finden. Angeführt von ihrem Leiter, nahmen sie die Herausforderung an und begann, ihre eigene Lösung zu entwerfen. Dieses Ereignis war ein wichtiger Wendepunkt, ein wichtiger „Gewinn“, von dem aus sie sich zu einem Team mit Selbstvertrauen entwickelten. Danach fingen sie an, sich so schnell weiter zu entwickeln, dass wir für sie den Weg räumen mussten.

Zwei Monate später wandte sich ihr Vorgesetzter nach einer Retrospektive an mich. „Ich habe ein Problem“, sagte er, indem er auf das Kanbanboard des Teams zeigte. „Wir haben keine echten Probleme, was sollen wir tun?“

31.2 Planung neu erfinden

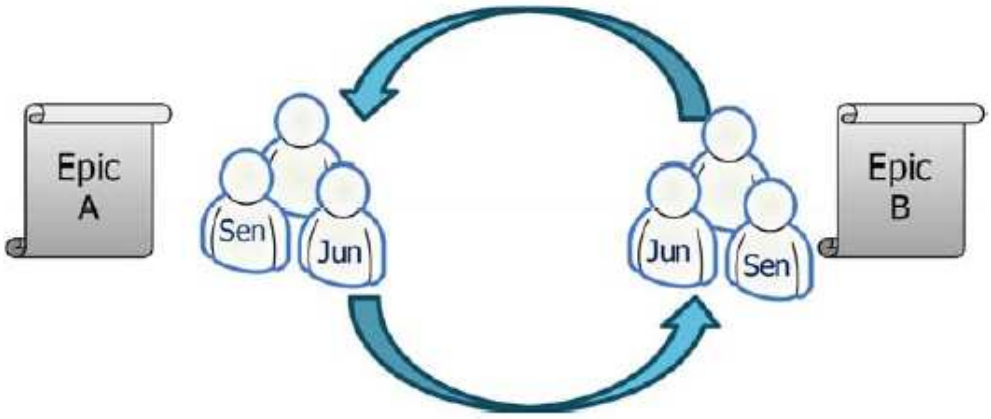
Planning-Poker-Schätzungen mit allen Teammitgliedern funktionierte nicht für jedes der Technikteams gut. Einige Gründe:

1. Wissen war zu ungleichmäßig über das Team verteilt.
2. Meist sprach nur eine Person.
3. Teammitglieder wollten dringende Punkte ansprechen, die sie auf ihren Schreibtischen liegen hatten.

Aber durchs Experimentieren ließen sich die Teams unabhängig zwei verschiedene Schätzprozesse einfallen. Jeder funktionierte für das jeweilige Team gut.

[2] Zur Erinnerung: Datenbankadministratoren, Systemadministratoren und weiterführender Support

31.2.1 Ansatz 1 – Verschieben und Bewerten



- Für jedes Projekt/jede Story, bilde ein Senior+Junior-Paar, um zu schätzen (d.h. eine Person, die die jeweilige Story gut kennt, und eine Person, die sie nicht kennt). Das hilft, Wissen zu verteilen.
- Die übrigen Teammitglieder wählen aus, welche Story sie schätzen helfen möchten (aber mit einer Begrenzung auf vier Leute pro Story, um die Diskussionen effektiv zu halten).
- Jedes Schätzteam übernimmt das Herunterbrechen der Aufgaben für ihre Story und, falls nötig, schätzt.
- Dann tauschen die Teams die Storys und bewertet gegenseitig ihr Ergebnis (eine Person pro Team „steht im Hintergrund“, um das Ergebnis seines Teams den Bewertern zu erklären).
- Fertig!

Typischerweise dauerte die gesamte Iterationsplanungssitzung etwa 45 Minuten und der Energielevel blieb während der Sitzung hoch. Ein bis zwei Anpassungen wurden normalerweise gemacht, wenn die Storys rotierten und durch ein neues Augenpaar bewertet wurden.

31.2.2 Ansatz 2 – Erfahren Hochgebirge passieren, dann Schätzen

Zwei Seniorteammitglieder machten vor dem Planen eine Bewertung der Story/des Projekts auf einer hohen Ebene. Sie analysierten Architekturlösungen und entschieden sich für das Problem für eine. Einmal festgesetzt, kam das Team dazu und brach die Story auf Aufgaben herunter, indem es die vorgeschlagenen Lösungen als Startpunkt nutzte.

Abbildung 8. In der Iterationsplanung Aufgaben herunterbrechen, begutachtet durch Experten eines anderen Teams.

32 Was messen?

Es gibt viele Dinge, die man messen könnte – Durchlaufzeit (Zeit von wann ein Bedarf entdeckt wird bis zu wann der Bedarf gedeckt ist), Entwicklungsgeschwindigkeit, Warteschlangen, Abarbeitung… Die wichtige Frage ist, welche Maße *genutzt* werden können, um den Prozess zu verbessern. Mein Rat ist, zu experimentieren und zu sehen, was für Sie funktioniert. Wir haben die Erfahrung gemacht, dass Burndown-Charts[3] für alle Projekte kürzer als 4 Wochen des Guten zuviel waren. Der Gesamtfortschritt konnte einfach durch einen Blick aufs Kanbanboard festgestellt werden (wie viele Storys waren im Backlog und wie viele waren erledigt).

Kandidat für Metrik	**Pro**	**Kontra**
Durchlaufzeit	Leicht zu messen. Keine Schätzung erforderlich. Beginnt und endet mit dem Kunden.	Berücksichtigt Größe nicht.
Entwicklungsgeschwindigkeit gesamt (aggregiert über alle Arbeitstypen)	Ein grober, aber einfacher Indikator für die Richtung der Verbesserung und Abweichung.	Hilft nicht, Liefertermine für spezielle Arbeitstypen vorherzusagen.
Entwicklungsgeschwindigkeit pro Arbeitstyp	Präziser als die Gesamtgeschwindigkeit.	Um nutzbringend zu sein, muss man beim Kundenbedarf anfangen und bis zur Auslieferung messen. Braucht etwas mehr Zeit, Dinge ausfindig zu machen, verglichen zur Gesamtgeschwindigkeit.
Längen der Warteschlangen	Ein schneller Indikator für Anforderungsfluktuation. Einfach zu visualisieren.	Sagt Ihnen nicht, ob der Grund ein ungleichmäßiger Bedarf ist oder unausgeglichene Kapazitäten. Eine leere Warteschlange könnte tatsächlich für Überkapazität sprechen.

Wir begannen damit, „Geschwindigkeit pro Arbeitstyp“ und „Längen der Warteschlangen“ zu messen. Entwicklungsgeschwindigkeit pro Arbeitstyp ist einfach zu messen und tut’s. Warteschlangenlängen sind ein guter Hauptindikator, da sie sofort gesehen werden können (sobald Sie wissen, wo Sie danach suchen müssen).

[3] Diagramme, die zeigen, wieviel der gesamten Arbeit schon erledigt ist

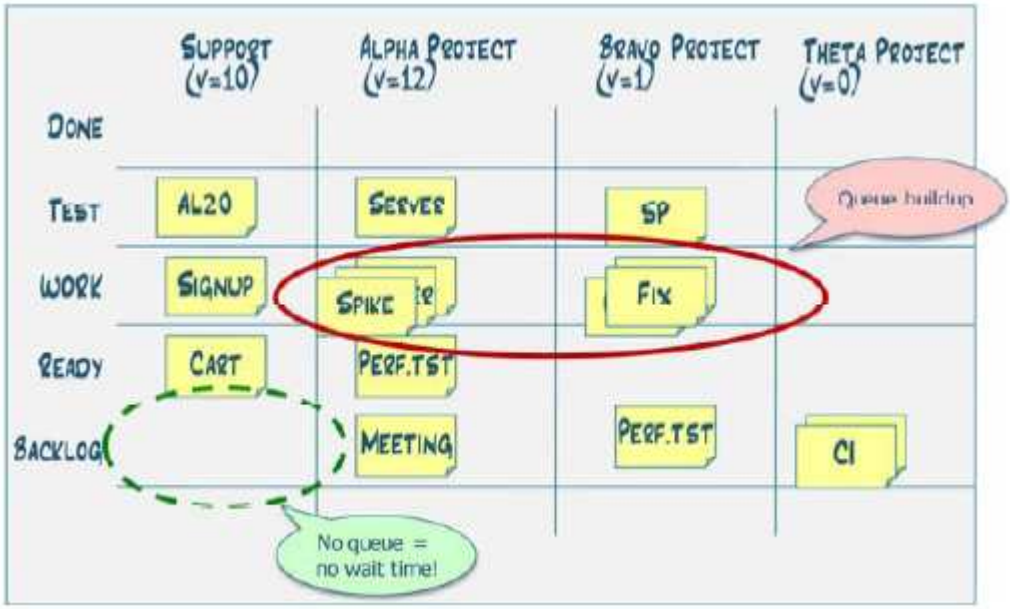
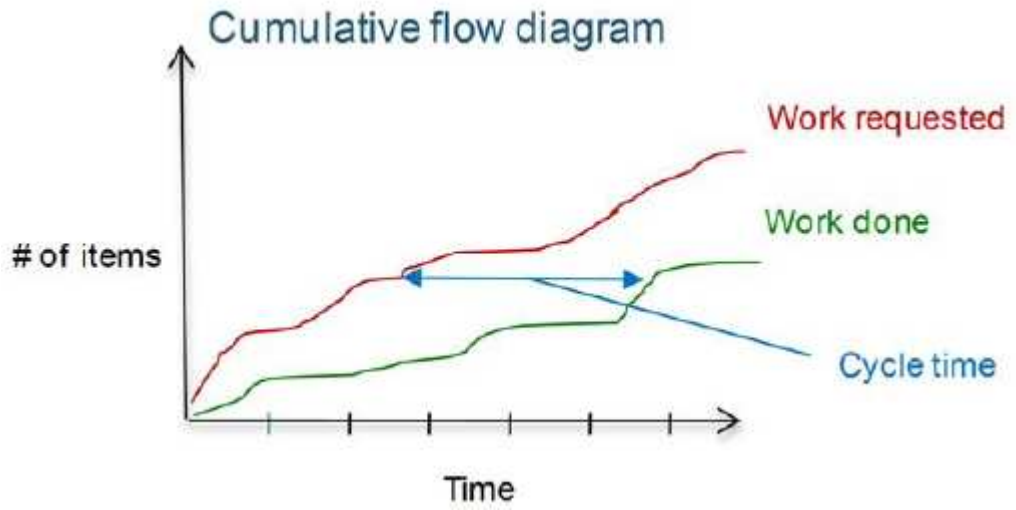


Abbildung 9. Engpässe und Freiräume. Die rote Markierung zeigt, wie die Warteschlangen sich aufgebaut haben, und macht einen Testengpass sichtbar. Die Abwesenheit jeglicher Warteschlange in der Supportspalte deutet darauf hin, dass es keine Wartezeit für neue Supportaufgaben gibt. Das ist ein gutes Zeichen für einen hohen Level an Kundenservice.

Wir haben kein gestapeltes Flächendiagramm (Cumulative flow diagram) eingesetzt, obwohl das schon interessant gewesen wäre.



Wir haben kein gestapeltes Diagramm eingesetzt, weil das Kanbanboard und das Geschwindigkeitsdiagramm uns ausreichend Informationen lieferten, zumindest in unseren frühen Phasen. Engpässe, Unausgeglichenheit und Überstunden konnten trotzdem leicht identifiziert werden, und diese Dinge zu lösen, beschäftigte uns über die ersten sechst Monate ständig.

33 Wie Dinge sich zu verändern begannen

Drei Monate nach der Einführung von Kanban wurde das Systemadministrationsteam vom Management als „Team mit der besten Leistung“ in der IT-Abteilung ausgezeichnet. Zur selben Zeit wurde das Systemadministrationsteam außerdem in der Unternehmensretrospektive als eins der drei besten „positiven Erfahrungen“ gewählt. Die Unternehmensretrospektive ist ein unternehmensweites Ereignis, das alle 6 Wochen stattfindet, und diesmal war es das erste Mal, dass ein *Team* auf der Liste der ersten Drei erschien! Und nur 3 Monate früher hatten diese Teams Engpässe, über die sich die meisten Leute beschwerten.

Die Servicequalität hatte sich deutlich verbessert. Also, was war passiert?

Der entscheidende Moment war der, als alle anfingen, an einem Strang zu ziehen. Vorgesetzte gaben einen klaren Fokus vor und schützten das Team vor Arbeit, die dort nicht hingehörte, und die Teams übernahmen Verantwortung für Qualität und Deadlines. Es brauchte ungefähr drei bis vier Monate, bis dies auftrat, aber danach lief es reibungslos. Es ist nicht so, dass jedes Problem in der Welt verschwunden war (das würde uns allen unsere Arbeit wegnehmen, richtig? ☺) – aber wir sahen uns neuen Herausforderungen gegenüber wie „wie hält man ein Team motiviert, um Verbesserungen zu erreichen (wenn es nicht länger einen Engpass darstellt)?“

Ein wichtiger Teil des Selbstorganisationspuzzles war die Einführung des Konzepts „ein Ansprechpartner pro Team“. Das bedeutete, jedem Entwicklungsteam seinen eigenen Supportkontakt im Technikteam zu geben. Kanban machte das möglich dadurch, dass es Technikteammitgliedern erlaubte, sich um ihre Arbeit herum selbst zu organisieren, Überstunden zu vermeiden und kontinuierliche Verbesserungen zu ermöglichen. Vorher nahm sich eine zufällige Person Arbeit aus der Warteschlange, löste das so weit sie es konnte und fing dann das nächste an. Jedes Missverständnis bedeutete, wieder ganz neu mit einer Supportanfrage anzufangen. Als das 1-zu-1-Konzept eingeführt wurde, hatte das Supportteam plötzlich die Möglichkeit, schnell zu antworten, wenn es schlechten Input und Qualitätsprobleme gab, die das System gefährdeten.

Schnell entstanden Kundenkommunikationsprotokolle; Technikmitarbeiter begannen, Instant-Messaging zu benutzen, um mit den Entwicklern zu sprechen, die sie gut kannten, Emails denjenigen zu schreiben, die besser schreiben als sprechen konnten, und zu telefonieren, wenn das der schnellste Weg war, das Problem zu lösen ☺.

Vorher

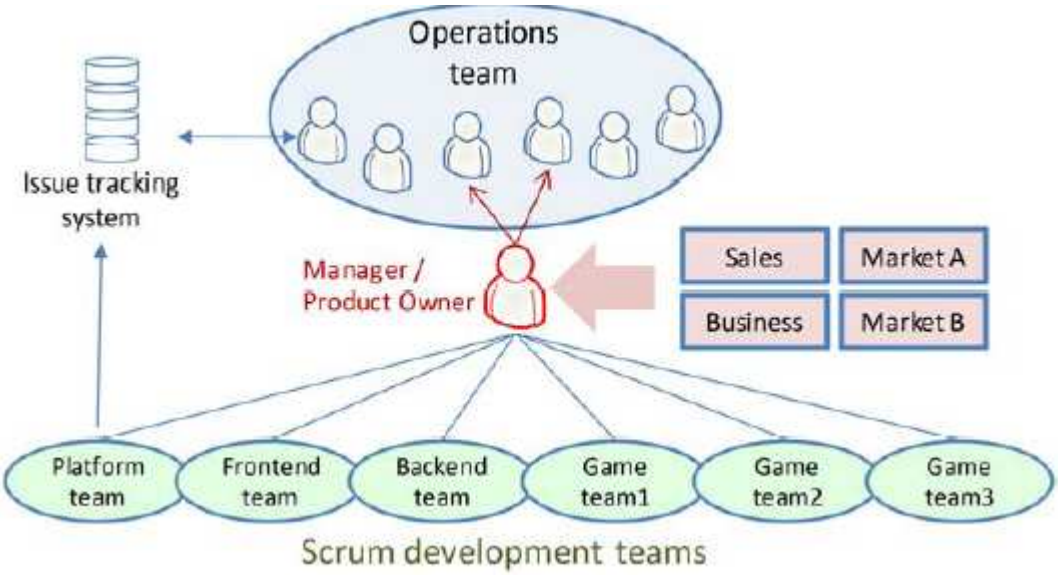


Abbildung 10. Vorher: Der direkte Vorgesetzte ist der Hauptansprechpartner für das Team. Alles, was wichtig ist, geht über ihn. Kleinere Angelegenheiten, typischerweise Probleme der Entwickler, werden durch ein Trackingsystem erfasst. Wenige direkte Interaktionen von einer Person zur anderen.

Nachher

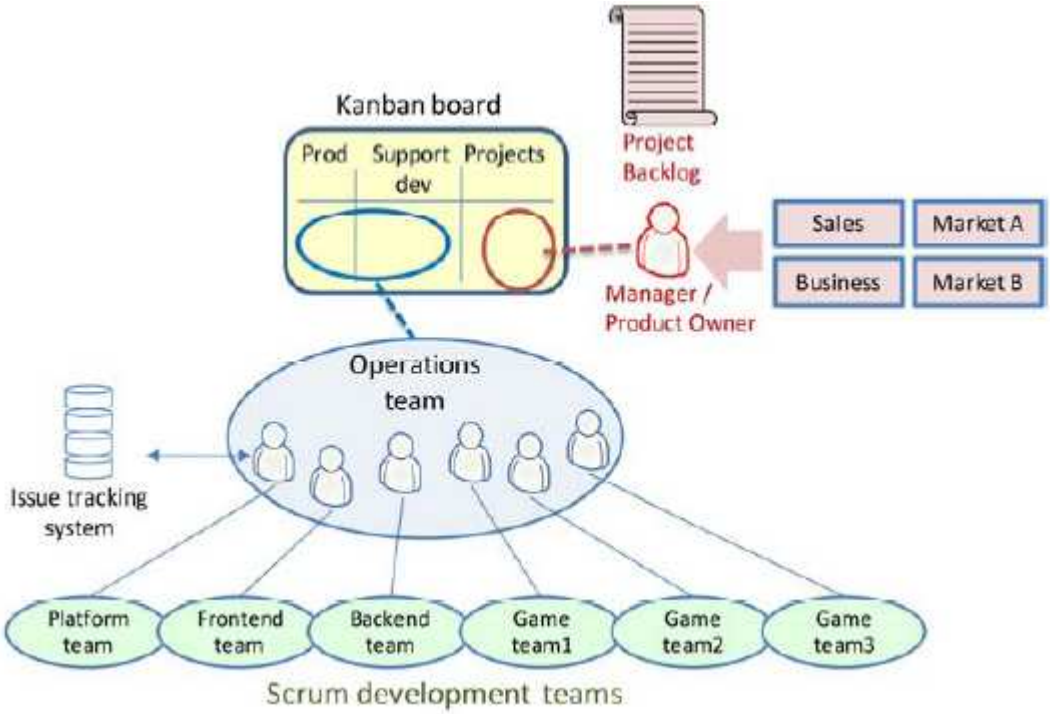


Abbildung 11. Nachher: „ein Ansprechpartner pro Team“ eingeführt. Entwicklungsteams reden direkt mit einem definierten Ansprechpartner im Technikteam. Viele Interaktionen von Person zu Person. Mitglieder des Technikteams organisieren ihre Arbeit selbst, indem sie ein Kanbanboard nutzen. Der Vorgesetzte verschiebt den Fokus darauf, größere Projekte zu priorisieren und macht seinen Leuten den Rücken frei, wenn Probleme auftreten.

Und, was ist die Auswirkung auf die Arbeitsleistung des Teams?

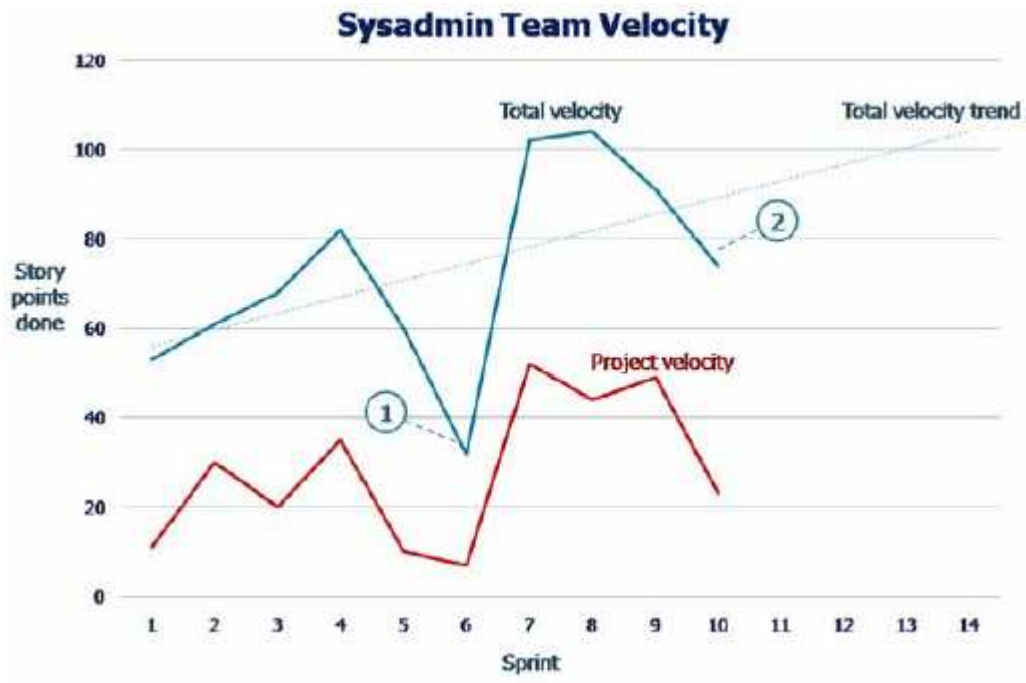


Abbildung 12. Gesamtentwicklungsgeschwindigkeit und Projektgeschwindigkeit, gemessen als Story-Points „erledigt“ pro Woche. Gesamtsumme ist die Summe über alle Spalten, die Projektgeschwindigkeit stellt den Teil dar, der zu „Projekten“ gehört (größere Arbeitsstücke, zum Beispiel eine Hardwareplattform aufrüsten). Die zwei Absenkungen korrelieren mit 1) einer Woche, in der fast alle Teammitglieder auf Reisen waren und 2) einem größeren Release der Entwicklung.

Demnach zeigte das Team einen insgesamt positiven Trend. Zur selben Zeit investierte das Team viel in das Teilen von Wissen, indem sie Pair-Programming einsetzten.

Wo wir gerade dabei sind, lassen Sie uns einen Blick auf die Arbeitsleistung des Datenbankadministrationsteams werfen.

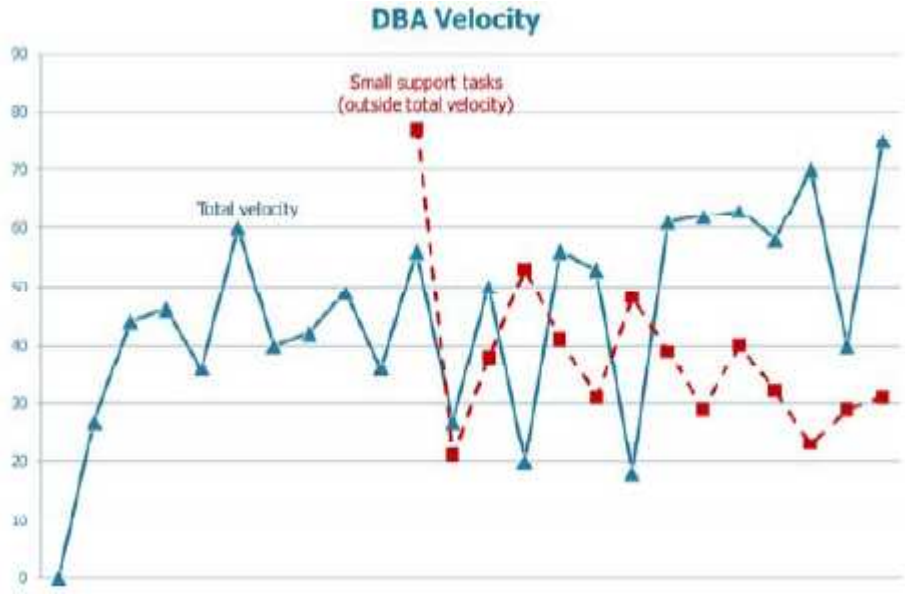


Abbildung 13. Gesamtgeschwindigkeit und kleine Supportaufgaben. Die Absenkung in der Mitte korrespondiert mit Weihnachten.

Die Gesamtgeschwindigkeit hat einen Abwärtstrend, obwohl die Varianz signifikant ist. Die Größe der Varianz inspirierte das Team dazu, die Anzahl der kleinen Supportaufgaben zu überwachen (Aufgaben, die

normalerweise zu klein sind, um sie ans Kanbanboard zu hängen). Wie Sie sehen können, weist der Graph darauf hin, dass es eine deutliche inverse Korrelation zwischen der Anzahl der kleinen Supportaufgaben und der Gesamtgeschwindigkeit gibt.

Das Supportteam begann später mit Kanban als die anderen beiden Teams, also haben wir noch nicht viele verlässliche Daten.

33.1 Es reift

Als wir anfingen, war das Finden von Problembereichen einfach. Aber die größten Chancen für Verbesserungen zu orten, war schwierig. Das Kanbanboard gab uns einen ganz neuen Transparenzlevel. Es war nicht nur einfacher, Probleme genau zu lokalisieren, auch wichtige Fragen brachte der Ablauf hoch, Varianz und Warteschlangen. Wir begannen, die Warteschlangen als Werkzeug zu nutzen, um Probleme zu entdecken. Vier Monate nach dem Starten von Kanban, brachten die Vorgesetzten Quellen für Abweichungen zur Strecke, die ihre Teams verletzten.

Als Teams aus einzelnen Individuen entstanden und selbstorganisierte Einheiten bildeten, wurden den Führungskräften bewusst, dass sie einer neuen Menge von Führungsaufgaben gegenüberstanden. Sie mussten mehr mit Angelegenheiten von Personen umgehen – mit Beschwerden umgehen, gemeinsame Ziele definieren, Konflikte lösen und Einigungen aushandeln. Kein schmerzloser Übergang – sie äußerten offen, dass dies zu lernen, Fähigkeiten und Energie erforderte. Aber sie nahmen die Herausforderung an und gelangten dahin, besser im Führen zu werden.

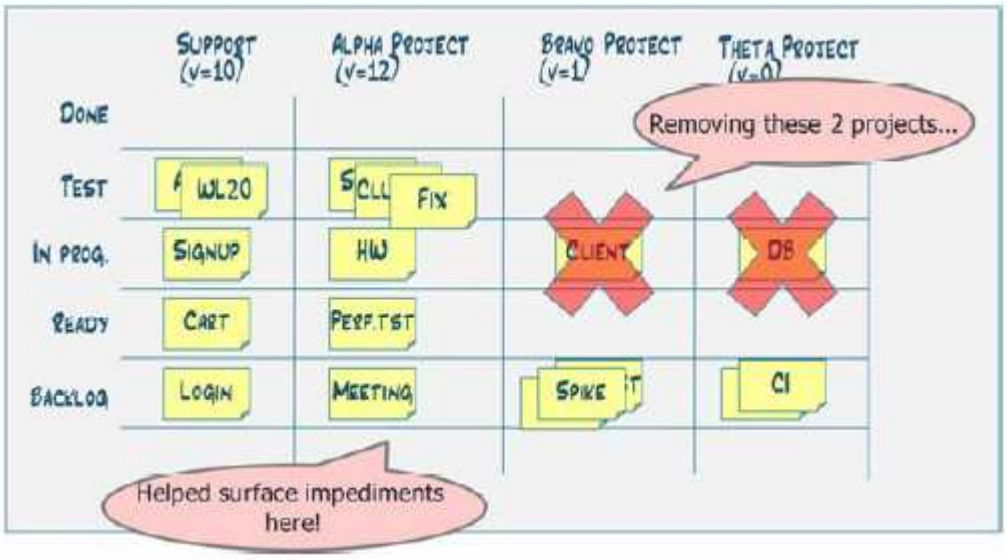
34 Grundlektionen

34.1 Wenn der Anteil der aktuell bearbeiteten Arbeitsaufgaben ansteigt, tauchen Hindernisse auf

Alle Teams fingen mit großzügigen WIP-Limits an. Zu dieser Zeit war die Energie meist dadurch gebunden zu versuchen, den Ablauf einzustellen, und sicherzustellen, dass die Organisation den Support bekam, den sie brauchte.

Zuerst wollten die Manager mehrere Projekte gleichzeitig haben, aber innerhalb weniger Wochen wurde offensichtlich, dass es nicht genug Kapazität gab, um sich mit niedriger priorisierten Projekten zu beschäftigen. Es erforderte nur einen kurzen Blick aufs Board, um zu sehen, dass Arbeit an niedrig priorisierten Dingen niemals begonnen wurde. Das brachte die Manager dazu, die Nummer der Projekte pro Team zu reduzieren.

Mit der Zeit, als der Ablauf für die hochpriorisierte Arbeit stabiler wurde, begannen wir, die WIP-Limits einzuengen. Das wurde durch die Reduzierung der Anzahl laufender Projekte (Spalten) von erst drei oder zwei auf dann eins realisiert. Als das geschah, begannen sich Hindernisse außerhalb des Teams zu zeigen. Teammitglieder begannen zu berichten, dass sie keine rechtzeitige Hilfe von Anderen bekamen, also richteten die Führungskräfte ihre Aufmerksamkeit darauf, sich damit zu beschäftigen.



Einige andere Dinge, die aufkamen, enthielten die Auswirkungen schlechter Inputs anderer Teams auf die Arbeitsleistung dieses Teams. Es war schwer, einen reibungslosen und schnellen Ablauf zu pflegen, wenn einkommende Arbeitspakete ständiger Korrektur bedurften.

Diese Probleme waren nicht sichtbar, bevor wir anfingen. Der Punkt war eher „welches Problem sollten wir uns zuerst vornehmen“ – und eine Einigung darüber zu erreichen. Mit dem Kanbanboard konnte Jeder sehen, wie ein spezielles Problem den *Ablauf* beeinträchtigte, was es leichter machte, in Schwung zu kommen und sich mit der Angelegenheit über organisatorische Grenzen hinweg zu beschäftigen.

34.2 Das Board wird sich auf dem Weg ändern, meißeln Sie es nicht in Stein

Alle Kanbanboards verändern sich unterwegs. Es brauchte üblicherweise zwei oder drei Neuentwürfe, bevor ein Team eins fand, das gut funktionierte. Also ist viel Zeit in das erste Layout zu stecken wahrscheinlich Verschwendung. Stellen Sie sicher, dass Sie das Board leicht umordnen können. Wir benutzten schwarze Klebebandstreifen für das Layout. Die waren leicht neu anzuordnen und konnten an Wänden wie an Whiteboards genutzt werden. Eine andere Möglichkeit, die ich gesehen habe, ist, am Board Gitterlinien zu ziehen mit dicken Markern (aber stellen Sie sicher, dass sie leicht weggewischt werden können! ☺)

Unten ist ein typisches Beispiel einer Gestaltungsoptimierung. Prioritäten verschoben sich anfangs regelmäßig, also um zu verhindern, eine ganze Spalte von Haftnotizen vor und zurück zu bewegen, klebte das Team stattdessen eine Prioritätenzahl über jede Spalte.

Abbildung 14. Frühes Kanbanboard mit Aufklebern für die aktuellen Prioritäten

34.3 Haben Sie keine Angst vorm Experimentieren und Scheitern

Die Lehre, die ich aus diesem Abenteuer gezogen habe, ist, dass es wirklich keinen Endpunkt gibt. Wir scheitern in dem Moment, wenn wir merken, es gibt einen. Es gibt nur endloses Experimentieren und Lernen. Niemals zu scheitern bedeutet nicht zu lernen. Wir scheiterten einige Male entlang des Wegs (schlechte Boardentwürfe, Schätzungen, überflüssige Burndowncharts etc.), aber jedes Mal lernten wir etwas Neues und Wichtiges. Wenn wir aufhörten es zu versuchen, wie könnten wir dann lernen?

Der Erfolg von Kanban hat jetzt die Managementteams und die Scrum-Entwicklungsteams inspiriert, auch mit Kanbanboards zu experimentieren. Vielleicht wird dieses Buch helfen!

Letzte Punkte zum Mitnehmen

Fangen Sie mit Retrospektiven an!

Viele Möglichkeiten und Dinge, um darüber nachzudenken, hm? Hoffe, dieses Buch half, einigen Nebel zu lichten. Zumindest funktionierte es für uns :o)

Wenn Sie daran interessiert sind, Ihren Prozess zu ändern und zu verbessern, lassen Sie uns genau jetzt eine Entscheidung für Sie treffen. Wenn Sie noch keine regelmäßigen Retrospektiven machen, fangen Sie damit an! Und stellen Sie sicher, dass sie zu einer echten Veränderung führen. Holen Sie einen externen Moderator, falls nötig.

Sobald Sie effektive Retrospektiven intus haben, beginnen Sie ihre Reise, einfach den richtigen Prozess für Ihren Kontext zu entwickeln – ob das auf Scrum, XP, Kanban, eine Kombination davon oder was auch immer sonst basiert.

Hören Sie nie auf zu experimentieren!

Kanban oder Scrum sind nicht das Ziel, kontinuierliches Lernen ist es. Eins der tollen Dinge bei Software ist die kurze Feedbackschleife, die der Schlüssel zum Lernen ist. Also nutzen Sie diese Feedbackschleife! Hinterfragen Sie alles, experimentieren Sie, scheitern Sie, lernen Sie, dann experimentieren Sie nochmal. Sorgen Sie sich nicht darum, es von Beginn an richtig hinzubekommen, weil sie das nicht schaffen werden! Fangen Sie einfach irgendwo an und entwickeln es von da an weiter.

Das einzig *echte* Scheitern ist daran zu scheitern, aus dem Scheitern zu lernen.

Aber hey, Sie können selbst daraus lernen.

Viel Glück und genießen Sie den Ritt!

/Henrik & Mattias, Stockholm 2009-06-24

H: Ist das alles, was wir haben?

M: Ich denke schon. Lass uns hier aufhören.

H: Vielleicht sollten wir ihnen erzählen, wer wir sind?

M: Guter Punkt. Wenn wir es so aussehen lassen, als ob wir nette Kerle sind, könnten wir Beratungsauftritte ergattern.

H: Dann lass es uns tun! Dann machen wir Schluss.

M: Ja, wir haben auch noch anderes zu tun, genau wie die Leser.

H: Und überhaupt, mein Urlaub beginnt ungefähr genau jetzt :o)

M: Hey, reib's mir nicht unter die Nase.

Über die Autoren

Henrik Kniberg und Mattias Skarin sind Berater bei Crisp in Stockholm. Sie helfen gern Unternehmen dabei, sowohl mit der technischen als auch mit der menschlichen Seite der Softwareentwicklung Erfolg zu haben, und haben Dutzenden von Unternehmen geholfen, Lean und Agile Prinzipien in ihrer Arbeit in die Praxis umzusetzen.

Henrik Kniberg

Während des letzten Jahrzehnts war Henrik Technischer Direktor von 3 schwedischen IT-Firmen und half vielen mehr, ihre Prozesse zu verbessern. Er ist zertifizierter Scrum-Trainer und arbeitet regelmäßig mit Lean- und Agil-Pionieren wie Jeff Sutherland, Mary Poppendieck und David Anderson.

Henriks letztes Buch, „Scrum and XP from the Trenches", hat über 150.000 Leser und ist eins der beliebtesten Bücher zu diesem Thema. Er hat mehrmals den Preis für den besten Sprecher für Vorträge auf internationalen Konferenzen gewonnen.

Henrik wuchs in Tokio auf und lebt jetzt mit seiner Frau Sophia und drei Kindern in Stockholm. In seiner Freizeit ist er aktiver Musiker, komponiert Musik und spielt Bass und Keyboard in lokalen Bands.

henrik.kniberg@crisp.se

http://blog.crisp.se/henrikkniberg

http://www.crisp.se/henrik.kniberg

Mattias Skarin

Mattias arbeitet als Coach für Lean, dabei hilft er Softwarefirmen, Spaß am Nutzen von Lean und Agil zu haben. Er ist als Mentor in allen Hierarchiestufen tätig, von Entwicklern bis zum Management. Er hat geholfen, bei einer Entwicklungsfirma für Spiele die Spielentwicklungszeit von 24 Monaten auf 4 zu reduzieren, brachte einer ganzen Abteilung die Vertrauenswürdigkeit wieder und war einer der frühen Pioniere von Kanban.

Als Unternehmer hat er zwei Unternehmen mit gegründet und leitet sie.

Mattias hat einen Master of Science in Qualitätsmanagement und hat 10 Jahre lang als Entwickler in geschäftskritischen Systemen gearbeitet.

Er lebt in Stockholm und genießt Rock'n'Roll, Tanzen, Rennsport und Skifahren.

mattias.skarin@crisp.se

http://blog.crisp.se/mattiasskarin

http://www.crisp.se/mattias.skarin