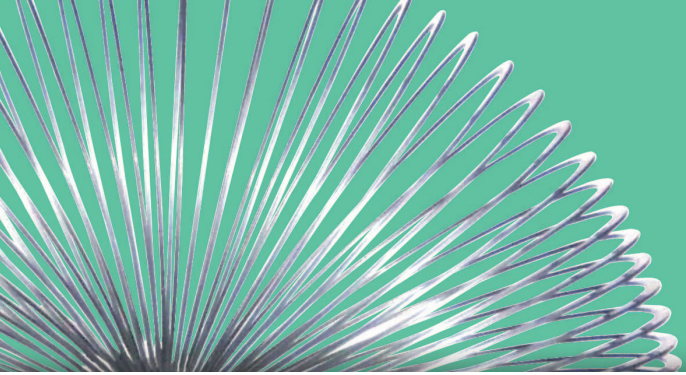


Henning Wolf · Stefan Roock · Stefan Zumbrägel

Agile Softwareentwicklung

Ein Überblick

5., aktualisierte Auflage

it-agile

dpunkt.verlag

Henning Wolf
henning.wolf@it-agile.de
Tel.: 01 72/4 29 76 20

Stefan Roock
stefan.roock@it-agile.de
Tel.: 01 72/4 29 76 17

Stefan Zumbrägel
stefan.zumbraegel@it-agile.de
Tel.: 01 72/4 29 76 18

it-agile GmbH
Willy-Brandt-Str. 1
20457 Hamburg
www.it-agile.de

5. Auflage 2023
Copy Editing: Ursula Zimpfer, Herrenberg
Satz und Herstellung: Frank Heidt
Umschlaggestaltung: Anna Diechtierow, Heidelberg
Umschlagfoto: Jasna Wittmann, Geesthacht
Druck: wörmann PRODUCTION CONSULT, Heidelberg

Artikel-Nr. 077.95720

Copyright © 2023 dpunkt.verlag GmbH
Wieblinger Weg 17
69123 Heidelberg

Die vorliegende Publikation ist urheberrechtlich geschützt. Alle Rechte vorbehalten. Die Verwendung der Texte und Abbildungen, auch auszugsweise, ist ohne die schriftliche Zustimmung des Verlags urheberrechtswidrig und daher strafbar. Dies gilt insbesondere für die Vervielfältigung, Übersetzung oder die Verwendung in elektronischen Systemen.
Es wird darauf hingewiesen, dass die im Buch verwendeten Soft- und Hardware-Bezeichnungen sowie Markennamen und Produktbezeichnungen der jeweiligen Firmen im Allgemeinen warenzeichen-, marken- oder patentrechtlichem Schutz unterliegen.
Alle Angaben und Programme in diesem Buch wurden mit größter Sorgfalt kontrolliert. Weder Autor noch Verlag können jedoch für Schäden haftbar gemacht werden, die in Zusammenhang mit der Verwendung dieses Buches stehen.
5 4 3 2 1 0

Henning Wolf · Stefan Roock · Stefan Zumbrägel

Agile Softwareentwicklung

Ein Überblick

5., aktualisierte Auflage

dpunkt.verlag

it-agile

Agil in Management

Ihr Vorteil

Transparenz und Flexibilität

und Entwicklung

Ihr Vorteil

Qualität und Wartbarkeit durch testgetriebene Entwicklung und inkrementelles Design

www.it-agile.de

info@it-agile.de

Einleitung

Agile Softwareentwicklung ist in aller Munde. Fast jede größere IT-Konferenz bietet inzwischen einen eigenen agilen Track an, Zeitschriften bringen regelmäßig agile Themen, immer mehr Firmen steigen auf einen agilen Entwicklungsprozess um (oder behaupten dies zumindest) und auch Behörden wollen mit Agilität auf aktuelle Herausforderungen reagieren. Allerdings stellen wir immer wieder fest, dass viele Entwickler:innen, Projektverantwortliche und Führungspersonen immer noch verkürzte oder verzerrte Vorstellungen von Agilität haben: Pair Programming ist nicht identisch mit eXtreme Programming, und einfach das Lastenheft oder die Dokumentation wegzulassen und auf Zuruf zu entwickeln, macht keinen agilen Prozess aus. Entgegen weitverbreiteter Auffassung existieren auch und gerade in agilen Methoden klare Regeln und Abläufe – es sind zwar wenige, aber diese müssen dafür umso disziplinierter eingehalten werden, um erfolgreich zu sein.

Wir möchten mit unserer inzwischen über zwanzigjährigen Erfahrung dazu beitragen, etwas Licht ins Dunkel zu bringen, und in Form dieser Broschüre einen ersten Einstieg in die agile Softwareentwicklung geben. Dass in diesem Rahmen nicht auf Feinheiten und Detailfragen eingegangen werden kann, versteht sich von selbst.

Nach einer Beschreibung, was wirkungsvolle Agilität ausmacht, und einer Darstellung der allgemeinen, methodenübergreifenden Grundsätze agiler Softwareentwicklung werden die drei bekanntesten Methoden *Scrum, eXtreme Programming (XP)* und *Kanban* kurz vorgestellt. Im Anschluss folgt ein Vergleich der drei Methoden unter dem Aspekt, welche Methode unter welchen Voraussetzungen die geeignete ist. Es wird der Frage nachgegangen, wie erwachsen agile Methoden schon geworden sind und welche Vorteile das Management sowie auftraggebende und entwickelnde Personen aus ihnen ziehen können. Zum Abschluss zeigen wir die Grundprinzipien agiler Skalierung für große Projekte mit mehr als zehn Personen im Team.

Wir freuen uns, dass unsere Broschüre so großen Anklang gefunden hat. Deshalb haben wir uns entschlossen, eine 5. und erneut aktualisierte Auflage zu veröffentlichen. Und ganz im Sinne agiler Vorgehensweisen sind wir für jede Form von Feedback dankbar!

Henning Wolf, Stefan Roock und Stefan Zumbrägel
Hamburg & Vechta, im September 2022

PS: Unser Dank gilt Arne Roock für seinen großen Anteil an den ersten drei Auflagen dieser Broschüre.

Wirkungsvolle Agilität

Agilität scheint heute oft Selbstzweck zu sein. Das macht man halt so. Alle anderen machen das auch. Und oft kann man beobachten, dass Agilität in Unternehmen missverstanden wird als »Es geht vor allem um das Wohlbefinden der Teams!«. Führungspersonen berichten uns, dass aus ihrer Sicht oft ignoriert zu werden scheint, dass dabei auch Ergebnisse geliefert werden müssen, die das Unternehmen braucht.

Wie Agilität gemeint ist

Dabei bezieht das Agile Manifest[1] klar Stellung (siehe Abb. 1): Es geht darum, Kund:innen durch wertvolle Software zufriedenzustellen, Änderungen zum Wettbewerbsvorteil der auftraggebenden Personen zu nutzen, dass Entwickler:innen mit Fachleuten zusammenarbeiten und die Software (gelieferte Ergebnisse) das wichtigste Fortschrittsmaß ist.

Abb. 1 Auszüge Agiles Manifest

1. Siehe *https://agilemanifesto.org/iso/de/manifesto.html*.

Was dagegen nicht im Agilen Manifest steht, ist, dass das alles für das Team besonders angenehm vonstatten gehen sollte.

In diesem Sinne gilt:

Wirkungsvolle Agilität = Agilität, wie sie immer schon gemeint war

Ergebnisse und Wirkungen

Agile Teams stellen *Ergebnisse* her – in Scrum z. B. werden diese Ergebnisse Produktinkremente genannt. Letztlich sind diese Ergebnisse aber immer nur Mittel zum Zweck. Mit ihnen sollen *Wirkungen* erzielt werden; es soll sich etwas ändern (siehe Abb. 2).

Diese Wirkungen können sich auf die auftraggebenden Personen beziehen – für diese wird durch die Ergebnisse irgendetwas besser. Meist können sie mit dem Ergebnis etwas tun, was sie vorher nicht oder nicht so gut machen konnten. Mit MP3-Playern konnten Personen Musik unterwegs hören. Das konnten sie auch mit portablen CD-Playern, diese waren aber größer und schwerer als MP3-Player und hatten eine kürzere Akku-Laufzeit. Die Verwendung portabler CD-Player musste man planen. Mit MP3-Playern konnte man immer Tausende von Liedern dabeihaben.

Die Wirkungen können aber auch für das eigene Unternehmen eintreten: geringere Kosten, höhere Umsätze, neue Kundengruppen etc.

Abb. 2 Ergebnisse als Mittel zum Zweck – Wirkungen erzielen

In vielen Fällen wird man durch dasselbe Ergebnis Wirkungen für die auftraggebenden Personen *und* das eigene Unternehmen anstreben. Dadurch, dass durch das Ergebnis für die Auftraggebenden etwas besser wird, sind diese bereit, das auch finanziell wertzuschätzen. Es kann aber auch Ergebnisse geben, die nur dem eigenen Unternehmen nützen (z. B. wenn dadurch Geschäftsprozesse effizienter werden und Kosten eingespart werden können).

Diese Broschüre ist z. B. das Ergebnis der gemeinsamen Schreibarbeit der Autoren. Mit ihr wollen wir den Leser:innen einen schnellen Überblick über Agilität und die bekanntesten Ansätze ermöglichen. Wir hoffen aber auch, durch die dargestellte Expertise zu unserer Bekanntheit am Markt beizutragen.

Nach unserer Erfahrung liegt der Fokus viel zu häufig auf Ergebnissen, und die Wirkungen werden nicht thematisiert. Es wird die Velocity (Geschwindigkeit) des Teams genau beobachtet und eine Steigerung gefordert. Ob die Ergebnisse allerdings wirklich geeignet sind, um die beabsichtigten Wirkungen zu erzielen, wird gar nicht oder erst sehr spät diskutiert. Dabei sind nutzlose Ergebnisse, die in der halben Zeit erstellt wurden, immer noch nutzlose Ergebnisse.

Risiken und Nebenwirkungen

Jede Wirkung kann auch Nebenwirkungen haben, das kennen wir nicht nur aus der Pharmazie. Der neugenerierte Umsatz eines Möbelunternehmens mit konfigurierten Möbeln könnte sich ausschließlich auf Kosten des bisherigen Umsatzes ergeben, was möglicherweise nicht wünschenswert ist, wenn die Marge früher höher war. Der Umsatz könnte sogar sinken, weil den Kund:innen die Kaufentscheidung viel schwerer fällt.

Insofern wird unsere Beschreibung der erwünschten Wirkungen schnell komplex: neuen Umsatz generieren, ohne den alten Umsatz zu gefährden oder die bisherige Marge zu unterschreiten.

Diese Unwägbarkeiten schrecken mitunter so sehr ab, dass gar kein Versuch mehr unternommen wird, die Wirkung zu beschreiben. Das scheint uns keine besonders nützliche Lösung zu sein. Wie in allen anderen Bereichen agilen Arbeitens wollen wir in dieser Broschüre gemeinsam erfahren und lernen, auf welche Art im jeweiligen Kontext die Beschreibung der Wirkung nutzbringend gelingt.

Leistungsfähigkeit des Teams

Welche Ergebnisse das Team in welcher Zeit erstellen kann, hängt von seiner Leistungsfähigkeit ab (siehe Abb. 3). Die Leistungsfähigkeit des Teams ergibt sich aus den individuellen Fähigkeiten der handelnden Personen im Team, der Art ihrer Zusammenarbeit und der Einbettung des Teams ins Unternehmen.

Abb. 3 Leistungsfähigkeit des Teams

Die drei Ebenen Wirkungen, Ergebnisse, Leistungsfähigkeit des Teams lassen sich damit auf eine einfache Formel bringen: »Was wollen wir warum erreichen und was benötigen wir dazu?« (siehe Abb. 3). Die (Weiter-) Entwicklung der Leistungsfähigkeit des Teams wird an Ergebnisse und damit indirekt an Wirkungen gekoppelt: Welche Teamfähigkeiten müssen wir aufbauen, um die notwendigen Ergebnisse (besser) liefern zu können. Damit bekommt die Weiterentwicklung der Leistungsfähigkeit eine klare Richtung und erfolgt oft mit deutlich mehr Fokus und Energie. Es stellt sich nicht mehr die Frage, was wir verbessern *können*, sondern was wir verbessern *müssen*.

Es gibt bereits eine Menge bewährter Werkzeuge, mit denen eine wirkungsvollere Produktentwicklung gestaltet werden kann: wirkungsorientierte Roadmaps, Challenge Maps, Impact Mapping, Story Mapping, Business Stories, Dimensional Planning, True North/Nordstern, Toyota Kata, Agile Fluency Model™ etc. Sie betrachten jeweils sehr unterschiedliche Fragestellungen. Einfach alle einzusetzen, ohne den jeweiligen Kontext zu berücksichtigen, wird nicht zum Erfolg führen. Es geht auch hier darum, mit Blick auf die jeweils spezifischen Herausforderungen die passenden Hilfsmittel auszuwählen.

Was ist agile Softwareentwicklung?

Iteratives Vorgehen

Im Kern geht es bei agiler Softwareentwicklung um möglichst häufige Rückkopplung und zyklisches (iteratives) Vorgehen auf allen Ebenen: bei der Programmierung, im Team und beim Management.

Anders als in der klassischen Vorgehensweise wird das neu zu entwickelnde System nicht im Voraus in allen Einzelheiten genau geplant und dann in einem einzigen langen Durchgang entwickelt, denn schließlich können sich die Anforderungen während der Projektlaufzeit noch ändern, und oft sind sie zu Projektbeginn noch gar nicht vollständig bekannt.

Stattdessen wechseln sich beim agilen Vorgehen kurze Planungs- und Entwicklungsphasen miteinander ab. Nachdem eine Vision für das neue System entwickelt wurde, also die Ziele festgelegt und gewichtet wurden, die mit der Software erreicht werden sollen, wird ein Plan für eine erste Version ausgearbeitet, und die Entwicklung beginnt. Danach werden notwendige Anpassungen vorgenommen: Wie hoch ist die aktuelle Entwicklungsgeschwindigkeit? Stimmen die technischen und organisatorischen Rahmenbedingungen? Welche Probleme gilt es zu beseitigen? Hieraus und aus dem Feedback und ggf. Einsatz der bereits entwickelten Systemteile können sich wiederum Auswirkungen auf die Vision ergeben, die ggf. zu modifizieren ist, bevor der Plan für die nächste Iteration erstellt wird und ein neuer Zyklus beginnt.

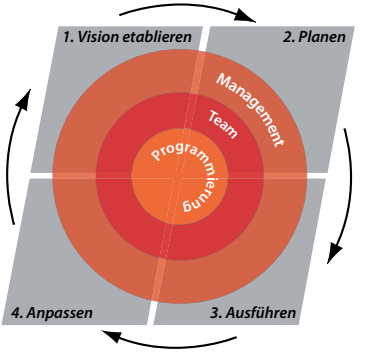


Abb. 4 Agiles Vorgehen beinhaltet Rückkopplungs- und Lernprozesse auf den Ebenen Programmierung, Team und Management.

Dabei wird in allen agilen Methoden großer Wert darauf gelegt, möglichst schnell eine Basisversion mit den wichtigsten Features zu entwickeln. Diese wird dann gemeinsam mit den Kund:innen/Produktverantwortlichen und Stakeholdern bewertet, um aus der bisherigen Entwicklungsarbeit zu lernen, erforderliche Anpassungen vorzunehmen und die kommenden Versionen zu optimieren. Diese Lern- und Verbesserungsprozesse sind natürlich nur dann möglich, wenn im Projekt häufig und offen kommuniziert wird. Daraus folgt insbesondere:

- Alle Projektbeteiligten dürfen Fehler machen und trauen sich auch, diese einzugestehen (auch gegenüber Personen in höheren Hierarchiestufen), ohne befürchten zu müssen, dass sie dafür bestraft werden. Jeder Fehler wird als Chance begriffen, zu lernen und in Zukunft noch bessere Leistung zu bringen.
- Es muss akzeptiert werden, dass alle Projektbeteiligten einzigartig sind und ihre ganz eigenen Stärken und Schwächen aufweisen, aber trotzdem (oder genau deshalb) jede und jeder einen wichtigen Beitrag für das Projekt leistet.
- Alle sind bereit, kritisiert zu werden und die anderen zu kritisieren. Dafür ist es wiederum notwendig, möglichst viel Transparenz zu schaffen und den anderen Projektbeteiligten Einblicke in die eigene Arbeit zu gewähren. In agilen Projekten sollte es deshalb keinerlei Geheimwissen geben.

Agile Methoden werden diesen hohen Anforderungen an Kommunikation gerecht, indem sie verschiedene Feedbacktechniken institutionalisieren: Bei *Review* und *Planung* legen Kund:innen und Entwickler:innen gemeinsam fest, wie die Anforderungen für den kommenden Entwicklungszyklus zu priorisieren sind; im *Standup-Meeting* (auch *Daily Scrum* genannt) reflektiert das Team kurz über die Arbeit des letzten Tages und plant den kommenden Arbeitstag; und *Retrospektiven* dienen dazu, über den Entwicklungsprozess zu reflektieren und diesen für die Zukunft zu verbessern. Für die auftraggebenden Personen und das Management bedeutet dies eine enge Zusammenarbeit mit den Entwickler:innen und damit verbunden ein vermeintlich höherer Zeitaufwand als in traditionellen Projekten. In Wahrheit spart diese enge Zusammenarbeit jedoch Zeit (und Geld) in ganz erheblichem Maße, weil Fehler generell mit umso weniger Aufwand zu beheben sind, je früher sie entdeckt werden.

Früher Systemeinsatz

Mit jeder neuen Software soll eine klare Wirkung für das Unternehmen und die Kund:innen erreicht werden, die in der Regel jeweils von diesen vorgegeben wird. Diese Wirkungen werden in notwendige Ergebnisse überführt, denen dann wiederum das neue System entsprechen soll. Es findet also ein doppelter Übersetzungsprozess statt: von den Wirkungen zu den Ergebnissen und von den Ergebnissen zum System. Diese beiden Übersetzungsprozesse stellen die größten Herausforderungen in der Softwareentwicklung dar, denn es ist weder trivial, für eine Wirkung die exakten Ergebnisse zu definieren, noch gegebene Ergebnisse in ein softwaretechnisches Artefakt zu überführen. Wie aber können die Ergebnisse dieser Übersetzungsprozesse überprüft werden und zu welchen Zeitpunkten geschieht dies?

Der Übersetzungsschritt von den Ergebnissen zum Softwaresystem lässt sich mittels Tests überprüfen. Agile Methoden sehen hierfür eine Reihe unterschiedlicher Testverfahren vor.

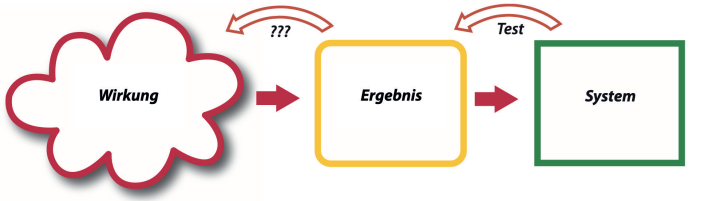


Abb. 5 Auf direktem Wege lässt sich nicht testen, ob definierte Anforderungen geeignet sind, die Geschäftsziele zu erreichen.

Für eine rein theoretische Überprüfung des Übersetzungsprozesses zwischen Wirkungen und Anforderungen ist uns hingegen kein systematischer Weg bekannt. Natürlich könnte eine entsprechende Überprüfung von Fachleuten vorgenommen werden. Wie viel Sicherheit diese letztlich erbringt, ist aber fraglich. Deshalb gehen agile Methoden einen indirekten Weg, der darin besteht, das neue System möglichst früh einzusetzen und so gegen die erwarteten Wirkungen zu prüfen. Es wäre äußerst ungünstig, wenn diese Überprüfung erst am Projektende erfolgen würde, weil es dann für Korrekturen und Umstellungen zu spät wäre. Stattdessen halten wir es für notwendig, dass diese Überprüfung so früh wie möglich erfolgt, damit bei Fehlentwicklungen noch während der Projektlaufzeit eingegriffen und gegengesteuert werden kann. Früher Systemeinsatz ist also die agile Antwort auf die Frage, wie sich überprüfen lässt, wie gut ein neues System seine Wirkungen erreicht.

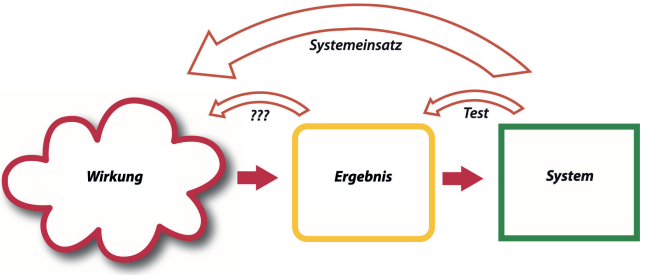


Abb. 6 Agile Methoden setzen auf frühen Systemeinsatz, um herauszufinden, ob die neue Software den Geschäftszielen gerecht wird.

Scrum

Das agile Managementframework Scrum[1] stellt heute den bekanntesten agilen Vertreter dar, weil es durch seine einfache Struktur und die klar definierten Verantwortlichkeiten einfach verständlich ist und sich schnell produktiv einsetzen lässt.

Autonomes, cross-funktionales Team

Im Mittelpunkt von Scrum steht das autonome, cross-funktionale Scrum-Team. Das Team entscheidet ohne Einfluss von außen, wer was wann und wie macht. Es ist selbstgemanagt, das bedeutet, das Team verfügt über alle notwendigen Fähigkeiten sowie Entscheidungsfreiheiten und kommt ohne Projektleitung oder weiteres Management aus. Um dem Team eine störungsfreie Arbeit zu ermöglichen und es beim Selbstmanagement und der Cross-Funktionalität zu unterstützen, gibt es den *Scrum Master*. Dem *Product Owner* (Produktverantwortlicher) kommt die Aufgabe zu, Anforderungen zu definieren, zu priorisieren und später ggf. auch auszutauschen. Allerdings ist in Scrum klar geregelt, wann er neue oder geänderte Anforderungen beauftragen darf: So gibt es ungestörte Entwicklungszyklen von wenigen Wochen (sogenannte *Sprints*), in denen Änderungen an den Anforderungen nicht zugelassen sind. Während der Sprints arbeiten die Entwickler:innen alle Anforderungen ab, die für diesen Sprint vorgesehen waren, und zwar innerhalb einer *Timebox* – es steht also nur ein genau festgelegter Zeitrahmen zur Verfügung, und kein Tag mehr. Wie lang die Sprints dauern, wird je nach Gegebenheiten zu Beginn des Projekts festgelegt, in der Regel sind es 1–4 Wochen. Die Entwickler:innen sind in einem regelmäßigen Austausch mit dem Product Owner, sodass dieser stets auf dem aktuellen Stand ist. Am Ende jedes Sprints werden die neuen Funktionalitäten allen relevanten Stakeholdern vorgestellt, um mit ihnen über notwendige Anpassungen des bisherigen Entwicklungsplans zu diskutieren. Dem Scrum Master kommt während der Sprints die Aufgabe zu, externe Störungen vom Scrum-Team fernzuhalten. Er begleitet die Lösung aufkommender Probleme durch Impulse, Moderation und vereinzelt, indem sie von ihm selbst gelöst werden.

1. Siehe *https://to.it-agile.eu/scrumguide*.

Umgang mit Anforderungen und Sprint Planning

Die gesamten Anforderungen an das neue System werden im *Product Backlog* festgehalten, das immer auf ein einzelnes Produktziel ausgerichtet ist. Der Product Owner kann jederzeit Ideen für neue Anforderungen in das Product Backlog eintragen. Welche Anforderungen dann jedoch tatsächlich im nächsten Sprint umgesetzt werden sollen, wird im *Sprint Planning Meeting* besprochen. Der Product Owner legt die Reihenfolge fest, in der die Anforderungen umgesetzt werden sollen, die Entwickler:innen bestimmen, wie viele dieser Anforderungen im Sprint möglich sind. Im *Sprint Backlog* stehen dann genau die Anforderungen für den nächsten Sprint sowie ein konkreter Umsetzungsplan. Änderungen am Sprint Backlog von außerhalb des Scrum-Teams während des Sprints sind nicht erlaubt. Im Ergebnis entsteht am Ende des Sprints das *Inkrement*, ein konkreter Schritt in Richtung *Produktziel*. Das Scrum-Team einigt sich darauf, welche Maßnahmen erfüllt sein müssen, damit ein Inkrement von sehr guter Qualität erzeugt wird. Diese Maßnahmen bilden gemeinsam die *Definition of Done*.

Review und Retrospektive

Am Ende jedes Sprints finden zwei Meetings statt:

- Das Sprint-Review, um Feedback und Verbesserungsideen für das entstehende Produkt zu finden.
- Die Sprint-Retrospektive, um über den verwendeten Prozess zu reflektieren und diesen kontinuierlich zu verbessern.

Im Sprint-Review demonstrieren die Entwickler:innen gemeinsam mit dem Product Owner das Produkt möglichst vielen am Produkt interessierten Personen. Ziel des Meetings ist es, möglichst viel Feedback zu bekommen, ob man das richtige Produkt baut und damit effektiv unterwegs ist. Der Product Owner entscheidet, welcher Teil des Feedbacks im Anschluss zu neuen oder veränderten Einträgen im Product Backlog führt.

In der Sprint-Retrospektive reflektiert das Scrum-Team darüber, was gut und was weniger gut gelaufen ist im letzten Sprint, um gemeinsam konkrete Verbesserungsmaßnahmen zu finden. Diese Verbesserungsmaßnahmen werden dann im Optimalfall direkt im nächsten Sprint umgesetzt und verbessern kontinuierlich das Vorgehen nach Scrum.

Daily Scrum

Neben dem Sprint-Zyklus gibt es bei Scrum noch eine zweite, kleinere Feedbackschleife: Täglich zur gleichen Zeit treffen sich die Entwickler:innen zum Daily Scrum. Dieses Treffen dauert nur etwa 15 Minuten und wird direkt am Ort des Sprint Backlog durchgeführt. Die Entwickler:innen synchronisieren sich zu den folgenden Themen:

1. Was ist seit dem letzten Daily Scrum für das Sprint-Ziel erledigt worden?
2. Auf welche Hindernisse sind wir in der Zeit gestoßen?
3. Was wollen wir bis zum nächsten Daily Scrum tun, um dem Sprint-Ziel näher zu kommen, und wer macht es?

So bleiben alle im Team stets auf dem Laufenden, zudem werden Probleme frühzeitig erkannt und können (ggf. mithilfe des Scrum Masters) behoben werden.

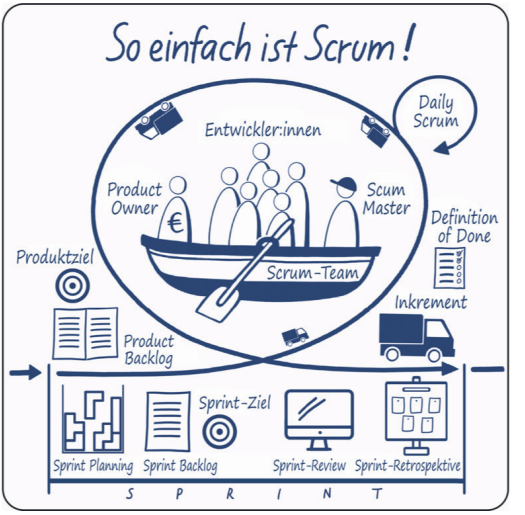


Abb. 7 Die wesentlichen Elemente von Scrum lassen sich auf einem Bierdeckel darstellen.

Der Scrum Master und der Product Owner sind bei diesem Termin optional. Der Scrum Master ist aber oft dabei, um den Entwickler:innen dabei zu helfen, die 15-Minuten-Timebox einzuhalten und Hindernisse direkt zu erkennen.

Es ist verhältnismäßig einfach, sich Scrum anzueignen, und schnell einzusetzen. Somit kann es häufig den ersten Schritt darstellen, um Entwicklungsprojekte agil zu machen. Darüber hinaus definiert Scrum klare Verantwortlichkeiten (Entwickler:innen, Product Owner und Scrum Master) und schafft somit Transparenz und eindeutige Zuständigkeiten im Team. Die Entwickler:innen können sich tatsächlich auf die Entwicklung konzentrieren und werden dabei kaum gestört, weil der Scrum Master alle Probleme, die die Organisation und Infrastruktur betreffen, aus dem Weg räumt. Allerdings müssen die wenigen Vorgaben, die Scrum macht, mit großer Disziplin eingehalten werden, weil sonst aus dem agilen Entwicklungsprozess schnell Chaos entstehen kann. Außerdem stellt die Idee von selbstgemanagten Teams eine große Herausforderung für viele Entwickler:innen und Organisationen dar und ist in der Praxis nicht trivial umzusetzen. Schließlich muss man sich im Klaren darüber sein, dass Scrum einen reinen Managementrahmen darstellt und keinerlei Vorgaben für die Programmierung macht – hier ist also eine Kombination mit anderen Methoden wie z. B. eXtreme Programming angebracht.

Scrum auf einen Blick:

- Verhältnismäßig leicht zu lernen
- Lässt sich schnell einsetzen
- Definiert klare Verantwortlichkeiten (Entwickler:innen, Product Owner und Scrum Master) und einen gut strukturierten, aber dennoch flexiblen Entwicklungsprozess
- Setzt auf ein selbstgemanagtes Team
- Macht als reine Managementmethode keine Vorgaben bzgl. der Programmierung

eXtreme Programming

eXtreme Programming (XP) definiert neben einem Scrum-ähnlichen Managementrahmen agile Entwicklungspraktiken. Diese Entwicklungspraktiken komplett umzusetzen, ist anspruchsvoll und ihre effektive Verwendung benötigt monate- bis jahrelanges Einüben.

Ineinander verzahnte Rückkopplungszyklen

Die »XP-Maschinerie« wird durch Rückkopplungsmechanismen von ganz unterschiedlicher Dauer angetrieben:

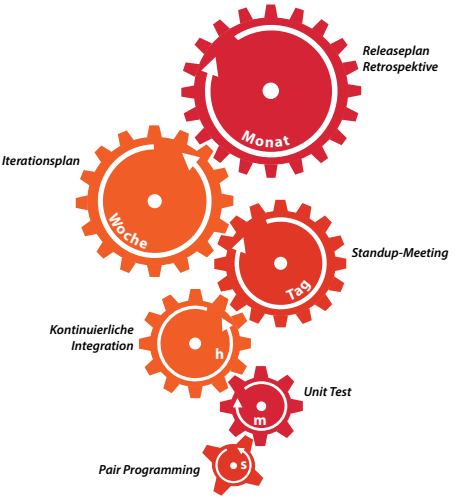


Abb. 8 Rückkopplungsmechanismen in XP auf unterschiedlichen Ebenen

***Sekundentakt*:** Das *Pair Programming* (zwei Personen programmieren gemeinsam an einem Rechner) führt dazu, dass sich die Entwickler:innen kontinuierlich austauschen und gegenseitig inspirieren. Programmierfehler, umständlicher Entwurf etc. werden mit hoher Wahrscheinlichkeit sofort entdeckt und beseitigt.

***Minutentakt*:** In XP wird testgetrieben entwickelt, die *Unit Test*s werden also *vor* dem Code geschrieben. Mit diesem Ansatz testgetriebener

Entwicklung (engl. *Test Driven Development, TDD*) wird nicht nur eine sehr hohe Testabdeckung des Codes erreicht, sondern es wird auch der softwaretechnische Entwurf inkrementell getrieben.

Stundentakt: Neu entwickelte Komponenten werden mehrmals täglich in das lauffähige Gesamtsystem integriert (*kontinuierliche Integration,* engl. *Continuous Integration, CI*). So lassen sich Fehler schneller finden – denn je später Fehler entdeckt werden, umso aufwendiger sind sie zu beheben.

Tagestakt: In XP findet ein tägliches Treffen, das *Standup-Meeting*, statt, bei dem das Entwicklungsteam kurz über den Projektfortschritt reflektiert und sich für den anstehenden Tag koordiniert.

Wochentakt: Durch kurze Releasezyklen erhalten Kund:innen immer wieder lauffähige Systemversionen, um diese zu testen und ihre fachlichen Anforderungen auf den neuesten Stand zu bringen. Deshalb werden in regelmäßigen Abständen neue *Iterationspläne* erstellt.

Monatstakt: Regelmäßig – am besten monatlich – werden neue Releases produktiv gestellt, um möglichst früh Geschäftswert zu generieren. In Abständen von wenigen Monaten werden *Releasepläne* erstellt, in denen die nächsten Iterationen und Releases festgehalten werden. Mindestens einmal im Monat finden Retrospektiven statt, in denen über den Entwicklungsprozess reflektiert und Verbesserungsmaßnahmen beschlossen werden.

Die fünf Werte des eXtreme Programming

XP nennt als Grundlage die folgenden fünf Werte: *Kommunikation*, *Rückkopplung*, *Einfachheit*, *Mut* und *Respekt*. Kommunikation ist sowohl für die Teamarbeit als auch die Zusammenarbeit mit Kund:innen sehr wichtig. Rückkopplung ist ein wesentlicher Schlüssel zur ständigen Verbesserung. Einfachheit wird angestrebt, um pragmatische, schnelle und unkomplizierte Lösungen zu finden – sowohl technisch als auch organisatorisch. Mut und Respekt schließlich sind wichtige Werte für den wechselseitigen Umgang miteinander: Mut erlaubt es, vom üblichen Weg abzuweichen, wechselseitiger Respekt gewährleistet dabei, dass dies für den Einzelnen keine negativen Folgen hat.

Die grundlegenden Werte stellen in Projekten eine große Entscheidungshilfe dar, weil bei jeder Entscheidung danach gefragt werden kann, ob sie mit dem Wertesystem konform ist. Darüber hinaus können sich XP-Teams zusätzliche projektspezifische Werte definieren.

Die 14 Prinzipien des eXtreme Programming

XP kennt 14 Prinzipien: Menschlichkeit, Wirtschaftlichkeit, gegenseitiger Vorteil, Selbstähnlichkeit, Verbesserungen, Mannigfaltigkeit, Reflexion, Fluss, Gelegenheiten wahrnehmen, Redundanz vermeiden, Fehlschläge hinnehmen, Qualität, kleine Schritte und akzeptierte Verantwortlichkeit.

Wir gehen im Folgenden nur auf sechs ausgewählte Prinzipien näher ein.

Wirtschaftlichkeit: XP adressiert explizit das Thema Wirtschaftlichkeit. Softwareprojekte ergeben nur Sinn, wenn die Software einen höheren Nutzen bringt, als ihre Entwicklung an Kosten verursacht.

Mannigfaltigkeit: Alle Projektbeteiligten sind mit ihrer jeweiligen Ausbildung, Meinung und Erfahrung willkommen. Alle Meinungen sind gefragt und wichtig.

Reflexion: Auf allen Ebenen im Entwicklungsprozess setzt XP auf Reflexion, weil hierdurch Lernprozesse und Verbesserungen ermöglicht werden.

Qualität: Qualität zum Prinzip zu erheben ist wichtig, um sich in Zweifelsfällen für Qualität und gegen andere Faktoren (Termine!) entscheiden zu können.

Kleine Schritte: Vorgehen in großen Schritten birgt immer die Gefahr, dass im Fall eines Fehlschlags dieser auch eine entsprechende Größe annimmt. Deswegen wird in XP-Projekten risikominimierend in möglichst kleinen Schritten vorgegangen.

Akzeptierte Verantwortlichkeit: XP setzt darauf, dass Verantwortlichkeit nur wahrgenommen werden kann von Leuten, die diese Verantwortung auch akzeptiert haben. Verantwortung kann nicht zugewiesen werden.

Die 13 Primärpraktiken

XP definiert 13 Primärpraktiken, die angewendet werden müssen, damit man von einem XP-Projekt sprechen kann.

***Räumlich zusammensitzen*:** Alle Projektbeteiligten sollen räumlich möglichst nah beieinandersitzen, idealerweise in einem Raum. Die Nähe erleichtert die direkte Kommunikation und reduziert Reibungsverluste indirekter Kommunikation.

***Komplettes Team*:** Das XP-Team soll alle Qualifikationen enthalten, die für das Projekt erforderlich sind. Damit sind sowohl technische wie fachliche Qualifikationen gemeint. So kann das Team schnell voranschreiten, ohne auf die Verfügbarkeit externer Ressourcen warten zu müssen.

***Informative Arbeitsumgebung*:** Die Arbeitsumgebung des Teams soll den aktuellen Projektzustand widerspiegeln. Das betrifft die noch offenen Aufgaben, den Zustand des Systems bzgl. Tests, den Kernentwurf des Systems etc. Den größten Teil dieser Praktik kann man realisieren, indem man geeignete Flipchart-Blätter und Ausdrucke an den Wänden des Teamraums aufhängt.

***Energiegeladene Arbeit*:** Die Arbeit in einem XP-Projekt ist energiegeladen, d. h., alle Teammitglieder sind engagiert bei der Sache und bringen vollen Einsatz.

***Pair Programming*:** Zwei Personen aus dem Entwicklungsteam sitzen vor einem Rechner mit einer Tastatur und programmieren gemeinsam, wobei abwechselnd mal die eine, mal die andere Person die Tastatur bedient.

***Geschichten*:** Die Anforderungen werden in XP-Projekten als informelle Geschichten (engl. *User Stories*) aufgeschrieben. Wer genau diese Geschichten schreibt, Kund:innen oder Entwickler:innen, lässt XP heute offen. Zunächst ist nur wichtig, dass Anforderungsgeschichten existieren.

***Wöchentlicher Zyklus*:** Eine Iteration dauert eine Woche, denn eine Woche ist eine natürliche Zeiteinheit für Teams. Sie ist überschaubar und klein genug für eine detaillierte Planung. Dabei handelt es sich um einen Erfahrungswert.

***Quartalsweiser Zyklus*:** Ein Release dauert ein Quartal. Auch hier handelt es sich um einen Erfahrungswert. Ein großer Vorteil solcher vom Kalender vorgegebenen Zeiträume besteht darin, dass man sie nicht künstlich verlängern kann. Ein Quartal endet zu einem festgelegten Zeitpunkt und nicht zehn oder 20 Tage später (mit mehr Funktionalität im System).

***Freiraum*:** Entwickler:innen brauchen zwischendurch Freiraum, um sich mit den Neuerungen außerhalb des Projekts vertraut zu machen. Ansonsten werden sie schnell vom technologischen Geschehen abgehängt und sind dann im nächsten Projekt weniger gut einsetzbar. Es kann auch vorkommen, dass die Teammitglieder während ihrer Freiraumphasen auf Technologien oder Vorgehensweisen stoßen, von denen bereits ihr aktuelles Projekt profitieren kann.

***Zehn-Minuten-Build*:** Es darf maximal zehn Minuten dauern, das Projekt zu übersetzen und die Tests auszuführen. Diese Forderung mag in großen Projekten unerfüllbar erscheinen, mit einer geschickten Aufteilung in Teilprojekte kann man ihr aber sehr nahekommen.

***Kontinuierliche Integration*:** Entwickler:innen integrieren ihre Änderungen mehrfach am Tag in die gemeinsame Codebasis.

***Testgetriebene Entwicklung*:** Bei der testgetriebenen Entwicklung wird der Testcode vor dem Produktivcode geschrieben. Die Tests werden nach jedem Programmierschritt ausgeführt und liefern Rückmeldung über den Entwicklungsstand der Software.

***Inkrementeller Entwurf*:** Der Softwareentwurf soll inkrementell erfolgen, also schrittweise entlang der Anforderungen. Dabei sollen immer nur die nächsten, konkret bekannten Anforderungen berücksichtigt werden.

Die 11 Folgepraktiken

Erst wenn Teams die Primärpraktiken beherrschen und erfolgreich anwenden können, sollten auch die 11 Folgepraktiken zum Einsatz kommen, die XP kennt: echte Kundenbeteiligung, inkrementelle Auslieferung, Teamkontinuität, schrumpfende Teams, Ursprungsursachen-Analyse, gemeinsamer Code, Codierung und Testen, eine zentrale Codebasis, tägliche Auslieferung, verhandelbarer, vertraglicher Funktionsumfang, Bezahlung pro Nutzung.

XP auf einen Blick:

- Legt großen Wert auf Qualitätssicherung durch testgetriebene Entwicklung und Pair Programming
- Macht klare Vorgaben für Management, Team *und* Programmierung
- Äußerst mächtige, aber anspruchsvolle Methode
- Scrum-Teams verwenden häufig XP-Entwicklungspraktiken

Kanban

Bei Kanban handelt es sich weder um eine Entwicklungsmethode (wie bei XP) noch um ein Managementframework (wie bei Scrum). Vielmehr stellt Kanban eine Methode zum Change Management dar. Das Ziel besteht darin, den vorhandenen Prozess – wie auch immer der aussehen mag – schrittweise zu verbessern. Kanban wird also auf einen bestehenden Entwicklungsprozess aufgesetzt und lässt sich deshalb auch problemlos sowohl mit anderen agilen Ansätzen als auch mit herkömmlichen Methoden wie dem Wasserfallmodell kombinieren.

David Anderson[2], der Vater von Kanban in der Softwareentwicklung, hat sechs Grundprinzipien und sechs Kerneigenschaften von Kanban definiert.

Grundprinzipien

Die sechs Grundprinzipien unterteilen sich in Veränderungs- und Serviceprinzipien.

Die drei Veränderungsprinzipien:

1. Beginne mit dem, was du gerade tust:
   - verstehe aktuelle Prozesse, wie sie tatsächlich praktiziert werden, und
   - respektiere vorhandene Rollen, Verantwortlichkeiten und Job-Titel
2. Vereinbare, dass evolutionäre Veränderungen verfolgt wird
3. Fördere Leadership auf allen Ebenen der Organisation

fokussieren darauf, wie in dem vorhandenen Prozess Verbesserungen erzeugt werden können.

2. Vgl. Anderson, David J.: Kanban – *Evolutionäres Change Management für IT-Organisationen.* dpunkt.verlag, 2011.

Die drei Serviceprinzipien:

1. Verstehe die Bedürfnisse und Erwartungen deiner Kunden und fokussiere auf diese
2. Manage den Service, nicht die Menschen
3. Entwickle Strukturen, die den Service für deine Kunden und dich kontinuierlich verbessern

stellen die Qualität des Angebots in den Mittelpunkt.

In diesen Grundprinzipien drückt sich ein Hauptziel aus, das ausschlaggebend für die Entstehung von Kanban war: David Anderson wollte einen Weg finden, wie sich agile Entwicklungsmethoden und andere organisatorische Verbesserungen ohne größere Widerstände der Beteiligten im Unternehmen einführen lassen. Im Gegensatz zu vielen anderen Veränderungsinitiativen wird in Kanban kleinschrittig vorgegangen, während disruptive Veränderungen nicht vorgesehen sind. Vielmehr werden immer wieder kleine Anpassungen am bestehenden Prozess vorgenommen und dann beobachtet, welche Auswirkungen sich hieraus ergeben.

Kernpraktiken von Kanban

Kanban-Systeme weisen sechs Kernpraktiken auf:

1. Visualisiere (die Arbeit, den Fluss der Arbeit/Workflow und Geschäftsrisiken)

Visualisierung spielt eine große Rolle in Kanban und sie stellt stets den ersten Schritt bei der Kanban-Einführung dar. Dabei sollte die Visualisierung möglichst einfach und übersichtlich sein und sich flexibel anpassen lassen. Zu diesem Zweck haben sich Whiteboards als gutes Mittel herausgestellt. Die einzelnen Prozessschritte werden als Spalten dargestellt und die Anforderungen auf Karteikarten oder Haftnotizen notiert. Diese Tickets durchlaufen dann das Kanban-Board von links nach rechts. Gute Kanban-Boards machen sehr schnell deutlich, in welchem Zustand sich die einzelnen Tickets befinden, wie sich die Arbeit verteilt und wo Probleme bestehen.

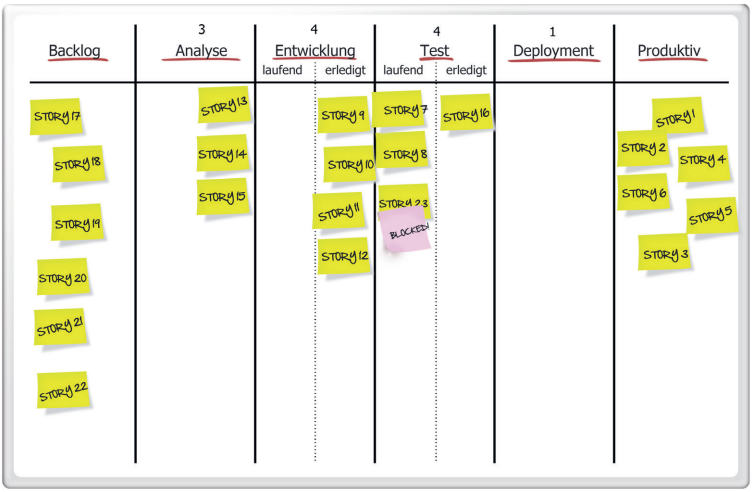


Abb. 9 Beispiel eines Kanban-Boards mit verschiedenen Visualisierungen

2. Limitiere die parallele Arbeit (Work in Progress)

In Kanban werden Limits für die einzelnen Prozessschritte definiert. Wenn beispielsweise das Limit für die Entwicklung vier beträgt, dürfen sich maximal vier Tickets gleichzeitig in dieser Spalte befinden. Diese Limitierung verhindert die schädlichen Auswirkungen von Multitasking (Zeitverschwendung durch Kontextwechsel, Flüchtigkeitsfehler), erhöht die Qualität und verkürzt die Durchlaufzeit für jedes einzelne Ticket. Darüber hinaus werden durch WIP-Limits Engpässe und Probleme schneller sichtbar, sodass das Team Verbesserungsmaßnahmen ergreifen kann.

3. Manage den Arbeitsfluss (Flow)

Kanban legt großes Gewicht auf quantitatives Management. Das bedeutet, dass konsequent Metriken erfasst und ausgewertet werden. Diese Metriken bilden dann den Ausgangspunkt für kleine Anpassungen am Kanban-System. Die wichtigste Metrik in Kanban stellt die Durchlaufzeit (engl. *Lead Time*) dar, also die Zeit, die ein Ticket benötigt, um das gesamte System zu durchlaufen. Diese Zeit soll kontinuierlich verkürzt werden, um dem Kunden möglichst häufig Wert zu liefern. Interessanterweise hat sich gezeigt, dass sich auch die Qualität erhöht und die Teamarbeit verbessert, wenn man die Durchlaufzeit kontinuierlich verkürzt.

Weitere Metriken, die häufig von Kanban-Teams verwendet werden, sind der Durchsatz, die Fehlerrate und die Termintreue.

4. Mache Prozessregeln explizit

In Kanban wird angestrebt, ein System aufzubauen, in dem sich die Teammitglieder selbstständig neue Tickets für die Bearbeitung ziehen können (Pull-System). Darüber hinaus sollen sie Probleme leicht erkennen, diskutieren und beheben können. Hierfür ist es nötig, die impliziten Prozessregeln (die es in jedem Entwicklungsprozess gibt) explizit zu machen und möglichst gut sichtbar zu haben (z. B. auf einem Flipchart neben dem Kanban-Board). Beispiele für solche Regeln sind: Wie kommen neue Tickets in das Kanban-System hinein? In welchem Takt? Wer priorisiert diese? In welcher Reihenfolge werden die Tickets bearbeitet? Unter welchen Umständen dürfen die WIP-Limits gebrochen werden? Wie geht das Team mit Bugs um? Welche Kriterien müssen erfüllt werden, damit der Status eines Tickets auf »fertig« gesetzt werden darf?

5. Implementiere Feedbackzyklen

Ein evolutionärer Prozess kann ohne Feedbackzyklen nicht funktionieren. Wenn man diese Zyklen auf der Ebene der angebotenen Dienstleistungen implementiert, erfordert Kanban die folgenden vier Praktiken: das tägliche Standup-Meeting des Teams, das regelmäßige Review der gelieferten Dienstleistung, das Operations Review (ähnlich einer Retrospektive, aber eine eher datenbasierte, empirische Auswertung) und systematisches Risikomanagement.

Das Ziel der Feedbackzyklen ist es, erwartete Ergebnisse oder Bedürfnisse mit tatsächlichen Ergebnissen abzugleichen und daraus Verbesserungsmaßnahmen abzuleiten.

6. Verbessere gemeinsam, entwickle experimentell weiter (und verwende dazu Modelle und wissenschaftliche Methoden)

Kaizen, also die kontinuierliche Verbesserung in kleinen Schritten, ist untrennbar mit Kanban verbunden. Erfahrungsgemäß lassen sich Verbesserungen am besten gemeinsam mit allen am Prozess Beteiligten erarbeiten. Zudem können kleine Experimente gut dazu verwendet werden, die Verbesserungen mit kleinerem Risiko zu betreiben. Um Chancen für solche Verbesserungen zu erkennen und zu diskutieren, hat es sich als sinnvoll erwiesen, verschiedene Modelle gemeinsam mit Kanban zu verwenden. Die Engpasstheorie, Systems Thinking, die Ideen Edwards Demings, das Toyota Production System und die Real-Options-Theorie stellen solche Modelle dar.

Kanban auf einen Blick:

- Agile Change-Management-Methode
- Änderungen werden in kleinen Schritten durchgeführt
- Dadurch höhere Akzeptanz bei allen Beteiligten
- Lässt sich mit anderen agilen und klassischen Methoden kombinieren
- Kontinuierliche Verbesserung als wichtiges Prinzip

Welche Methode passt für wen?

Die hier kurz vorgestellten agilen Methoden haben verschiedene Schwerpunkte und sind für unterschiedliche Projektkonstellationen, Teams und Organisationen unterschiedlich gut geeignet. Die wenigsten agilen Teams folgen dauerhaft genau einer dieser Methoden, sondern integrieren Praktiken aus unterschiedlichen agilen Ansätzen.

Trotzdem stellt sich zum Einstieg berechtigterweise die Frage, welche agile Methode als guter Startpunkt in die agile Softwareentwicklung dienen kann. Wir betrachten im Folgenden – abhängig von Zielen und Voraussetzungen –, für wen welche agile Methode ein geeigneter Startpunkt wäre. Bedenken Sie dabei aber immer, dass Sie damit nur eine grobe Richtung erhalten. Sie müssen für die individuelle Situation jeweils im Einzelfall überprüfen, was für Sie passt.

Faktisch reduziert sich die Frage nach dem geeigneten Ansatz für den Einstieg in die agile Entwicklung heute darauf, ob Sie mit Scrum oder Kanban starten. eXtreme Programming komplett auf einen Schlag einzuführen, überfordert nach unserer Erfahrung die Beteiligten. Wenn Scrum oder Kanban etabliert wurden, werden in der Regel XP-Praktiken integriert. Für die Entscheidung zwischen Scrum und Kanban spielen die aktuelle Situation und die Zielsetzung eine entscheidende Rolle.

Läuft die Entwicklung eigentlich ganz gut und suchen Sie nach einer Möglichkeit, noch besser zu werden, werden Sie die mit Scrum verbundenen gravierenden Veränderungen vielleicht nicht in Kauf nehmen wollen. Sie fahren dann vermutlich mit Kanban zur kontinuierlichen Verbesserung des existierenden Vorgehens besser. Gegebenenfalls ist es auch sinnvoll, XP-Praktiken in Ihr jetziges Vorgehen zu integrieren.

Haben Sie heute große Probleme mit der aktuellen Entwicklung und/oder nur wenig Zeit, um erhebliche Verbesserungen zu erzielen, führt Kanban vielleicht nicht ausreichend schnell zu den gewünschten Ergebnissen. Dann kann es sinnvoller sein, Scrum einzuführen.

Ein weiterer Anwendungsfall für Scrum ist die Abwesenheit funktionierender Entwicklungsprozesse. Dann gibt es nichts, worauf Kanban zur Verbesserung angewendet werden kann, und es ist sinnvoll, mit Scrum zunächst einmal einen Entwicklungsprozess zu etablieren. Auf diesen kann man dann ggf. später Kanban zur weiteren Verbesserung anwenden.

Nicht zuletzt spielt die Art der Arbeit eine wichtige Rolle. Arbeitet das Team Anfragen ab, die miteinander nicht im Zusammenhang stehen und die einzeln geliefert werden können, stellen die Planung in Scrum und das Sprint-Konzept einen unnötigen Overhead dar. Das gilt noch verstärkt, wenn häufig schnell reagiert werden muss. Kanban ist dann die leichtgewichtigere Alternative. Beispiele für solche Anwendungsfälle umfassen Wartung und Betrieb.

Geht es hingegen um nennenswert innovative Produktentwicklung hat Scrum mit cross-funktionalen Teams, nach Wert priorisierten Product Backlogs und der Product-Owner-Verantwortlichkeit nützliche Mechanismen im Gepäck. Will man Kanban für innovative Produktneuentwicklung anwenden, müssen vermutlich zumindest einige der Scrum-Praktiken integriert werden.

Einführungsstrategien

Auch wenn agile Methoden einfach zu beschreiben sind und nicht ohne Grund als leichtgewichtig bezeichnet werden (wegen des geringeren Overheads), ist es je nach Ausgangslage gar nicht so einfach, einen guten Einstieg in agile Softwareentwicklung zu finden. Dabei stehen einem nicht nur alte, schwer abzulegende Gewohnheiten im Wege. Viele der agilen Praktiken sind wenig formal und erfordern gerade deshalb viel Disziplin bei allen Projektbeteiligten.

Rückkopplung und Lernen sind agile Tugenden, die man sich auch bei der Einführung agiler Methoden zunutze machen kann. Es gilt also, das Lernen anzustoßen und systematisch Rückkopplung begleitend zu einer Einführung zu geben bzw. einzuholen. So kann schrittweise gelernt und der Einsatz agiler Methoden oder Praktiken inkrementell angepasst werden.

Idealtypisch kann eine Einführung wie in Abbildung 10 dargestellt aussehen.

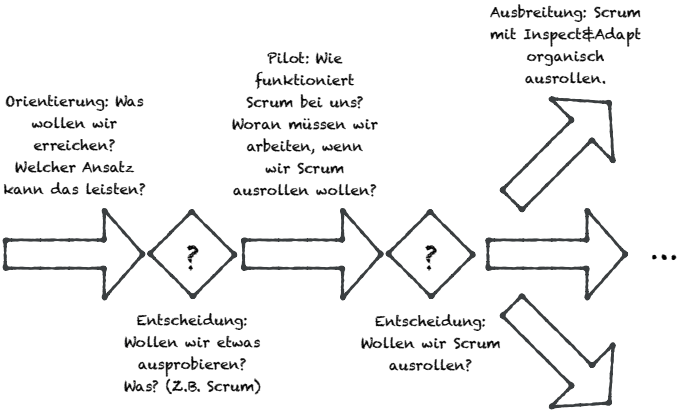


Abb. 10 Einführungsstrategie

Wir beginnen mit einer Orientierungsphase, in der wir zunächst klären, was wir überhaupt erreichen wollen. Mit agilen Ansätzen kann man vielfältige Vorteile für das Unternehmen erreichen:

- Kürzere Time-to-Market
- Angemessenere Software (durch schnellere Reaktion auf Änderungswünsche)
- Höhere Qualität (weniger Bugs)
- Höhere Produktivität
- Mehr Innovation

Auch wenn man all diese Vorteile bekommen kann, muss man fokussieren. Man kann leider nicht alles auf einmal bekommen.

Wenn geklärt ist, welcher Vorteil zuerst im Fokus steht, kann man sich mit verschiedenen agilen Ansätzen beschäftigen und abschätzen, welcher für das Ziel am besten geeignet erscheint.

Diese Orientierungsphase sollte nicht zu lange dauern und nicht den Anspruch auf Korrektheit oder Vollständigkeit haben. Sie werden erst dann wirklich verstehen, was der jeweilige Ansatz bedeutet, wenn Sie ihn praktizieren.

Daher sollten Sie möglichst schnell zu einer Entscheidung kommen, ob und welchen agilen Ansatz Sie ausprobieren möchten (siehe voriger Abschnitt). Dann setzen Sie ein Pilotprojekt auf, um Erfahrung mit dem Ansatz zu sammeln und zu lernen, was das für Sie im Unternehmen bedeutet. Wir empfehlen, im Pilotprojekt die gewählte Methode »by the book« (nach Lehrbuch) einzusetzen. So erkennen Sie am schnellsten, welche Vorteile Sie erreichen können, aber auch welche Herausforderungen mit einer späteren Verankerung im Unternehmen verbunden wären.

Auf dieser Basis werten Sie das Pilotprojekt aus (i.d.R. nach 3 bis 6 Monaten) und entscheiden, ob die Vorteile überwiegen und Sie die Vorteile dauerhaft im Unternehmen nutzen wollen. Sie legen dann auch fest, in welchem Umfang der Ansatz eingeführt werden soll. Im Pilotprojekt haben Sie gesehen, was es bedeutet, die Methode »by the book« einzusetzen, und können jetzt eine fundierte Entscheidung treffen, ob Sie die weitere Ausbreitung auf die gleiche Weise betreiben oder Anpassungen vornehmen.

Ausbreitung: PRN, Software-Studio, Enterprise Agility

Beim Ausbreitungsumfang unterscheiden wir in Anlehnung an Ken Schwaber und Jeff Sutherland3 zwischen *PRN*, *Software-Studio* und *Enterprise Agility*.

PRN ist die Abkürzung für das lateinische »pro re nata«, das bei Medikamentenverordnungen dafür steht, dass das Medikament bei Bedarf anzuwenden ist (und nicht regelmäßig). Ein Asthma-Akutspray ist ein Beispiel. Übertragen auf die Softwareentwicklung bedeutet *PRN*, dass das Unternehmen weiterhin wie bisher operiert und der gewählte Ansatz, z. B. Scrum, nur verwendet wird, wenn bestimmte Indikatoren vorliegen (z. B. sehr unklare Anforderungen oder hoher Zeitdruck). Der Nachteil dieses Ansatzes ist, dass kaum Wissensaustausch zwischen den agilen Projekten stattfindet; es ist unwahrscheinlich, dass die Mitglieder aus dem agilen Projekt A auch in einem späteren agilen Projekt B arbeiten werden.

Das *Software-Studio* begegnet diesem Problem: Es ist eine dauerhafte Einrichtung im Unternehmen, in dem Software agil entwickelt werden kann. Das Software-Studio bewahrt die Erfahrungen aus den agilen Projekten und entwickelt seine agilen Fähigkeiten immer weiter. Am Anfang besteht das Software-Studio vielleicht nur aus einem Agile Coach, an den man sich wenden kann, wenn man ein agiles Projekt aufsetzen will. Je erfolgreicher das Software-Studio ist, desto mehr Anfragen wird es geben und das Software-Studio wächst organisch. Am Ende kann es über eigene Räume und feste Entwicklungsteams verfügen. Der grundsätzliche Gedanke bleibt aber immer derselbe: Agile Entwicklung ist *ein* Angebot im Unternehmen und das Software-Studio fungiert als interner Dienstleister für agile Projekte.

Enterprise Agility geht noch einen Schritt weiter. Jegliche Softwareentwicklung wird agil durchgeführt und das agile Mindset wird immer weiter im Unternehmen verbreitet – auch deutlich über die Entwicklung hinaus. Größere Reorganisationen von funktionalen Silos hin zu einer produktorientierten oder marktorientierten Organisationsform können die Folge sein.

3. Schwaber, Ken; Sutherland, Jeff: *Software in 30 Tagen*. dpunkt.verlag, 2014.

Transitionsteam

Streben Sie eine umfangreichere Einführung von Agilität im Unternehmen an, hat es sich bewährt, ein Transitionsteam zu bilden, das die Ausbreitung begleitet (siehe Abb. 11).

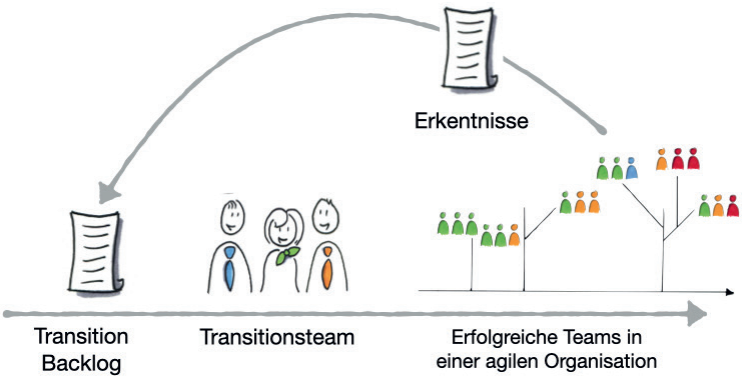


Abb. 11 Transitionsteam

Das Transitionsteam arbeitet nach dem im Unternehmen gewählten agilen Verfahren (z.B. Scrum oder Kanban). Es entwickelt jedoch selbst keine Software, sondern begleitet die gesamte Organisation auf ihrer agilen Reise. Kernaufgabe dieses Teams ist es, Hindernisse, die in der Organisation existieren, zu bewerten und Lösungsmöglichkeiten zu finden. Das Ergebnis (Produkt dieses Teams) sind erfolgreiche selbstorganisierte Teams, die bei ihrer Arbeit kontinuierlich lernen und selbst erkennen, was gut funktioniert und wo organisatorische Hindernisse existieren, die sie selbst nicht beseitigen können. Diese geben Sie an das Transitionsteam weiter, das sie in einem *Transition Backlog* verwaltet.

Um dieses Aufgabe zu erfüllen, ist es wichtig, dass auch das Transitionsteam cross-funktional aufgestellt ist. Für die Begleitung der Transition bedeutet dies eine gute Mischung aus verschiedenen Perspektiven und Entscheidungskompetenzen. Die Besetzung ist von Organisation zu Organisation sehr unterschiedlich. Wichtig ist, dass das Team aufkommende Hindernisse schnell und ohne Abhängigkeiten lösen kann. Oft sieht man, dass es notwendig ist, hinderliche Prozesse abzuschaffen oder auszusetzen, Rahmenbedingungen bei der technischen Ausstattung zu verändern

oder Personen von anderen Aufgaben zu befreien. Hierfür braucht es Entscheidungskompetenzen aus dem Management genauso wie die Teamperspektive, um gute Lösungen zu finden.

Die Initiierung des Transitionsteams erfolgt in der Regel durch das Management in dem Moment, in dem die Transition beschlossen wird. Dann sollten auch die notwendigen Personen gefunden und mit ausreichend Zeit für die Aufgaben in diesem Team ausgestattet werden.

Wie erwachsen sind agile Methoden?

1994 erschien der erste Artikel über Scrum, der jedoch keine besonders große Beachtung fand. 1998 präsentierte Kent Beck eXtreme Programming zum ersten Mal der Öffentlichkeit und erntete mit seinen radikalen Ideen wahre Stürme der Entrüstung. Nur wenige Unerschrockene wagten es damals, eXtreme Programming ernsthaft einzusetzen – und holten sich dabei nicht selten eine blutige Nase.

Seitdem hat sich die Einsicht durchgesetzt hat, dass agile Methoden zwar keineswegs *die* Antwort auf alle offenen Fragen der Softwareentwicklung sind, aber viele dieser Fragen überzeugender beantworten können als klassische Methoden.

Im Laufe der Jahre hat Scrum eine enorm weite Verbreitung gefunden und auch Kanban hat in den letzten Jahren relevante Bedeutung erlangt. Beide Verfahren werden häufig mit XP-Techniken angereichert. Andere agile Vertreter wie z.B. Feature Driven Development oder Crystal sind faktisch bedeutungslos geworden, sodass sie in dieser Auflage unserer Broschüre auch nicht mehr auftauchen.

Nach wie vor stellen überzogene Budgets und weit überschrittene Deadlines in Softwareprojekten eher die Regel als die Ausnahme dar – ganz zu schweigen von den vielen Projekten, die niemals produktiv gehen oder ergebnislos abgebrochen werden. Diese Probleme lassen sich durch kurze Releasezyklen, einen schnellen Systemeinsatz und permanentes Feedback auf den unterschiedlichen Ebenen (Programmierung, Team, Management) gut in den Griff bekommen oder zumindest stark eindämmen.

Agile Methoden sind längst den Kinderschuhen entwachsen und haben viele Kritiker überzeugt. Nach Geoffrey A. Moore[4] durchlaufen erfolgreiche Innovationen fünf Stadien, in denen sie von unterschiedlichen Gruppen angewendet werden: Eine sehr geringe Anzahl von *Innovatoren* macht den Anfang, worauf schon deutlich mehr *Visionäre* folgen. Danach muss die »Kluft« überwunden werden, bevor auch die *Pragmatiker* und die *Konservativen* zu Nutzer:innen werden und somit eine massenhafte Verbreitung stattfindet. Viele Innovationen schaffen diesen Sprung nicht, weil sie zu sehr auf die Bedürfnisse der Innovatoren und Visionäre abgestimmt sind, die sich grundlegend von denen der Pragmatiker und Konservativen unterscheiden.

4. Vgl. Moore, Geoffrey A.: *Crossing the Chasm*. HarperBusiness Essentials, 2006.

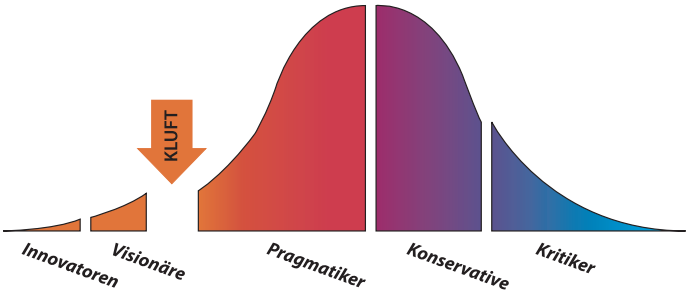


Abb. 12 Agile Methoden haben den Sprung in den Breitenmarkt geschafft.

Die Anwendung agiler Methoden stellt längst kein großes Risiko mehr dar, es wurde bereits vielfach bewiesen, dass sie auch in großen industriellen Projekten zum Erfolg führen. Zur Verbreitung von Agilität hier einige Beispiele:

- Die Standish Group empfiehlt in ihrem Chaos Report bereits seit 2005 explizit agile Methoden, um Softwareprojekte erfolgreich(er) durchzuführen.
- Die meisten Unternehmen in der E-Business-Branche wie z. B. Otto gehen agil vor.
- Es gibt kaum noch ein Softwareunternehmen, das nicht mindestens in Teilbereichen agile Verfahren einsetzt. Die ganz Großen der IT-Branche wie Google, SAP und Microsoft nutzen schon längst intensiv die Vorteile agiler Methoden für sich.
- Agile Verfahren werden auch in sicherheitskritischen Bereichen wie der Medizintechnik verwendet.
- Immer häufiger wird auch Hardware mit agilen Verfahren entwickelt. Als Beispiel ist die Entwicklung eines spritsparenden Pkw durch die Wikispeed-Gruppe zu nennen.
- Seit 2001 haben weltweit mehr als 1,2 Millionen Menschen an Schulungen und Zertifizierungskursen zum Certified Scrum Master oder Certified Product Owner[5] teilgenommen.

5. Quelle: *https://www.scrumalliance.org*.

- Alle relevanten IT-Konferenzen haben sich inzwischen des Themas Agilität angenommen – die meisten bieten eigene Tracks zu agilen Methoden an. Auch haben sich nationale und internationale Konferenzen etabliert, die sich nur um agile Entwicklung kümmern (z. B. Agile Testing Days, Global Scrum Gathering, Lean Kanban Conference, XP Days Germany, Play4Agile).

Unsere eigenen Erfahrungen

Wir setzen agile Methoden (anfangs XP, später auch Scrum und Kanban) seit 1999 ein. Dabei sind wir anfangs gegen so manche Wand gelaufen und mussten manche bittere Lektion lernen. Dabei nicht aufzugeben, sondern uns stetig weiterzuentwickeln, gibt uns heute die Möglichkeit, viele wertvolle Erfahrungen in die Arbeit mit unseren Kund:innen einzubringen.

Zunächst setzten wir in unseren eigenen Entwicklungsprojekten auf agile Verfahren und in den ersten Jahren sind nicht alle Projekte zu unserer vollständigen Zufriedenheit verlaufen. Allerdings ist kein einziges Projekt komplett aus dem Ruder gelaufen oder sogar gescheitert. Und trotz gelegentlicher Meinungsverschiedenheiten mit unseren Kund:innen waren diese zum Schluss doch *immer* zufrieden mit ihrem neuen System.

Über die Jahre hat sich unser Geschäft gewandelt, und zwar hin zur agilen Organisationsberatung und Coaching. Auch hier konnten wir bereits in vielen größeren und kleineren Projekten Erfolge erzielen, wenn sich Kund:innen auf das agile Mindset eingelassen haben. Dabei versprechen wir nicht, dass man mit agilen Methoden doppelt so schnell zum Ziel kommt oder mit weniger Problemen zu tun hat. Wir können jedoch versprechen, dass durch das agile Vorgehen eine nie dagewesene Transparenz im Team und in der Organisation hergestellt wird, was für Agilität Voraussetzung ist. Auch wenn wir dabei immer wieder erfahren mussten, dass das, was an die Oberfläche kommt, unangenehm sein kann (z. B. ist der Fortschritt langsamer als erhofft). Wie stark Sie von agilen Methoden profitieren, hängt am Ende davon ab, wie Sie mit dieser Transparenz umgehen. Wenn Sie jedoch die Augen verschließen und weitermachen wie gewohnt, werden Sie vermutlich kaum Nutzen aus agilen Verfahren ziehen. Nur mit einer offenen Haltung und Experimentierfreudigkeit werden sich viele neue Optionen eröffnen und damit die Chance, nachhaltig Wettbewerbsvorteile und Resilienz des Unternehmens zu erzielen.

Fazit

- Sind agile Methoden also erwachsen geworden? Ja!
- Sind sie einfach einzuführen und stellen sie eine Erfolgsgarantie dar? Nein!

Wer agil vorgehen möchte, sollte deshalb behutsam anfangen und sich immer wieder fragen, ob er auf dem richtigen Weg ist und woraus er die größten Vorteile ziehen kann. Dann werden sich nach und nach auch deutliche Erfolge einstellen.

Vorteile für das Management und die Auftraggeberseite

In den Anfangszeiten der agilen Bewegung waren die Entwickler:innen begeistert. Das Management und die Auftraggeberseite aber waren eher skeptisch. Dieses Bild hat sich längst gewandelt. Auch diese Personengruppen wollen nun die Vorteile agiler Entwicklung nutzen.

Transparenz

Traditionelle Entwicklungsmethoden wie das Wasserfallmodell sind in höchstem Maße intransparent, denn sie kennen nur Kontrollzeitpunkte am Ende der einzelnen Phasen, wobei für Auftraggebende oder das Management die Erstellung der Entwurfsdokumente schwer überprüfbar ist. Wenn sich nun aber aus irgendeinem Grund am Ende herausstellt, dass nicht das System entstanden ist, das die Auftraggebenden eigentlich haben wollen, so bleibt keine Zeit mehr, um zu reagieren. Da helfen auch noch so detaillierte Pflichtenhefte nichts, denn es ist nahezu unmöglich, sich im Voraus alle Features eines größeren Systems haargenau zu überlegen und diese dann auch noch unmissverständlich aufzuschreiben – von sich ständig ändernden Marktsituationen einmal ganz zu schweigen.

Diesem Dilemma setzen agile Methoden häufige kleine Releases und ständige Rückkopplung mit den auftraggebenden Personen entgegen. Diese bekommen so häufig wie möglich lauffähige Systemversionen (keine Testversionen!) zu sehen und können so immer wieder entscheiden, ob sich die Software tatsächlich in die gewünschte Richtung entwickelt.

Auch die Geschwindigkeit und der Projektfortschritt sind in agilen Projekten transparent, weil hier visuelle Trackings in Form von Burndown- oder Burnup-Charts verwendet werden: Die Kund:innen können jederzeit erkennen, wie hoch die aktuelle Geschwindigkeit ist, wie viele Features bereits abgearbeitet sind und wann voraussichtlich alle vereinbarten Features erledigt sein werden. Dabei werden diese Diagramme regelmäßig aktualisiert und geben so ein realistisches Bild wieder.

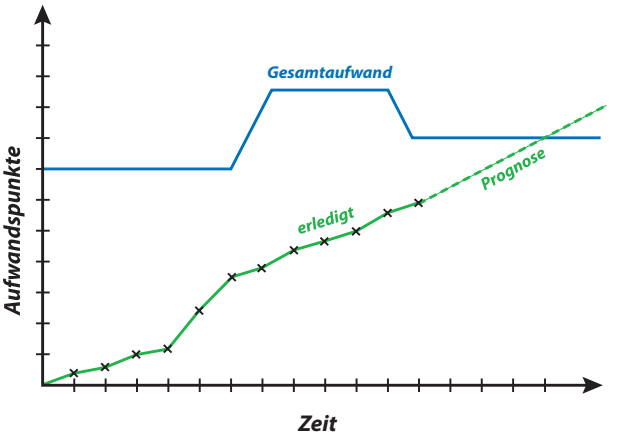


Abb. 13 Durch Feature-Burnup-Charts ist der aktuelle Projektzustand jederzeit sichtbar.

Kürzere Time-to-Market

Untersuchungen zeigen, dass in klassisch entwickelter Software 50-70% der Funktionen selten oder nie genutzt werden. Das zeigt das große Potenzial, das im Weglassen geringwertiger Funktionalitäten liegt. Agile Entwicklungen starten mit dem Allerwichtigsten und fügen schrittweise nur die Funktionalität hinzu, die nützlich ist. Alleine dadurch kann die sogenannte Time-to-Market (also die Zeit von der Idee bis zur Auslieferung) oft schon halbiert werden.

Ein weiterer Aspekt ist die Verschränkung der Entwicklungsaktivitäten. Dadurch, dass mehr oder weniger alles gleichzeitig stattfindet, kann die Entwicklungszeit zusätzlich gestaucht werden.

Angemessenere Software

Der Entwicklungsstand wird immer wieder der Auftraggeberseite und den Nutzer:innen zur Begutachtung vorgelegt. Das Feedback geht in die weitere Entwicklung ein; so stellen wir sicher, dass das entwickelt wird, was tatsächlich benötigt wird. Außerdem können veränderte Marktsituationen noch während der Entwicklung berücksichtigt werden.

Höhere Qualität

Qualität steht kontinuierlich im Fokus der Betrachtung und wird nicht ans Projektende verschoben. Die Qualitätssicherung einer Funktionalität findet direkt nach ihrer Entwicklung statt. So werden Fehler bereits wenige Stunden oder Tage, nachdem sie produziert wurden, entdeckt. Dadurch sind sie um Größenordnungen leichter zu lokalisieren und zu beheben, als wenn man sie erst nach Wochen oder Monaten entdeckt hätte. Insgesamt erhöht sich das Qualitätsbewusstsein bei allen Beteiligten.

Höhere Produktivität

Neben dem Einsparen nicht benötigter Funktionen kann agile Entwicklung auch produktiver sein als im klassischen Vorgehen. Durch die kontinuierliche Kommunikation der Beteiligten wird weniger in die falsche Richtung entwickelt. Durch die enge Einbindung der Qualitätssicherung werden Fehler früher entdeckt und können dadurch viel kostengünstiger beseitigt werden. Außerdem lernen alle Beteiligten ständig dazu und können sich so immer besser einbringen.

Mehr Innovation

Die Zusammenarbeit in cross-funktionalen Teams hoher Diversität hat ein großes Innovationspotenzial.

Vorteile für die Entwicklung

Nicht nur für das Management sowie die Auftraggeberseite bieten agile Entwicklungsmethoden wichtige Vorteile, sondern auch für die Entwicklung.

Agile Methoden führen zur direkten Kommunikation mit den Nutzer:innen
In agilen Projekten sprechen die Entwickler:innen direkt mit den Kunden vor Ort – und das am besten täglich. Diese direkte Kommunikation führt nicht nur dazu, dass die Entwickler:innen besser verstehen, was die Kundenseite eigentlich für eine Software benötigt; sondern dadurch wächst auch das Verständnis der Nutzer:innen, wenn einmal eine Schätzung aus der Entwicklung korrigiert werden muss. Insgesamt sind eine gute Zusammenarbeit der beteiligten Gruppen sowie eine gemeinsame Vertrauensbasis von unschätzbarem Wert für jedes Projekt. Natürlich kann es mitunter schwieriger sein, es Menschen anstatt Plänen gerecht zu machen, aber dafür ist es auch ungleich befriedigender.

Wer agil vorgeht, liefert Qualitätsarbeit ab
Natürlich möchten alle Entwickler:innen sich mit ihrer Arbeit identifizieren und wirklich gute Software erstellen. Durch den engen Kontakt mit der Auftraggeber- und der Kundenseite wird für die Entwickler:innen erfahrbar, wie sie Wert für sie schaffen und eine Agilität wirkungsvoll leben können (vgl. Kapitel 1). Das wirkt sinnstiftend und motivierend.

Bei der Erreichung der kundenorientierten Ziele sind die Entwickler:innen autonom und entscheiden selbst, wie sie diese Ziele erreichen. So haben sie die Möglichkeit, Qualitätsarbeit abzuliefern, und müssen keine faulen Qualitätskompromisse eingehen, die ihnen später auf die Füße fallen. Auf der handwerklichen Seite hilft testgetriebene Entwicklung, Pair Programming und Continuous Integration dabei, hohe interne Qualität bei der Produktabwicklung abzuliefern.

Mehr Spaß bei der Arbeit durch agiles Vorgehen
Zugegeben: Alle Menschen sind verschieden, und was dem einen Spaß bringt, kann für den anderen ein Graus sein. Dennoch zeigt die Erfahrung, dass die meisten Entwickler:innen in agilen Projekten mehr motiviert sind als in klassischen Projekten. Hierzu dürfte die enge Zusammenarbeit mit den anderen Teammitgliedern (tägliche Standup-Meetings, Retrospekti-

ven, Pair Programming) ebenso beitragen wie die direkte Kommunikation mit der Kundenseite, das häufige Ausliefern neuer Versionen (und damit verbundene Erfolgserlebnisse), die eigenverantwortliche Arbeit oder die Möglichkeit eines lebenslangen Lernens z.B. durch Experimente.

Skalierung

Mit dem Agilen Manifest ist nicht nur ein Wandel der Mechanik im Entwicklungsbereich eingeläutet worden (kurze Iterationen, die lauffähige Software erzeugen), sondern damit verbunden sind grundsätzlich auch veränderte Verhaltensweisen bei allen Beteiligten, also ein tiefgreifender *Kulturwandel* im Unternehmen. Die mechanische Anwendung agiler Praktiken unternehmensweit ohne einen solchen Kulturwandel bringt letztendlich nur marginale Effekte.

Kulturwandel kann man nicht verordnen. Die Unternehmenskultur wird im Wesentlichen durch das Verhalten aller Mitarbeitenden – das Management eingeschlossen – geprägt. Doch wie ist die Fehlerkultur im Unternehmen? Wofür wird Anerkennung gezollt? Wie ist der Umgangston? Welche gemeinsamen Rituale gibt es? Wie wird mit Konflikten umgegangen? Etc. Tatsächlich findet also ein Wandel vor allem auch über Verhaltensänderungen auf Managementseite (Stichwort: Agile Leadership) und in den Teams statt.

In einem größeren Kontext die Unternehmenskultur mit der Gießkanne zu ändern, ist extrem aufwendig und risikoreich. Es ist daher sinnvoll, Agilität und somit den Wandel schrittweise im Unternehmen auszubreiten – ausgehend von einer oder mehreren agilen »Keimzellen«.

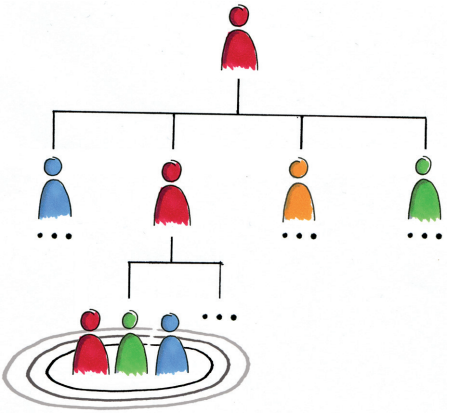


Abb. 14 Die neue Kultur breitet sich schrittweise von Keimzellen aus.

Diese Keimzellen sind Teams, die *verlässlich lieferbare Produktinkremente* entwickeln. Solange diese Fähigkeit im Team nicht vorhanden ist, würde die Skalierung zu dem führen, was Jerry Weinberg als das erste Gesetz schlechten Managements charakterisiert:

Wenn etwas nicht funktioniert, mach' mehr davon!

Für ein organisches Ausbreiten von Keimzellen aus hat es sich bewährt, mit einem Team zu beginnen, dieses nach einer Weile aufzuteilen und die entstandenen neuen zwei bis drei Teams mit neuen Teammitgliedern aufzufüllen. Wenn diese neuen Teams erfolgreich agil gearbeitet und die neue Kultur verinnerlicht haben, werden sie nach dem gleichen Schema wieder aufgeteilt usw. So wird das agile Mindset im persönlichen Arbeitskontakt der Teammitglieder transportiert. Dafür nehmen wir die erhöhten Aufwände für Teambuilding in Kauf – in der Praxis eine Rechnung, die gut aufgeht.

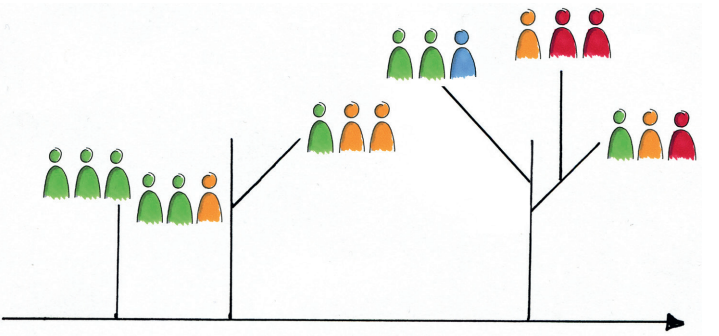


Abb. 15 Organische Ausbreitung von Verhaltensweisen und Erfahrungen

Für die Verbreitung der agilen Kultur im Unternehmen eignen sich außerdem regelmäßige *Open-Space-Veranstaltungen* und *Communities of Practice*, bei denen alle teilnehmen, lernen und Erfahrungen austauschen können. Diese beiden Instrumente werden umso wichtiger, je stärker von der dargestellten organischen Ausbreitung durch Mitarbeiterrotation abgewichen wird.

Wichtig dabei ist, im Blick zu behalten, dass die primäre Herausforderung der agilen Skalierung nicht in den konkreten Praktiken für die Koordination mehrerer Teams liegt, sondern in der Ausbreitung der agilen Kultur.

Der Agile Scaling Cycle

Bei der Entwicklung einer agilen Unternehmenskultur im Kleinen muss auch die Frage geklärt werden, wie die Teams in einem Großprojekt organisiert und koordiniert werden können. Dabei ist darauf zu achten, dass die Agilität, die auf Ebene eines Teams erreicht wurde, nicht bei der Skalierung durch klassische hierarchische Steuerungsmechanismen wieder verloren geht. Die Antwort kann auch nicht darin bestehen, einen Blueprint zu kopieren: Schließlich liegt der agilen Entwicklung die Annahme zugrunde, dass sich die optimale Struktur schrittweise entwickeln muss, um den optimalen Nutzen zu gewährleisten.

Der *Agile Scaling Cycle* (siehe Abb. 16) setzt diese Denkweise in ein zyklisches Vorgehen mit drei Schritten um. Im Zentrum stehen die agilen Werte und Prinzipien, die wir bei jedem der drei Schritte als Kompass verwenden. Wir beginnen ein neues skaliertes Projekt damit, dass wir *Abhängigkeiten reduzieren*, soweit es möglich ist. Anschließend arbeiten wir im Projekt und *koordinieren die Teams* bzgl. der verbliebenen fachlichen und technischen Abhängigkeiten. Auf Basis der im Projekt gewonnenen Erkenntnisse *entwickeln wir die Organisation* weiter, sodass wir im nächsten Zyklus des *Agile Scaling Cycle* mehr Optionen zur Reduktion von Abhängigkeiten haben. Für die Weiterentwicklung der Organisation zeichnet das weiter oben beschriebene Transitionsteam verantwortlich.

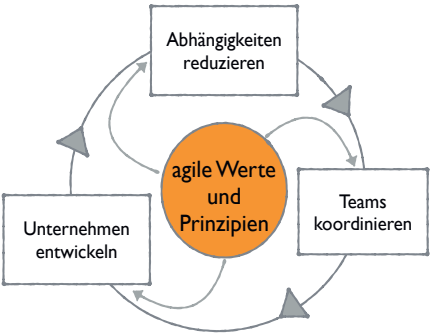


Abb. 16 Agile Scaling Cycle

Sowohl für die Reduktion von Abhängigkeiten wie auch für die Koordination der Teams stehen Dutzende von Praktiken zur Verfügung, aus denen Sie sich anfänglich bedienen können[6]. Und je häufiger Sie den Agile Scaling Cycle durchlaufen, desto mehr eigene Praktiken werden sich herausbilden, die immer besser auf Ihren speziellen Kontext angepasst sind.

6. Siehe *http://scaledprinciples.org*.

it-agile

www.it-agile.de info@it-agile.de 0800/itagile

Sandra Sieroux, Stefan Roock, Henning Wolf

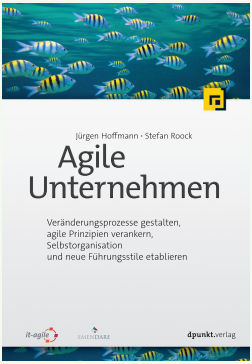
Agile Leadership

Führungsmodelle, Führungsstile und das richtige Handwerkszeug für die agile Arbeitswelt

Das Autorenteam gibt einen fundierten, praxisorientierten Überblick, wie agile Organisationsstrukturen aussehen können und wie der Weg dorthin mittels Agile Leadership gestaltet werden kann.

Sie finden in diesem Buch einen Werkzeugkasten an Methoden, aus dem Sie sich bedienen können, um den individuellen Wandel Ihrer Organisation zu begleiten.

2020, 312 Seiten, komplett in Farbe, Broschur
Euro 34,90 (D) / ISBN 978-3-86490-696-1



Jürgen Hoffmann, Stefan Roock

Agile Unternehmen

Veränderungsprozesse gestalten, agile Prinzipien verankern, Selbstorganisation und neue Führungsstile etablieren

Agile Unternehmen agieren flexibler am Markt, entwickeln begeisternde Produkte und bieten Mitarbeiter*innen sinnstiftendere Arbeitsplätze. Die Autoren beschreiben, was agile Unternehmen ausmacht, und bieten konkrete Praktiken an, mit denen das eigene Unternehmen schrittweise agiler gestaltet werden kann – mit vielen Fallbeispielen aus der Praxis.

2018, 214 Seiten, komplett in Farbe, Broschur
Euro 29,90 (D) / ISBN 978-3-86490-399-1

Fritz-Ulli Pieper, Stefan Roock

Agile Verträge

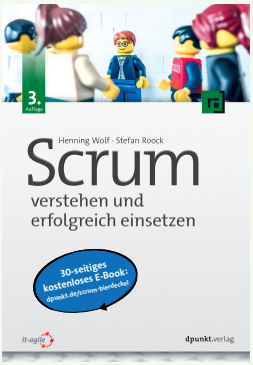
Vertragsgestaltung bei agiler Entwicklung für Projektverantwortliche

Dieses Buch bietet Ihnen einen kompakten Einstieg in die vertragsrechtlichen Grundlagen bei agiler Softwareentwicklung, die verschiedenen Varianten der Vertragsgestaltung sowie die einzelnen Vertragsformen mit ihren Eigenschaften, Funktionsweisen, Vorteilen und Risiken, wobei auch eine formalrechtliche Einordnung vorgenommen wird.

2017, 168 Seiten, komplett in Farbe, Broschur
€ 26,90 (D) / ISBN 978-3-86490-400-4

dpunkt.bücher für Softwerker

dpunkt.verlag GmbH, Wieblinger Weg 17, 69123 Heidelberg, www.dpunkt.de, bestellung@dpunkt.de



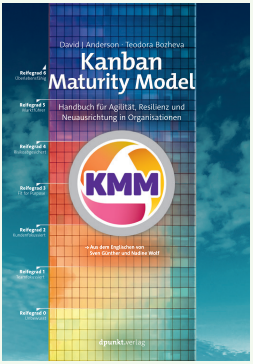
Henning Wolf, Stefan Roock

Scrum – verstehen und erfolgreich einsetzen

3., überarbeitete und erweiterte Auflage

Die Autoren beschreiben in kompakter Form die Scrum-Grundlagen und die hinter Scrum stehenden Werte und Prinzipien sowie die kontinuierliche Prozessverbesserung. Die 3. Auflage wurde an die neue Fassung des Scrum Guide aus November 2020 angepasst, was neben Details auch das Konzept des Produktziels umfasst. Zudem gibt es einen Ausblick auf das Konzept der wirkungsvollen Agilität.

2021, 264 Seiten, komplett in Farbe, Broschur
Euro 29,90 (D) / ISBN 978-3-86490-848-4



David J Anderson, Teodora Bozheva

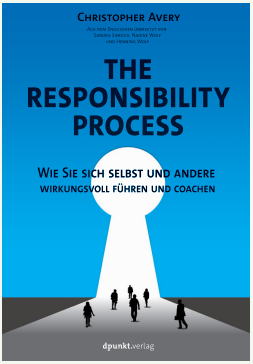
Kanban Maturity Model

Handbuch für Agilität, Resilienz und Neuausrichtung in Organisationen

Aus dem Englischen von Sven Günther und Nadine Wolf

Das Kanban Maturity Model entstand durch die Arbeit in der letzten Dekade bei der Einführung von Kanban in kleinen und großen Unternehmen verschiedener Branchen. Es spiegelt die Erfahrung wider, dass die angewendeten Kanban-Praktiken zur organisatorischen Reife des Unternehmens passen müssen. Die KMM-Roadmap und konkrete Maßnahmen ermöglichen es, die gewünschte Business-Agilität zu erreichen.

2021, 648 Seiten, komplett in Farbe, Festeinband
Euro 59,90 (D) / ISBN 978-3-86490-608-4



Christopher Avery

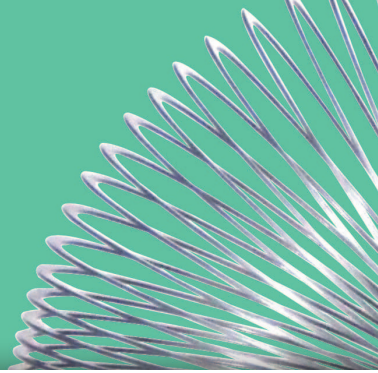
The Responsibility Process

Wie Sie sich selbst und andere wirkungsvoll führen und coachen

Aus dem Englischen von Sandra Sieroux, Nadine Wolf und Henning Wolf

Der Autor zeigt mit The Responsibility Process™ den Weg und die Zwischenschritte hin zu echter Verantwortungsübernahme. Er gibt Ihnen konkrete Werkzeuge, Praktiken und Leadership-Weisheiten an die Hand, mit denen Sie lernen, diesen Prozess bewusst einzusetzen, um sich selbst und anderen kraft- und wirkungsvolles Handeln zu ermöglichen.

2019, 294 Seiten, Broschur
Euro 24,90 (D) / ISBN 978-3-86490-577-3



Henning Wolf · Stefan Roock · Stefan Zumbrägel

Agile Softwareentwicklung

Agile Softwareentwicklung ist in aller Munde. Aber was verbirgt sich hinter diesem Schlagwort? Was ist unter »agiler Vorgehensweise« tatsächlich zu verstehen? Diese Broschüre möchte Ihnen eine Hilfe bieten, um die Vorteile agiler Methoden für Ihr Business besser einschätzen zu können.

Sie lernen die Grundlagen agiler Softwareentwicklung kennen und was unter wirkungsvoller Agilität zu verstehen ist. Weiter werden Sie den drei bekanntesten agilen Methoden Scrum, eXtreme Programming (XP) und Kanban begegnen. Dabei erfahren Sie, welche der drei Methoden unter welchen Voraussetzungen die geeignete ist.

Zur Abrundung haben wir für Sie die Vorteile agilen Vorgehens für die Management- und Auftraggeberseite einerseits und die Entwicklung andererseits gegenübergestellt. Und da agile Methoden immer häufiger in großen Projekten eingesetzt werden, thematisieren wir auch die Frage der agilen Skalierung.

Art.-Nr.: 077.95720
Schutzgebühr: 3,00 €

www.dpunkt.de