Ausgabe 2011
www.agilereview.de

agile review

Das Kundenmagazin von it-agile

...schneller als die Konkurrenz!


editorial

Es ist vollbracht! Sie halten die zweite Ausgabe der agile review in der Hand. Seit der letzten Ausgabe hat sich viel getan - sowohl intern bei uns als auch bei der Weiterentwicklung agiler Softwareentwicklung.
Freuen Sie sich auf eine spannende Lektüre!

In Konferenzen, Foren und auch in unserer Beratungsarbeit zeigte sich in diesem Jahr ein deutlicher Trend: Agilität hat die Geschäftsebene erreicht. Das ist am Interesse erkennbar, welches das obere Managements agilen Ideen inzwischen entgegenbringt. Und wir bekommen zunehmend Aufträge, Product Owner in agilen Teams zu unterstützen - ein Auftrag, der weit über rein methodische Unterstützung hinaus geht. Schließlich sind automatisierte Akzeptanztests ebenso Stützpfeiler agiler Projekte wie das Verständnis der Kundenbedürfnisse.
Erste Unternehmen gehen aber noch einen wesentlichen Schritt weiter: Wer mit einer bahnbrechenden Innovation an den Markt möchte, ist häufig zunächst gar nicht daran interessiert, ein eigenes Entwicklungsteam aufzubauen, denn schließlich ist der Markterfolg unsicher. Hier geht es vor allem darum, sich durch neue Software schnell Feedback vom Markt einzuholen. Funktioniert die Idee, muss die Software in der Lage sein, dem sich entwickelnden Markt flexibel zu folgen, ihn zu gestalten und den möglicherweise explodierenden Bedarf zu befriedigen, um Nachfolgern keine unnötigen Schwachstellen zu bieten. Scheitert die Idee, so möchte man dies möglichst früh wissen. Um solche Vorhaben optimal unterstützen zu können, haben wir gemeinsam mit einigen unserer Kunden die „Feedback Force“ entwickelt: Ein Komplettpaket aus Beratung, Entwicklung und Betrieb. Sie erhalten so die Möglichkeit, Ihre Geschäftsidee bei minimalem Risiko zu realisieren und sich schnell Feedback vom Markt einzuholen, während wir unsere führende Kompetenz für agile Ansätze nutzen, um Sie bei der Anforderungsdefinition zu unterstützen und Ihnen die Umsetzung abzunehmen. Ist die Idee erfolgreich, haben Sie Zeit, eigene Strukturen aufzubauen so dass wir Ihnen dann Know- How, Software und Betrieb geregelt übergeben. Mehr dazu finden Sie Im Artikel “Mit Feedback zur Innovation” auf Seite 52.

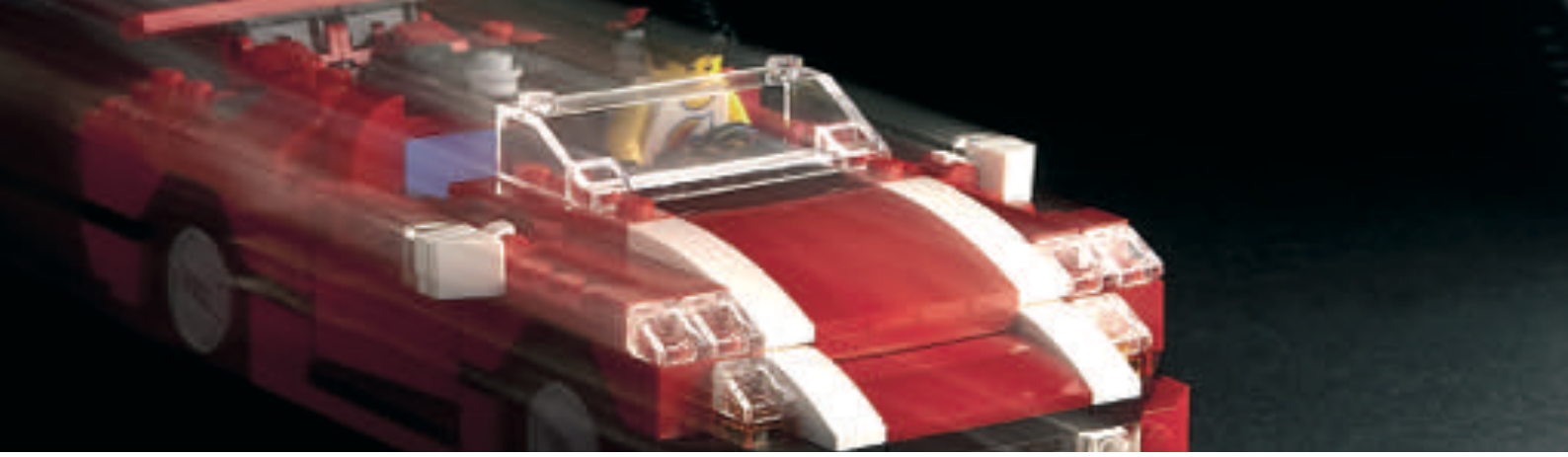
Aber auch, wer nicht so weit gehen möchte, kann sich die Vorteile agiler Entwicklung sichern. Die Erkenntnis, dass der gute alte Werkvertrag in der Softwareentwicklung mehr Probleme schafft, als durch ihn gelöst werden, setzt sich mittlerweile immer mehr durch. Agile Entwicklung bietet hier Alternativen. Vertragsmodelle, mit denen unsere Kunden und wir gute Erfahrungen in unseren Entwicklungsprojekten gemacht haben, beschreiben wir auf Seite 37.
Und schließlich hilft alle fachliche und methodische Unterstützung nichts, wenn letztlich doch qualitativ schlechte Software entsteht. Als Nebenprodukt unserer internen Trainings haben unsere Entwickler daher in diesem Jahr die Web-Site CodersDojo.org aufgebaut, die es einer immer größer werdenden Community erlaubt, „Katas“ online zu trainieren - also Übungen, um seine eigenen Fähigkeiten in testgetriebener Entwicklung zu verbessern. ■

Viel Spaß beim Lesen wünschen Ihnen

Henning Wolf, Geschäftsführer von it-agile, Hamburg

Jens Coldewey, Geschäftsführer von it-agile, München

Inhalt



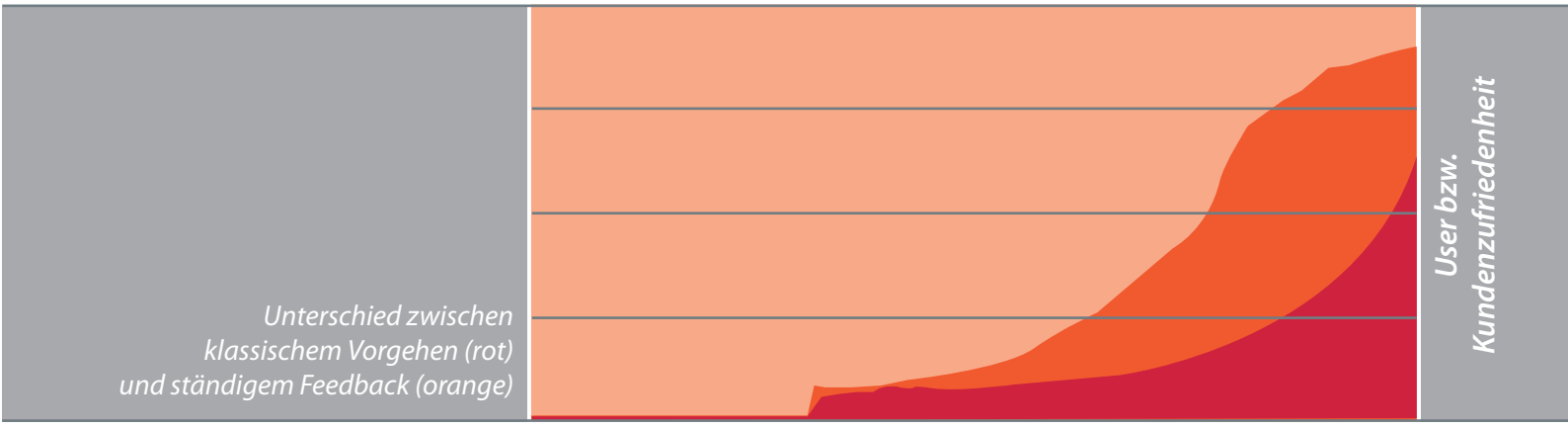
Schattenboxen: Warum Metriken nicht funktionieren können – und es trotzdem tun

Seit mehreren Dekaden sind Metriken Kennzeichen „wissenschaftlich fundierten Managements" – sowohl im Projektmanagement als auch in der Unternehmensführung. Indes legen Kritiker überzeugend dar, dass Metriken nicht funktionieren können. Dennoch kennt jeder Profi Praxisberichte über den erfolgreichen Einsatz von Metriken. Dieser Artikel versucht eine Erklärung hierfür zu finden. Von Jens Coldewey

metrikon
it-agile
Entwicklung, Beratung, Schulung

Risikominimierung durch ständiges Feedback

- Steht bei Ihnen ein riskantes Projekt ins Haus?
- Möchten Sie das Risiko minimieren, an den Bedürfnissen der Benutzer vorbei zu entwickeln?
- Arbeiten Sie in einer Domäne, die durch häufige Änderungen geprägt ist?
- Möchten Sie einen möglichst frühen Return on Investment?



Feedback Force

Wir unterstützen Sie mit einem kompletten Team, das

- in kurzer Zeit erste Versionen Ihres Systems an den Markt bringt
- dafür sorgt, ausreichend Feedback einzuholen, um wirklich das passende System zu entwickeln
- Ihnen bei der Definition und dem Zuschnitt von Anforderungen hilft
- Ihnen bei Wunsch das System für einen geregelten Betrieb übergibt

Treten Sie mit uns in Kontakt!

040 88173 300 **info@it-agile.de** **www.it-agile.de**

Kanban in der Softwareentwicklung

In letzter Zeit macht 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. Von 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. Diese kurze Lieferzeit führt nicht nur zu höherer Kundenzufriedenheit, sondern auch zu geringeren Kosten. Lange Lieferzeiten bedeuten 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, aus verschiedenen Gründen:

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 wichtig für Apple, sich früh zu positionieren. Und das vermeintlich fehlende Feature hat dem Erfolg keinen Abbruch getan und wurde später hinzugefügt.

Buchtipps

Welche Bücher sind wirklich hilfreich, wenn es darum geht, erfolgreich agile Vorgehensweisen zu etablieren, flexible und wartbare Software zu erstellen sowie innovative Produkte zu entwickeln?

Marktorientierte Innovation. Geniale Produktideen für mehr Wachstum.

Clayton M. Christensen und Michael E. Raynor (2004), Campus Verlag. 298 Seiten.

Mit marktorientierter Innovation meinen Christensen und Raynor die Innovationen, die zu innovativen vermarktbaren Produkten oder Dienstleistungen führen. In diesem Buch untersuchen sie die Mechanismen, die zu marktorientierter Innovation führen bzw. diese verhindern. Sie unterscheiden dazu evolutionäre von disruptiven Innovationen. Evolutionäre Innovationen gehen i.d.R. einher mit der schrittweisen Verbesserung existierender Produkte (z. B. leitungsfähigere Röhren). Disruptive Innovationen hingegen verdrängen existierende Produkte/Technologien (z. B. Transistoren im Vergleich zu Röhren).

Christensen und Raynor stellen auf dieser Basis zwei Kernthesen auf:

1. Es ist extrem schwierig, in einem profitablen Unternehmen eine disruptive Innovation nutzbar zu machen. Man macht damit seinen eigenen erfolgreichen Produkten Konkurrenz und stellt häufig auch etablierte Zielgruppenstrukturen und Vertriebswege in Frage.
2. Wenn man als Newcomer in ein Marktsegment eintritt, sollte man das nicht mit evolutionärer Innovation tun. Der Marktführer wird die Innovation schnell abkupfern, und der Markt kauft generell lieber beim Marktführer. Man braucht dafür disruptive Innovationen.

Diese Thesen stützen sie mit zahlreichen Studien über neue Produktideen und die Entwicklung von Unternehmen. Das Buch ist absolut empfehlenswert für alle, die sich mit der Definition/Entwicklung innovativer Produkte beschäftigen, beispielsweise Product Owner.

Stefan Roock

Im Coding Dojo

Perspektivenwechsel

Von Marko Schulz und Arne Roock

Bob und Jens, zwei alte Freunde, treffen sich abends in ihrer Stammkneipe.

Jens: Hi Bob! Wie siehst du denn aus? Du bist ja total durchnässt.

Bob: Hallo Jens! Das ist Schweiß. Ich komme gerade vom Karatetraining und hatte keine Zeit mehr zum Duschen.

Jens (rümpft die Nase und rückt etwas von Bob weg): Äh, ach so. Naja, so sehr hättest du dich dann ja auch wieder nicht beeilen müssen, um pünktlich zu kommen.

Bob (grinst): Du legst doch immer so viel Wert darauf, dass sich niemand verspätet.

Schon gut. Aber was habt ihr denn gemacht, dass du so sehr geschwitzt hast? Bretter durchgeschlagen? Autos poliert? Oder wieder Fliegen mit Essstäbchen gefangen?

Sehr witzig! Nein, wir haben Kata geübt. Heute war Gangaku dran – das bedeutet „Kranich auf dem Felsen". Eine sehr schöne Kata. Da steht man auf einem Bein, und dann...

Moment, ganz langsam. Was ist denn ein Kata?

Eine Kata ist ein stark formalisierter und stilisierter Kampf gegen mehrere imaginäre Gegner. Da ist genau festgelegt, wann man welche Technik machen muss, in welche Richtung man sich zu bewegen und welchen Stand man einzunehmen hat. Sogar die Form der Atmung und der Körperspannung ist geregelt – mal atmet man gleichmäßig aus und steigert die Körperspannung kontinuierlich an, mal atmet man explosionsartig aus und spannt alle Muskeln gleichzeitig an, und zwei Mal in jeder Kata macht man einen Kiai, das ist ein Kampfschrei.

Und wozu soll das gut sein? Wenn mich jemand auf der Straße angreift, weiß ich doch vorher auch nicht, wie genau der jetzt zuschlägt.

Das stimmt. Kata hat auch zuerst einmal sehr wenig mit Selbstverteidigung zu tun. Es geht darum, wichtige Grundprinzipien des Karate zu lernen: Spannung und Entspannung, die richtige Atmung, einen festen Stand, den korrekten Abstand zum Gegner, Wachsamkeit – und natürlich die korrekte Ausführung von Schlag- und Tritttechniken. Wenn man diese Prinzipien verinnerlicht hat, dann helfen sie einem auch bei der Selbstverteidigung.

Aha, das hört sich logisch an. Aber warum braucht man dafür eine genau festgelegte Form? Reicht es nicht, die Techniken einfach isoliert zu üben? Im Vorgehen oder einfach im Stand?

Das machen wir auch – nennt sich dann Kihon. Aber Kata ist anders. Jede Kata hat einen ganz eigenen Charakter. In jeder Kata werden ganz spezielle Aspekte des Karate geübt. Außerdem macht es Spaß, so eine Kata zu üben, die sind nämlich teilweise schon sehr alt. Katas werden auch auf Wettkämpfen und bei Gurtprüfungen vorgeführt. Weil die Bewegungsabläufe genau festgelegt sind, kann man dadurch unterschiedliche Karateka sehr gut vergleichen. Als fortgeschrittener Schüler macht man sich dann übrigens daran, diese festen Formen wieder etwas aufzulösen und verschiedene Anwendungen auszuprobieren. Das nennt man dann Bunkai.

Moment mal. Wie viele Katas gibt es denn? Das müssen doch hunderte sein, wenn man die jahrelang übt?!

27.

Du meinst 270?

Nein, im Shotokan-Karate gibt es tatsächlich nur 27 Katas. Und während der ersten acht bis zehn Jahre seiner Karatelaufbahn lernt man nicht mehr als etwa zehn davon.

Im Ernst? Das hört sich ja total öde an.

Das ist ja gerade der Witz: Man soll die Bewegungsabläufe immer und immer wieder üben. Irgendwann sind sie in Fleisch und Blut übergegangen, und man kann sie

Flexible Architekturen

Früher oder später werden Softwaresysteme unwartbar. Änderungen sind dann faktisch unbezahlbar. Irgendwann werden die Schmerzen so groß, dass ein neues System entwickelt wird. Und das Spiel beginnt von vorne. Dabei handelt es sich aber nicht um ein Naturgesetz. Wir können Software entwickeln, die sich langfristig zu moderaten Kosten weiterentwickeln lässt – und das, obwohl wir auch bei der initialen Entwicklung nur moderate Kosten verursachen. Von Stefan Roock

Der Schlüssel zum Erfolg sind flexible Softwarearchitekturen. Sie sorgen dafür, dass sich das System leicht an die jeweiligen Anforderungen anpassen lässt. Dieser Artikel beschreibt Grundprinzipien flexibler Softwarearchitekturen, die sich in vielen Projekten bewährt haben. Sie sind also keinesfalls neu. Wir, die Softwareindustrie, haben sie nur viel zu lange ignoriert und mit Füßen getreten. Dafür sollten wir uns schämen!

Vergurkte Systeme

Ich habe meine ersten Erfahrungen mit agiler Softwareentwicklung 1999 gemacht – konkret mit eXtreme Programming (XP). Wir waren eine Gruppe junger, hochmotivierter Entwickler, die alle bereits einige Jahre Berufserfahrung hatten. Die Ansätze aus XP fühlten sich großartig an: Hochqualitative Software entwickeln, testen, Refactoring, Pair Programming. Tatsächlich waren wir mit XP sehr erfolgreich. Wir lieferten nützliche Software an unsere Kunden. Und wir lieferten schnell. Das war ziemlich cool!

Wir hatten das Gefühl, jedes Projekt erfolgreich abwickeln zu können. Die Projekte hatten eine kurze Laufzeit von drei bis sechs Monaten und überschaubare Teamgrößen von maximal sechs Entwicklern. Mit dem Erfolg wurden die Projekte größer. Nun hatten wir es häufiger mit zweistelligen Teamgrößen zu tun, und die Projekte liefen über Jahre. Jetzt wurde es schmerzhaft. Die Entwicklungskosten stiegen immer weiter an. Dazu kam Ärger mit den Kunden wegen verpasster Termine und Fehlern im System. Für diese Probleme gab es zwei Ursachen: Erstens sind wir zu schnell vorgegangen, und zweitens haben wir uns zu wenig Gedanken über unser Tun und die Strukturentwicklung des Systems gemacht. Wir hatten einen zu starken Fokus darauf, möglichst schnell zu liefern. Wir waren waghalsig. Zweitens waren wir unwissend. Die Universität hatte uns nicht ausreichend ausgebildet, um nicht-triviale Softwaresysteme zu entwickeln ebenso wenig wie die Unternehmen, in denen wir während und nach dem Studium gearbeitet hatten. XP hatte zwar Referenzen auf Techniken wie testgetriebene Entwicklung und Refactoring gesetzt, aber ausreichend verstanden hatten wir diese nicht. ▶

XP DAYS GERMANY
ALL ABOUT AGILE

17.-19. November 2011
in Karlsruhe

KONFERENZ ZU AGILER SOFTWREENTWICKLUNG & EXTREME PROGRAMMING

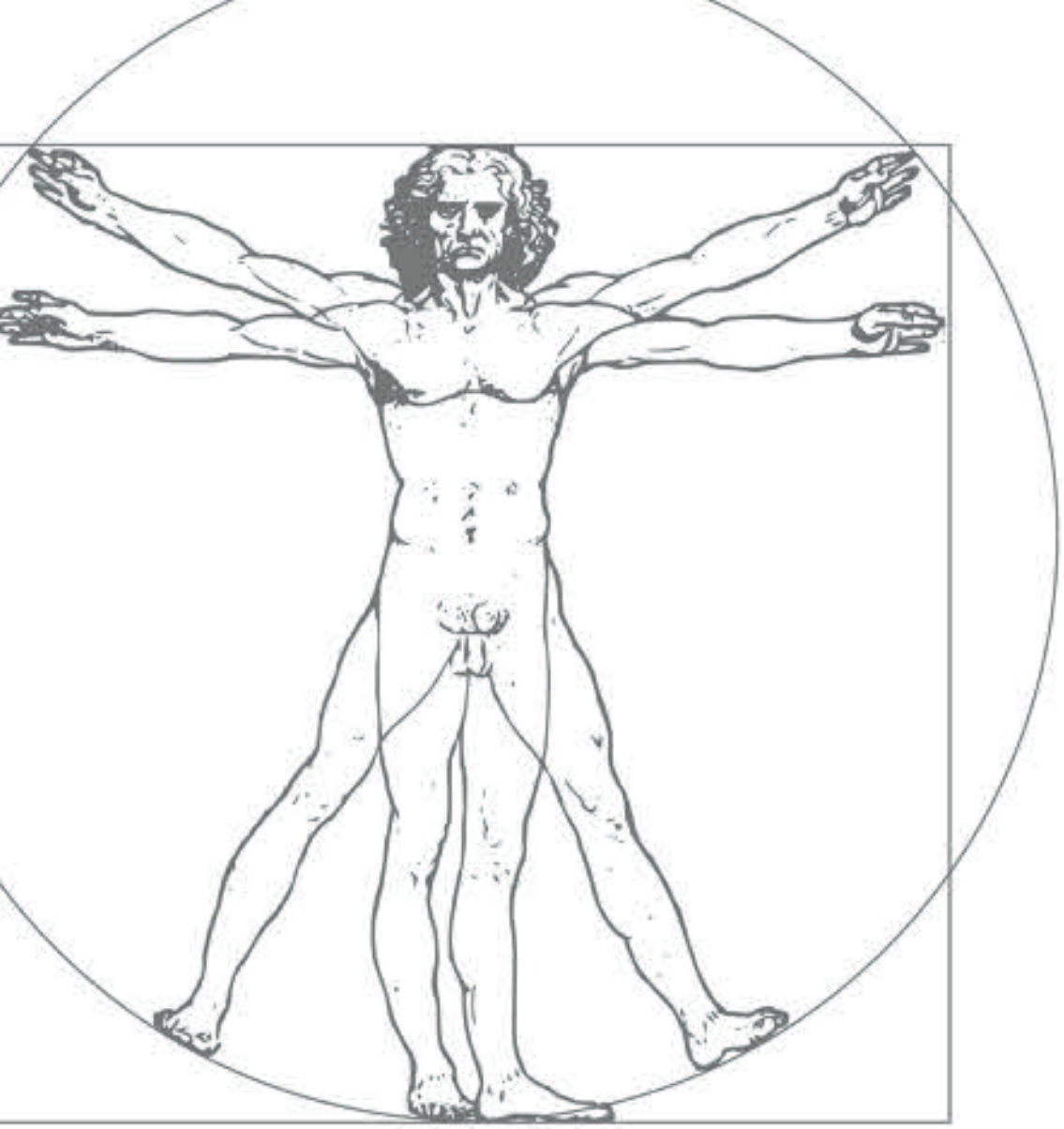
http://www.xpdays.de

Agil
Erfahrungen
Kanban
Kontakte
Lean
OpenSpace
Retrospektiven
Scrum
Soft Skills
Spaß
TDD
XP

& Vorträge, Tutorials
Diskussionsrunden
Community Day

it-agile

andrena OBJECTS



Der ganzheitliche Entwickler

Certified Scrum Developer (CSD)

von Andreas Havenstein

Als Mitte der Neunziger Jahre schwergewichtige, dokumentlastige Prozesse den Markt dominierten, gab es eine Gruppe von Softwareentwicklern und Projektmanagern, die der Trägheit der Wasserfallprozesse eine leichtgewichtige Alternative entgegenbieten wollten.

Kent Beck beispielsweise erkannte, dass die bis zum Maximum gesteigerte Anwendung bewährter Engineering-Praktiken in Kombination mit vielen kurzen Feedback-Schleifen ein Softwareprojekt zum Erfolg führen kann. Er fasste Entwicklungspraktiken zu Extreme Programming (XP) zusammen und schuf den ersten großen Vertreter dessen, was später als Agile Softwareentwicklung bezeichnet wurde [1].

Scrum entstand zu ähnlicher Zeit wie XP und wurde Anfang des neuen Jahrtausends bekannter, als Ken Schwaber ein erstes Buch hierzu veröffentlichte [2]. In den darauffolgenden Jahren setzte ein so großer Scrum-Boom ein, dass Agile Softwareentwicklung und Scrum für viele synonym wurden. Entwicklungspraktiken, in XP noch dominierender Bestandteil, rückten in den Hin-

Hochwertige Software - zufriedene Mitarbeiter und Kunden

Roman Pichler spricht über agiles Produktmanagement, die Herausforderungen der Product-Owner-Rolle und die Skalierung von Srum.

das Interview führte Arne Roock

Du hast gerade dein zweites Buch veröffentlicht, das 2011 auf Deutsch erscheinen wird. Das Buch handelt von Produktmanagement mit Scrum. Warum ist das ein eigenes Thema?

Mein erstes Buch „Scrum – Agiles Projektmanagement erfolgreich einsetzen" möchte dem Leser die richtige Anwendung von Scrum vermitteln. Dabei hatte ich als Zielgruppe primär Entwicklungsleiter, ScrumMaster sowie Team- und Projektleiter im Auge. Mein neues Buch „Agile Product Management with Scrum" wendet sich an Product Owner, Produktmanager, Manager im Produktmarketing und Business-Analysten. Es setzt Grundkenntnisse in Scrum voraus und zeigt als erstes Buch die richtige Anwendung agiler Produktmanagement-Praktiken in Scrum.

Agiles Produktmanagement ist ein wichtiges Thema, da Scrum mit dem Product Owner eine neue Rolle definiert, die in traditionellen Prozessen nicht existiert. Darüber hinaus verändert der Einsatz von Scrum die frühen Aktivitäten bei der Entstehung des Produkts, etwa Marktforschung und Produktplanung, nachhaltig.

Wo genau liegen eigentlich die Unterschiede zwischen einem klassischen Produktmanager und einem Product Owner in Scrum?

Traditionell kümmern sich Produktmanager um die frühen Phasen der Produktentstehung wie Marktforschung, Produktplanung und Formulierung einer Marketing-Strategie. Diese Arbeiten kulminieren in einer Anforderungsspezifikation, die versucht, das neue Produkt umfassend und detailliert zu beschreiben. Liegt diese vor, wird das Projekt meist an einen Projektleiter übergeben, der sich um die Umsetzung der Spezifikation in ein Produkt kümmert. Der Produktmanager fokussiert sich in dieser Phase normalerweise auf das Managen von Change Requests und die Vorbereitung der Markteinführung. Er arbeitet daher nicht besonders eng mit dem Team zusammen.

Die Product Owner-Rolle ist eine gänzlich neue Rolle, auch wenn es Überschneidungen mit traditionellen Produktmanagementaufgaben gibt. Der Product Owner ist verantwortlich für den Erfolg des Produktes. In dieser Rolle erstellt er die Produktvision, managed die Product Roadmap, pflegt das Product Backlog, beplant das Release, arbeitet eng mit ScrumMaster und Team über die gesamte Projektlaufzeit zusammen, nimmt an den Scrum-Besprechungen teil und hält Kontakt zu Kunden, Anwendern und andern Interessenvertretern. Dabei nutzt der Product Owner neue Techniken, wie zum Beispiel das progressive Verfeinern von Anforderungen oder das Beschreiben von Features in Form von User Stories.

Agile Projekte beauftragen

Damit agile Projekte erfolgreich sein können, muss auch die Auftraggeberseite ihren Teil beitragen. In diesem Zusammenhang ist die Wahl der richtigen Beauftragungsform ebenso wichtig wie die konkrete Ausgestaltung des Vertrags. Dieser Artikel beschreibt die Herausforderungen bei der Beauftragung agiler Projekte.

von Stefan Roock und Henning Wolf

Vorteile agiler Projekte für den Auftraggeber

Ein neues Vertragsmodell oder eine neue Vorgehensweise stellt immer eine Hürde dar. Warum sollte ein Auftraggeber über diese Hürde springen? Was bringt ihm ein agiles Projekt? Wir sehen hier drei wesentliche Vorteile:

- Transparenz des Projektfortschritts
- Flexibilität (im Umgang mit den Anforderungen)
- Frühe (und häufige) Systemauslieferung

Aber welchen Nutzen bringen diese Vorteile konkret?

Transparenz des Projektfortschritts.

Eigentlich könnte es mir als Kunde egal sein, wo das Projekt gerade steht. Denn klassisch vereinbare ich mit dem Auftragnehmer einen Termin, zu dem die gesamte Software fertig ist. Zu diesem Termin habe ich Schulungsräume reserviert, eine Urlaubssperre verhängt und den Vertrag für die Wartung des alten Systems gekündigt. Vielleicht habe ich meinen Kunden sogar schon die Vorteile des neuen Systems angekündigt. Bei genauerem Hinsehen sollte es mich aber sehr wohl interessieren, wo mein Projekt gerade steht. Und sei es nur deshalb, um rechtzeitig zu wissen, dass sich der Termin verschiebt oder sich der Funktionsumfang zum vereinbarten Termin reduziert. Denn wenn es solche Schwierigkeiten gibt, will ich das natürlich möglichst früh mitbekommen, und nicht erst zwei Tage vor dem geplanten Termin!

Auch andere Probleme würde ich gerne früh erkennen, z. B. ob der Anbieter meine Anforderungen verstanden hat und so umsetzt, dass sie gut für mich passen und das neue System leicht benutzbar ist.

Ich profitiere als Kunde also von Transparenz, weil ich früher zu wichtigen Erkenntnissen kommen kann. Die nützen mir aber nur, wenn ich auch in der Lage bin, noch rechtzeitig auf diese neuen Erkenntnisse zu reagieren.

Flexibilität (im Umgang mit den Anforderungen)

Auf Erkenntnisse reagieren zu können, bedeutet Flexibilität. Für den Lieferanten ist Flexibilität wichtig, um bei ▶

Test Test Test

Wartbare Akzeptanztests

von Markus Gärtner

Bei der Programmierung von Software halten regelmäßige Refactorings den Quellcode anpassbar und flexibel. Wie aber soll das Team mit seinen funktionalen Akzeptanztests umgehen? Automatisierte Tests müssen sowohl Änderungen an den Anforderungen als auch am Produktionscode standhalten können, um dem Team nicht als Hindernis im Weg zu stehen. Da Testautomatisierung ebenfalls Software-Entwicklung ist, liegt die Anwendung vergleichbarer Prinzipien nahe. Hierdurch können Tests so automatisiert werden, dass sie auch mit geringem Aufwand wartbar bleiben und somit langfristig zum Erfolg des Teams beitragen.

Ein Anwendungsbeispiel

Das Managementteam eines internationalen Flughafens ist auf Probleme mit einer Applikation auf ihrer Internetpräsenz aufmerksam geworden. Die Internetseite bietet Passagieren die Möglichkeit, Parkkosten vorab zu berechnen. Kürzlich ist eine Gruppe von Testern [1] auf die Applikation gestoßen und hat diverse Probleme identifiziert. Alarmiert durch die entdeckten Probleme und die damit verbundene Schädigung des Ansehens, verlangt das Management einen Bericht darüber, wie schlimm es wirklich um den Parkkostenrechner bestellt ist. Auf dieser Grundlage soll entschieden werden, ob die Fehler gezielt behoben werden oder ob die Applikation komplett neu entwickelt werden muss.

Das Management beschließt, dass das Softwareteam automatisierte Tests für den Parkkostenrechner erstellen soll. Für die Parkkosten gibt es fünf verschiedene Parkbereiche: Einen Kurzzeitparkplatz, einen überdachten und einen nicht-überdachten Langzeitparkplatz, verbilligte Economy-Parkplätze und Parkplätze mit Parkbetreuung. Für alle fünf gelten unterschiedliche Parkkosten mit Tages- und Wochenlimits. Im Folgenden werden nur die Kosten für nicht-überdachtes Langzeitparken betrachtet:

- Kosten pro Stunde: $ 2,00
- Tägliches Maximum: $ 10,00
- Kosten pro Woche: $ 60,00 (siebter Tag kostenfrei)

Die Eingabemaske für die derzeitige Applikation ist in Abbildung 1 zu sehen.

Die Maske beinhaltet eine Auswahlliste mit den Parkmöglichkeiten, Eingabefelder für den geplanten Parkbeginn und das geplante Ende der Parkzeit, ein Feld zur Darstellung der berechneten Kosten sowie einen Knopf zur Berechnung der Kosten für die eingegebenen Werte.

Story Maps – Landkarten für das Backlog

Der Product Owner in Scrum könnte sich allein gelassen vorkommen, wenn es darum geht, das Product Backlog zu pflegen. Story Maps bieten Unterstützung. Sie helfen sowohl, Benutzerbedürfnisse zu berücksichtigen, als auch, den Überblick über das Backlog zu behalten. Von Alex Bepple

Die ureigenste Aufgabe des Product Owner in Scrum ist, das Product Backlog zu pflegen. Das kann ganz schön herausfordernd sein. Einerseits muss er sich an der Produkt-Vision orientieren und darf die großen Themen nicht aus dem Blick verlieren. Andererseits muss er auch sprintbare User Stories erstellen und dafür die Anforderungen so weit zerkleinern, dass an ihnen unmittelbar gearbeitet werden kann und sich das Team nicht erst fragen muss, was sie fachlich bedeuten. Der Product Owner muss daher andauernd den Spagat zwischen verschiedenen Detailniveaus meistern und darf dabei den Überblick nicht verlieren.

Wie man das schafft, verrät Scrum nicht – und das ist auch gut so. Der Product Owner darf jedes Verfahren einsetzen, das ihm hilft, und hat die Freiheit, seine Arbeitstechnik im Laufe der Zeit zu verbessern.

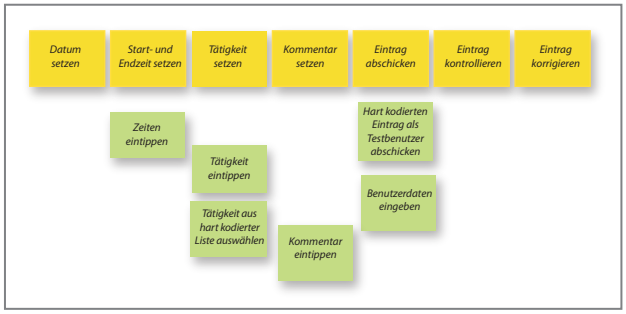
Nur – am Anfang kann das herausfordernd sein. Da mag sich der frisch gebackene Product Owner fragen: Wie arbeite ich konkret mit dem Backlog?

Was sind Story Maps?

Story Maps ([1], [2]) sind ein Satz von Techniken, die dem Product Owner konkrete Vorschläge machen, wie er sich die fachliche Domäne erschließen kann. Die zentrale Losung lautet: Orientiere dich am Arbeitsfluss der Benutzer. – Da überrascht es nicht, dass Jeff Patton, der Urheber von Story Maps, sich seit Langem mit Usability und Interaction Design beschäftigt.

Kombiniert man diese Vorgabe mit dem Grundprinzip

Abbildu ung 1: Eine sehr einfache Story Map



Mit Feedback zur Innovation

Warum wir gut daran tun, uns früh und häufig Feedback vom Markt einzuholen. Und warum auch frühes Scheitern wertvoll sein kann.

von Henning Wolf und Arne Roock

Was haben das Telefon, die Musikkassette und der Online-Dienst flickr gemeinsam? Bei allen drei handelt es sich um Erfindungen, die ursprünglich zu einem anderen Zweck gedacht waren, dann aber in ihrer jetzigen Form erfolgreich wurden. Bell wollte mit seinem Telefon ursprünglich Konzerte übertragen; die Musikkassette sollte eigentlich nur zum Diktieren von Briefen dienen; und flickr startete als Feature eines Multiplayer-Online-Spiels, mit dem User ihre Fotos in einem Chatraum austauschen konnten. Die wahren Innovationen, und damit der Erfolg, entstanden hier nicht (nur) durch eine geniale Idee, sondern dadurch, dass die Erfindungen in innovativer Weise genutzt wurden und dass die Erfinder sich auf die geänderten Gegebenheiten eingelassen haben.

Feedback, Feedback, Feedback

Was können wir daraus lernen? Natürlich möchten wir auch innovative Produkte oder Dienstleistungen entwickeln. Wer hätte nicht gern Twitter oder den iPod erfunden? Was wir dazu (neben einer Idee) vor allem brauchen, ist Feedback. Denn durch Feedback erhalten wir das Wissen, das wir dringend benötigen, um unser Produkt in die richtige Richtung zu entwickeln. Theoretisch kann man sich dieses Wissen natürlich auch auf anderem Wege beschaffen - indem man beispielsweise sämtliche vorhandene Literatur studiert, gründlich nachdenkt, dann den Plan für das perfekte Produkt aufstellt und mit der Umsetzung anfängt. Bei dieser “wissenschaftlichen Methode” (die ja in der Realität gar nicht so selten anzu- ▶

Meilensteine loswerden – Vom mühsamen Weg einer Scrum-Einführung

Von Jens Coldewey und Dr. Uwe Henker

1 Einleitung

Agile Verfahren wie Scrum verzichten sehr bewusst auf Meilensteine, um Risiken besser erkennen, steuern und ausräumen zu können. Bei der Einführung agiler Verfahren stellt das Phasenmodell im Kopf von Mitarbeitern und Management aber oft eine erhebliche Hürde dar. Dieser Artikel zeichnet den Weg eines Teams weg vom Meilensteindenken hin zu agilem Vorgehen nach.

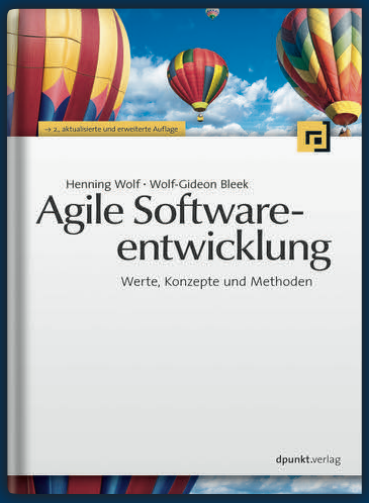
2 Das Umfeld

DOCexpert zählt zu den führenden Anbietern von Praxismanagementsystemen für Arztpraxen. Dies umfasst unter anderem Terminverwaltung, Patientenakten, Medikamentenverschreibung und Abrechnung mit Krankenkassen und Patienten. Die Domäne ist geprägt von hoher fachlicher Komplexität. Dazu kommen immer härtere Anforderungen an die Reaktionsgeschwindigkeit: Auch komplexe Änderungen werden zum Teil nur sechs Wochen vor Inkrafttreten bekannt gegeben. Die Entwicklung besteht aus etwa 45 Entwicklern und Testern sowie acht Mitarbeiterinnen im Produktmanagement. Hinzu kommen Hotline, Vertrieb und Verwaltung, so dass DOCexpert mit ca. 150 Mitarbeitern ein typisches mittelständisches Produkthaus darstellt.

3 Der Entschluss

Dr. Henker erzählt: Als ich 2007 als neuer Entwicklungleiter zu DOCexpert kam, fand ich das Team in einem desolaten Zustand vor: Produktmanagement, Entwicklung und Tester arbeiteten im Wesentlichen gegeneinander. Die Gruppen

dpunkt.bücher für Softwareentwickler



NEU

Henning Wolf, Wolf-Gideon Bleek
Agile Softwareentwicklung
Werte, Konzepte und Methoden

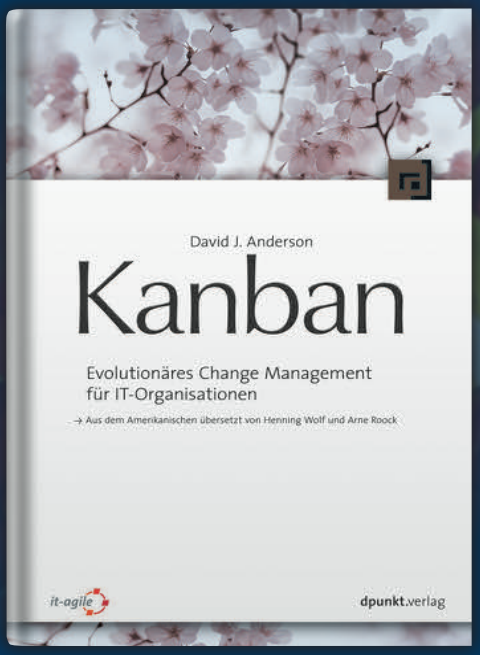
2., akt. und erw. Auflage

2010, 216 Seiten, Broschur
€ 29,90 (D) · ISBN 978-3-89864-701-4

„Übungsaufgaben und die Ausrichtung für das Selbststudium als auch für die konkrete Anwendung und Umsetzung in einem realen Projekt machen Agile Softwareentwicklung. Werte, Konzepte und Methoden zu einem klaren Leitfaden für das Verständnis einer Idee, die Softwareentwicklung in eine neue Perspektive rückt."

Wolfgang Treß, textico.de zur 1. Auflage

Die 2. Auflage wurde komplett aktualisiert. Neu hinzugekommen ist außerdem Software-Kanban, eine junge agile Methode, die aus den Ideen der Lean Production (Toyota-Production-System) entstanden ist.

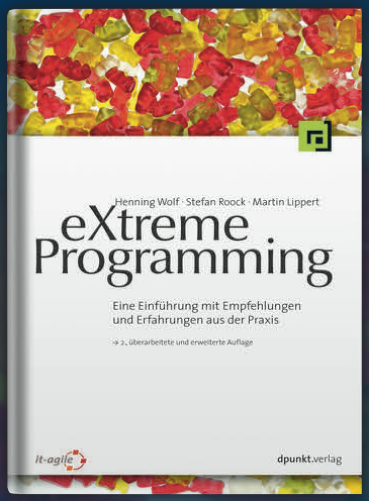


VORSCHAU

David J. Anderson
Kanban

Evolutionäres Change Management für IT-Organisationen

1. Quartal 2011, ca. 200 Seiten, Broschur
ca. € 34,90 (D) · ISBN 978-3-89864-730-4



E-BOOK

Henning Wolf, Stefan Roock, Martin Lippert
eXtreme Programming

Eine Einführung mit Empfehlungen und Erfahrungen aus der Praxis

2., überarb. und erw. Auflage

2005, 367 Seiten
€ 23,90 (D) · ISBN 978-3-89864-994-0
www.dpunkt.de/buecher/3514.html

VORSCHAU

Roman Pichler , Stefan Roock (Hrsg.)
Agile Entwicklungspraktiken mit Scrum

1. Quartal 2011, ca. 200 Seiten, Broschur
ca. € 34,90 (D) · ISBN 978-3-89864-719-9

dpunkt.verlag

Ringstraße 19 B · D-69115 Heidelberg · fon: 0 62 21 / 14 83 40 · fax: 0 62 21 / 14 83 99 · e-mail: bestellung@dpunkt.de · www.dpunkt.de