www.bt-magazin.de 1.10 Deutschland: € 14,90 Österreich: € 16,80 Schweiz: sFr 29,80 Luxemburg: € 16,90

Business Technology

Business Technology

Architektur & Management Magazin

Expertenwissen für IT-Architekten, Projektleiter und Berater

Erl: „SOA in a Box hat die Probleme verursacht"

Schwerpunkt:

Software Architektur

Businessorientierte Architekturansätze

Kanban in Softwareprojekten

Pragmatic SOA – Beschränken auf das Wesentliche

Servicesichten
Den passenden Service finden

People Change Management
Änderungsprozesse erfolgreich meistern

How does IT matter?
Den Wertbeitrag der IT bestimmen

Enterprise SOA Security
Teil 1: Die Herausforderungen

Aus alt mach neu
Die 10 Regeln der Modernisierung

4 191858 414906 01

Verbesserungsprozesse bringen echten Wettbewerbsvorteil

Kanban in der Softwareentwicklung

Während das agile Managementframework Scrum seinen Siegeszug ungehindert fortsetzt, macht in letzter Zeit ein Vorgehen namens „Kanban“ von sich reden. Kanban bietet durch seine Einfachheit und die kleinen Schritte, mit denen Änderungen durchgeführt werden, große Chancen für alle Unternehmen, die ihre Durchlaufzeiten verkürzen und die Qualität ihrer Software erhöhen wollen.

AUTOREN: ARNE ROOCK UND HENNING WOLF

Mary und Tom Poppendieck erzählen in ihrem Buch „Implementing Lean Software Development“ [1] die Geschichte zweier amerikanischer Versandhäuser: Versandhaus A, ein ausgesprochenes Traditionsunternehmen, das auf eine beinahe 100-jährige Geschichte zurückblicken kann, ist in den 1980er Jahren eigentlich konkurrenzlos in den USA. In jedem Haushalt liegen die dicken Kataloge, viele Familien bestellen dort regelmäßig ihre Kleidung. Versandhaus B kommt neu auf den Markt und tritt gegen den scheinbar übermächtigen Konkurrenten an. Dennoch dauert es nicht lange, und Versandhaus A muss aufgeben, weil es nicht mehr mit dem neuen Konkurrenten mithalten kann. Was ist passiert? Versandhaus B verspricht, jede Bestellung innerhalb von 24 Stunden zu liefern, während Versandhaus A nach wie vor 2 bis 3 Wochen benötigt. Und diese kurze Lieferzeit führt nicht nur zu einer höheren Kundenzufriedenheit, sondern auch zu wesentlich geringeren Kosten. Lange Lieferzeiten bedeuten in dem Fall nämlich, dass große Lagerflächen vorgehalten werden müssen. Und was noch entscheidender ist, die vielen halbfertigen Bestellungen führen zu einem riesigen Verwaltungsaufwand. Denn es muss immer wieder überprüft werden, in welchem Zustand sich welche Bestellung befindet. Darüber hinaus fragen die Kunden regelmäßig nach, wann ihre Bestellung wohl eintreffen wird – und falls sich ein Kunde umentscheidet, dass er doch lieber ein blaues statt eines grünen Shirts haben möchte, wird die Bestellung eben umgestellt, was wiederum Aufwand verursacht. Nun betreiben wir keine Versandhäuser, sondern erstellen Software. Aber für uns ist schnelle Lieferung mindestens ebenso wichtig wie für Versandhäuser. Dafür gibt es verschiedene Gründe.

1. *Früher Return on Investment:* Je schneller wir ausliefern, desto schneller bekommen wir unser Geld. Das ist eine Binsenweisheit. Allerdings halten wir uns selten daran. Und vor allem denken wir viel zu oft, wir könnten erst ausliefern, wenn das große ganze Gesamtpaket fertig ist. Das stimmt aber nicht. So wie Amazon heute auch Teile unserer Lieferung versendet, wenn andere Teile gerade nicht lieferbar sind, so können und sollten auch wir möglichst häufig Teile unserer Software ausliefern, auch wenn andere Teile noch fehlen. Das erste iPhone konnte kein Copy-and-Paste, obwohl dieses Feature bei den meisten anderen Smartphones schon zum Standard gehörte. Aber der Markt für Smartphones ist hart umkämpft, also war es sehr wichtig für Apple, sich früh zu positionieren. Und das vermeintlich fehlende Feature hat dem Erfolg keinen Abbruch getan und wurde später hinzugefügt.
2. *Positionierung am Markt:* Als Erster am Markt zu sein, bringt immer deutliche Wettbewerbsvorteile. Denn der Erste kann, so lange bis die Konkurrenten folgen, hohe Margen einstreichen. Und in vielen Fällen kann er darüber hinaus mit seinem Produkt Standards setzen, die sich nur sehr langfristig wieder ändern, etwa wenn es ihm gelingt, dass seine Software Teil eines größeren Produkts wird (z. B. der Steuerung eines Autos).
3. *Fast Follower:* Kurze Time-to-Market hat jedoch auch eine andere Dimension: Es muss nämlich gar nicht un-

bedingt angestrebt werden, als erster auf dem Markt zu sein. Häufig ist es mindestens ebenso lukrativ, schnell nachzuziehen und dafür das Original zu übertreffen, z. B. durch geringere Preise oder höhere Qualität. Diese kann ich deshalb bieten, weil ich später mit der Entwicklung begonnen habe und die gewonnene Zeit nutzen kann, um zusätzliche Informationen zu generieren, reifere Technologien einzusetzen usw.

4. *Kundenzufriedenheit:* Kunden wollen ihre neue Software schnell bekommen. Und dann akzeptieren sie in der Regel auch, wenn geforderte Features in späteren Versionen nachgeliefert werden. Schnelle, häufige und regelmäßige Lieferung verbessert das Vertrauensverhältnis zum Kunden und erleichtert die Zusammenarbeit.
5. *Schnelles Feedback:* Die beste Software entsteht immer dann, wenn wir schnell und häufig Feedback von den Anwendern bekommen. Dabei ist es gleichgültig, ob wir Inhouse-Entwicklung betreiben oder Produktentwicklung. In diesem Sinne ist es erstrebenswert, schnell auszuliefern, um sicherzustellen, dass wir die Kundenbedürfnisse auch wirklich treffen.

Die Argumente für schnelles und häufiges Ausliefern sind also bestechend. Es stellt sich jetzt nur die Frage, wie man das hinbekommt? Hierfür bietet Kanban gute Möglichkeiten. Kanban ist ein Vorgehen, das einfache, aber sehr effektive Mechanismen bereithält, um die Time-to-Market drastisch zu verkürzen.

WAS IST KANBAN?

Der Begriff „Kanban“ stammt ursprünglich aus der japanischen Automobilindustrie (Toyota Production System) und bedeutet übersetzt „Signalkarte“. Diese Signalkarten wurden verwendet, um die Produktion so zu steuern, dass jeweils nur so viele Zwischenprodukte hergestellt werden, wie die nachfolgende Station verbraucht. In der IT steht Kanban heute jedoch für ein Vorgehen, dass zwar grundlegende Prinzipien aus der Lean Production übernimmt, nicht jedoch konkrete Praktiken. Darüber hinaus bilden Erkenntnisse aus der Theory of Constraints [2] ein wichtiges Element in Kanban. Um ein erfolgreiches Kanban-System zu etablieren, sind vier Prinzipien entscheidend:

1. Visualisierung
2. Pull statt Push
3. Begrenzung paralleler Arbeit
4. Kaizen

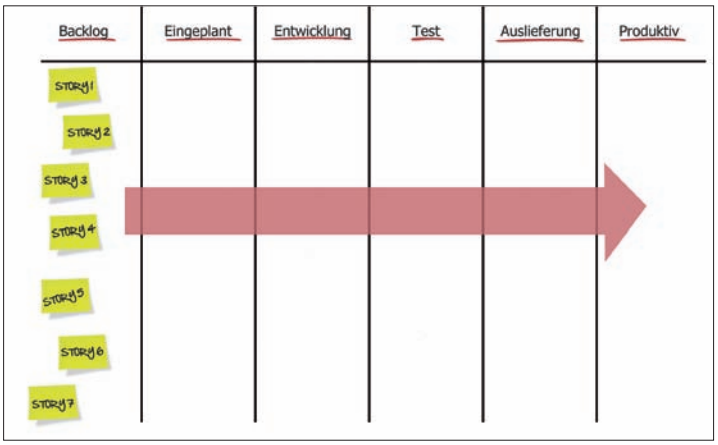

Abb. 1: Visualisierung einer einfachen Wertschöpfungskette. Die Anforderungen sind auf Haftnotizen festgehalten

VISUALISIERUNG

Ein wichtiger Punkt – und der erste Schritt bei der Einführung von Kanban besteht in der Visualisierung der bestehenden Wertschöpfungskette. Diese Kette könnte beispielsweise damit beginnen, dass Anforderungen in einem Dokument (Backlog) gesammelt werden. Dann gehen sie nach und nach in die Entwicklung, der eine nachgelagerte Testphase folgt. Und wenn keine Qualitätsmängel festgestellt wurden, werden die Anforderungen ausgeliefert. Für die Visualisierung wird meistens ein großes Whiteboard verwendet, auf dem jeder Prozessschritt als Spalte dargestellt wird. Die Anforderungen werden auf Haftnotizen oder Karteikarten notiert und durchlaufen als Tickets das Kanban-Board von links nach rechts. Bereits durch diese einfache Maßnahme lässt sich erkennen, wie lange welche Tickets benötigen, um das System zu durchlaufen und wo sich die Arbeit staut.

PULL STATT PUSH

Wesentlich für den Erfolg von Kanban ist, dass ein echtes Pull-System geschaffen wird. Das bedeutet, dass Arbeit niemals an nachgelagerte Prozessschritte übergeben werden darf – das wäre Push. Stattdessen zieht sich jeder Prozessschritt neue Arbeit bei seinem Vorgänger, sobald andere Arbeit erledigt wurde. Pull-Systeme verhindern Überlastung, gewährleisten ein nachhaltiges Arbeitstempo, und sie zeigen schnell Probleme auf und bieten so Möglichkeiten zur kontinuierlichen Verbesserung, insbesondere wenn gleichzeitig die Menge an paralleler Arbeit begrenzt wird.

BEGRENZUNG PARALLELER ARBEIT

Für die einzelnen Prozessschritte wird die Menge der Aufgaben, an denen gleichzeitig gearbeitet wird (Work in Progress) limitiert. Beispielsweise kann sich das Team

$$\text{Durchlaufzeit} = \frac{\text{Menge an paralleler Arbeit}}{\text{Durchsatz}}$$

Abb. 2: Little's Law

darauf einigen, dass nur vier Tickets gleichzeitig fest eingeplant werden sollen, sich niemals mehr als vier Tickets in der Entwicklung befinden dürfen, höchstens zwei beim Testen und maximal eins bei der Auslieferung. Diese Limitierung bewirkt, dass die einzelnen Tickets schneller erledigt werden, denn nach Little's Law definiert sich die Durchlaufzeit als Menge an paralleler Arbeit dividiert durch den Durchsatz. Wenn wir also die Durchlaufzeit verkürzen wollen, können wir entweder versuchen, den Durchsatz zu erhöhen. Oder wir können die Menge an paralleler Arbeit reduzieren. Das ist der Weg, der bei Kanban im Fokus liegt.

KAIZEN

Kaizen bedeutet, dass ein kontinuierlicher Verbesserungsprozess etabliert wird. Dieser beinhaltet, dass regelmäßige Abstimmungsmeetings stattfinden, auf denen die Teammitglieder über Prozessverbesserungen nachdenken. Es bedeutet außerdem, dass Fehler als notwendiges Übel und Chance zum Lernen begriffen werden, dass Fehler nicht kurzfristig korrigiert, sondern die Ursachen behoben werden, und dass jeder Mitarbeiter mit seinen Ideen und Verbesserungsvorschlägen willkommen ist.

ENGPÄSSE UND SINGLE PIECE FLOW

Das Pull-System in Verbindung mit der Begrenzung paralleler Arbeit führt dazu, dass auf dem Kanban Board sehr schnell deutlich wird, wo sich der Engpass befindet: In einer bestimmten Spalte auf dem Kanban Board werden sich nämlich schon bald die Tickets stauen, weil hier der Durchsatz geringer ist als anderswo. Die nachgelagerten Prozessschritte hingegen haben bald keine Aufgaben mehr. Und auch die vorgelagerten Schritte können irgendwann nicht mehr weiterarbeiten, weil sie ihre fertige Arbeit nicht weitergeben können und vom Engpass keine Arbeit mehr gezogen wird. Der Engpass erhält somit eine sehr große Aufmerksamkeit, und das gesamte Team ist gezwungen, hiermit umzugehen. Natürlich kann man einfach das Limit für den Engpass erhöhen – was kurzfristig auch sinnvoll sein kann. Weil aber der Durchsatz an dieser Stelle im Durchschnitt immer langsamer sein wird als anderswo, wird früher oder später wieder derselbe Stau auftreten.

Eine andere Möglichkeit besteht in der Aufstockung der Manpower beim Engpass, was aber mit Vorsicht zu verwenden ist, weil Skalierung erfahrungsgemäß nur begrenzt funktioniert. Eher zu empfehlen ist eine Umverteilung, sofern sie denn möglich ist: Beispielsweise könnte ein Entwickler als Tester arbeiten, auch wenn das nicht seiner Kernkompetenz entsprechen mag. In jedem Fall soll das Team gemeinsam überlegen, worin genau die Ursache dieses Engpasses besteht: Sind es wirklich zu wenig Personen an dieser Stelle? Ist es vielleicht eher ein Verfügbarkeitsproblem, weil die Tester noch in mehreren anderen Projekten arbeiten? Kann an bestimmten Aufgaben nicht weitergearbeitet werden, weil es technische Probleme gibt oder die Zuarbeit von Anderen nötig ist? Oder gibt es technische Möglichkeiten, um den Durchsatz zu erhöhen, z. B. automatisierte Akzeptanztests? In jedem Fall sollten alle Personen, die in einem Engpass arbeiten, von den Aufgaben entlastet werden, die auch von anderen erledigt werden können. Engpässe lassen sich also durch verschiedene Maßnahmen optimieren, beseitigen lassen sie sich allerdings nie. Denn ein Engpass ist immer nur relativ zu seiner Umgebung definiert, was bedeutet, dass an anderer Stelle sofort ein neuer Engpass auftaucht. Die Optimierung von Engpässen ist also eine permanente Herausforderung, will man die Durchlaufzeit kontinuierlich verkürzen.

Aus Kanban-Sicht ist als Ideal der Single Piece Flow anzustreben: Eine einzige Aufgabe fließt gleichmäßig von links nach rechts durch das gesamte Kanban-System, die Aufgabe ist zu jeder Zeit in Bearbeitung – ohne jegliche Wartezeiten. Das lässt sich natürlich in der Softwareentwicklung niemals realisieren, weil Features immer in der Größe variieren und Menschen ungleichmäßig schnell arbeiten. Aber das Bestreben sollte dahin gehen, diesem Ideal schrittweise möglichst nahe zu kommen, indem die Wartezeiten immer weiter verkürzt und in Folge auch die Limits reduziert werden.

UND DIE QUALITÄT?

Auch wenn es zuerst nicht gerade intuitiv klingt: Qualität ist kein Widerspruch zu Geschwindigkeit. Der Zusammenhang stellt sich genau andersherum dar: Wer wirlklich schnell sein will, erreicht das nur durch hohe Qualität. Denn eins der größten Hindernisse für hohe Geschwindigkeit ist das mehrfache Anfassen einer Aufgabe. Wenn beispielsweise in der nachgelagerten Testphase ein Fehler gefunden wird und das entsprechende Feature an die Entwicklung zurückgegeben werden muss, entstehen hierdurch erhebliche zusätzliche Aufwände: Die Entwickler müssen sich erneut in den Zusammenhang hineindenken, sie müssen ggf. ihre aktuelle Aufgabe liegenlassen und sich später erneut in diese einarbeiten, und sie müssen nun insgesamt mehr Aufgaben verwalten, z. B. im Tracking-System. Auch die Tester müssen die Aufgabe später ein zweites Mal anfassen. Wenn das gesamte Team den Fokus auf kurze Durchlaufzeiten legt und konsequent nach den Ursachen für Verzögerungen

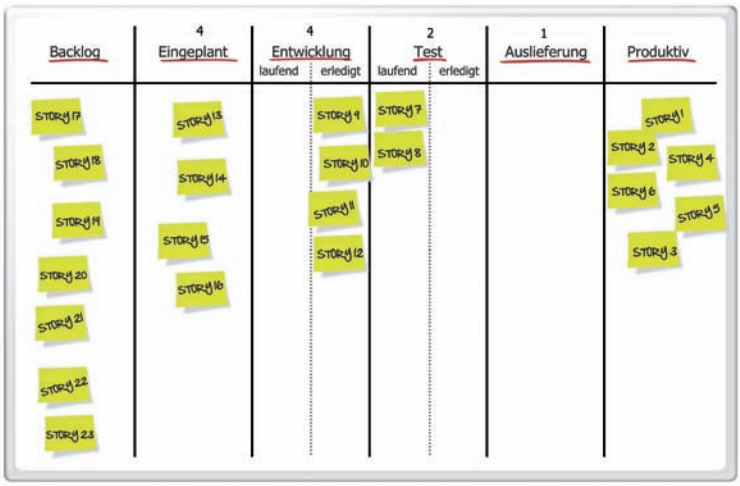

Abb. 3: Das Kanban Board macht deutlich, dass der Engpass beim Testen liegt. Bei der Entwicklung staut sich die Arbeit, während die Auslieferung nichts mehr zu tun hat

fragt, wird es schnell auf Qualitätsdefizite stoßen. Eine wirksame Maßnahme könnte darin liegen, automatisierte Akzeptanztests einzuführen oder die Tester mit in das Entwicklungsteam zu integrieren. Dadurch kann die Geschwindigkeit erhöht werden, und gleichzeitig erhöht sich auch noch die Qualität.

UMGANG MIT LEERLAUFZEITEN

Es wird bei Kanban immer wieder Teammitglieder geben, die temporär keine Arbeit haben. Das gilt es auszuhalten und nicht zu versuchen, die Auslastung zu optimieren. Das Schlimmste, was ein Projektleiter in diesem Fall tun kann, besteht darin, die entsprechenden Mitarbeiter in weitere Projekte zu stecken. Denn dadurch wird zwar die Auslastung optimiert, gleichzeitig entstehen aber große Verschwendung (Kontextwechsel, Verwaltung unterschiedlicher Projekte usw.), und die Engpässe werden in der Regel noch weiter verschlimmert. Viel sinnvoller ist es, dass der betreffende Mitarbeiter die freie Zeit dafür verwendet, um über die Optimierung des Engpasses oder generell über Prozessverbesserungen nachzudenken und konkrete Maßnahmen zu erarbeiten. Warum sollten Entwickler zwischendurch nicht mittesten, wenn hier gerade Bedarf besteht? Und es gibt so gut wie immer ein Refactoring, das schon lange liegen geblieben ist oder ein neues Framework, das sich die Entwickler ansehen wollten. Leerlaufzeiten sollten also nicht als Fehler begriffen werden, sondern als Chance zur Verbesserung.

KANBAN EINFÜHREN

Kanban führt man eher evolutionär statt revolutionär ein: Es sind nur kleine Änderungen nötig. Startpunkt ist stets die Visualisierung des eigenen Wertschöpfungsprozesses auf einem individuellen Kanban-Board. Dazu muss niemand seinen Jobtitel ändern, es müssen keine neuen Rollen geschaffen werden, es ist keine neue Technologie nötig, und man braucht auch keine neuen Tätigkeitsbereiche. Vielmehr werden die bestehenden Bereiche beibehalten und ggf. um Puffer ergänzt. Als nächstes wird ein echtes Pull-System etabliert und Kanban-Limits für die einzelnen Prozessschritte eingeführt. Für die Berechnung der Limits benötigt man keine mathematischen Formeln. Sinnvoller ist es, mit eher großen Limits zu beginnen und diese vorsichtig zu variieren, um die Effekte zu beobachten, die dabei auftreten. Spätestens jetzt werden Engpässe sichtbar, mit deren Optimierung man sich beschäftigen sollte. Durch Kaizen und systematische Reflexion wird das Kanban-System dann kontinuierlich verbessert, und es lassen sich immer kürzere Durchlaufzeiten erreichen.

FAZIT

Kurze Durchlaufzeiten stellen einen echten Wettbewerbsvorteil dar. Sie bedeuten nicht nur kurze Time-to-Market, sondern sie erhöhen auch die Kundenzufriedenheit, und der Fokus auf Geschwindigkeit macht außerdem deutlich, wie wichtig hohe Qualität ist. Kanban stellt ein einfaches Vorgehen dar, mit dem sich die Durchlaufzeiten immer weiter verkürzen lassen. Darüber hinaus wird durch Kanban, wenn es richtig verstanden wird, eine Kaizen-Kultur etabliert, in der jedes Teammitglied zur kontinuierlichen Verbesserung beitragen kann. Der besondere Charme von Kanban besteht darin, dass Änderungen in kleinen Schritten eingeführt werden und hierfür keine gravierenden Einschnitte erforderlich sind.

Dipl.-Inform. Henning Wolf ist Geschäftsführer der it-agile GmbH in Hamburg. Er verfügt über langjährige Erfahrung aus agilen Softwareprojekten (XP, Scrum, FDD) als Entwickler, Projektleiter und Berater. Er ist Autor der Bücher „Software entwickeln mit eXtreme Programming“ und „Agile Softwareentwicklung“. Henning Wolf hilft Unternehmen und Organisationen agile Methoden erfolgreich einzuführen.

Arne Roock arbeitet bei der it-agile GmbH in Hamburg. Als studierter Germanist interessiert er sich für informative, leicht verständliche und kooperative Kommunikation. Außerdem beschäftigt er sich seit Längerem mit den Themen Selbstorganisation und Zeitmanagement in der IT.

Links & Literatur

[1] Poppendieck, Mary und Tom: „Implementing Lean Software Development“, S. 34-35

[2] Goldratt, Eliyahu: „What is this Thing called Theory of Constraints?“, 1990

Business Technology

Architektur & Management Magazin

4 Ausgaben pro Jahr!

Jetzt Expertenwissen für IT-Architekten, Projektleiter und Berater abonnieren:

www.bt-magazin.de

Die Schwerpunkte des Hefts:

Enterprise Architecture Management | Cloud Computing, SOA, BPM | Agiles Projektmanagement | Change Management | Enterprise Integration | IT Governance | Open Source im Unternehmen