2 Agile Werte und Prinzipien

Mindset trifft Methode

Es gibt kein »perfektes« Rezept, um großartige Software zu bauen.

Manche Teams haben nach der Übernahme agiler Praktiken, Methoden und Vorgehensweisen viel Erfolg, während andere Probleme bekommen. Wir haben gelernt, dass der Unterschied im Mindset der Teammitglieder liegt. Was tun Sie also, wenn Sie diese großartigen agilen Ergebnisse in Ihrem eigenen Team erreichen wollen? Wie stellen Sie sicher, dass Ihr Team die richtige Denkweise besitzt? Hier kommt das **Agile Manifest** ins Spiel. Wenn Sie und Ihr Team dessen **Werte und Prinzipien** verinnerlichen, beginnen Sie, anders über die agilen Praktiken und ihre Funktionsweise zu denken, und diese werden nun *viel effektiver*.

In Snowbird ist etwas Wichtiges passiert

In den 1990ern gab es in der Welt der Softwareentwicklung eine wachsende Bewegung. Die Teams waren den klassischen Weg der Softwareentwicklung über einen **Wasserfall-Prozess** zunehmend müde. Dabei wurden zunächst strikte Anforderungen definiert, dann wurde ein vollständiges Design entworfen und schließlich die gesamte Softwarearchitektur auf Papier geschaffen, bevor der Code geschrieben wurde.

Den Leuten in Wasserfall-Teams war nicht immer zu 100 % klar, warum genau sie den Prozess nicht mochten, aber man war sich einig, dass er irgendwie zu mühsam, zu >>bleiern<< war.

Am Ende des Jahrzehnts herrschte zunehmend Konsens darüber, dass Teams einen »schlankeren« Ansatz zum Bauen von Software bräuchten, und diverse Vorgehensweisen – insbesondere Scrum und XP – verbreiteten sich immer stärker.

Treffen der klugen Köpfe

2001 traf sich eine Gruppe von 17 aufgeschlossenen Leuten im Ski-Resort Snowbird in den Bergen bei Salt Lake City in Utah. Zur Gruppe gehörten Vordenker verschiedenster »schlanker« Arbeitsmethoden, unter anderem die Erfinder von Scrum und XP. Man war sich zwar nicht wirklich sicher, was bei diesem Treffen herauskommen würde, aber man war sich ziemlich sicher, dass diese neuen schlanken Methoden zum Bauen von Software etwas gemeinsam hatten. Nun wollten sie herausfinden, ob sie recht hatten, und vielleicht einen Weg finden, diese Gemeinsamkeiten aufzuschreiben.

Branchen-Leader kamen zusammen, um sich zu überlegen, ob die verschiedenen und zunehmend beliebteren schlanken Methoden zum Bauen von Software etwas gemeinsam hätten.

Das Agile Manifest

Die Gruppe brauchte nicht lange, um sich auf vier gemeinsame Werte zu einigen. Diese Werte schrieben sie auf und nannten das Ganze das **Agile Manifest**.

Manifest für Agile Softwareentwicklung

Wir erschließen bessere Wege, um Software zu entwickeln,
indem wir es selbst tun und anderen dabei helfen.
Durch diese Tätigkeit haben wir diese Werte zu schätzen gelernt:

Individuen und Interaktion mehr als Prozesse und Werkzeuge
Funktionierende Software mehr als umfassende Dokumentation
Zusammenarbeit mit dem Kunden mehr als Vertragsverhandlung
Reagieren auf Veränderung mehr als das Befolgen des Plans

Das heißt, obwohl wir die Werte auf der rechten Seite wichtig finden,
schätzen wir die Werte auf der linken Seite höher ein.

Wenn diese Jungs so klug waren – warum haben sie dann nicht einfach den **besten Weg zur Softwareentwicklung** aufgeschrieben? Warum sollte ich mich überhaupt mit diesen ganzen »Werten« befassen?

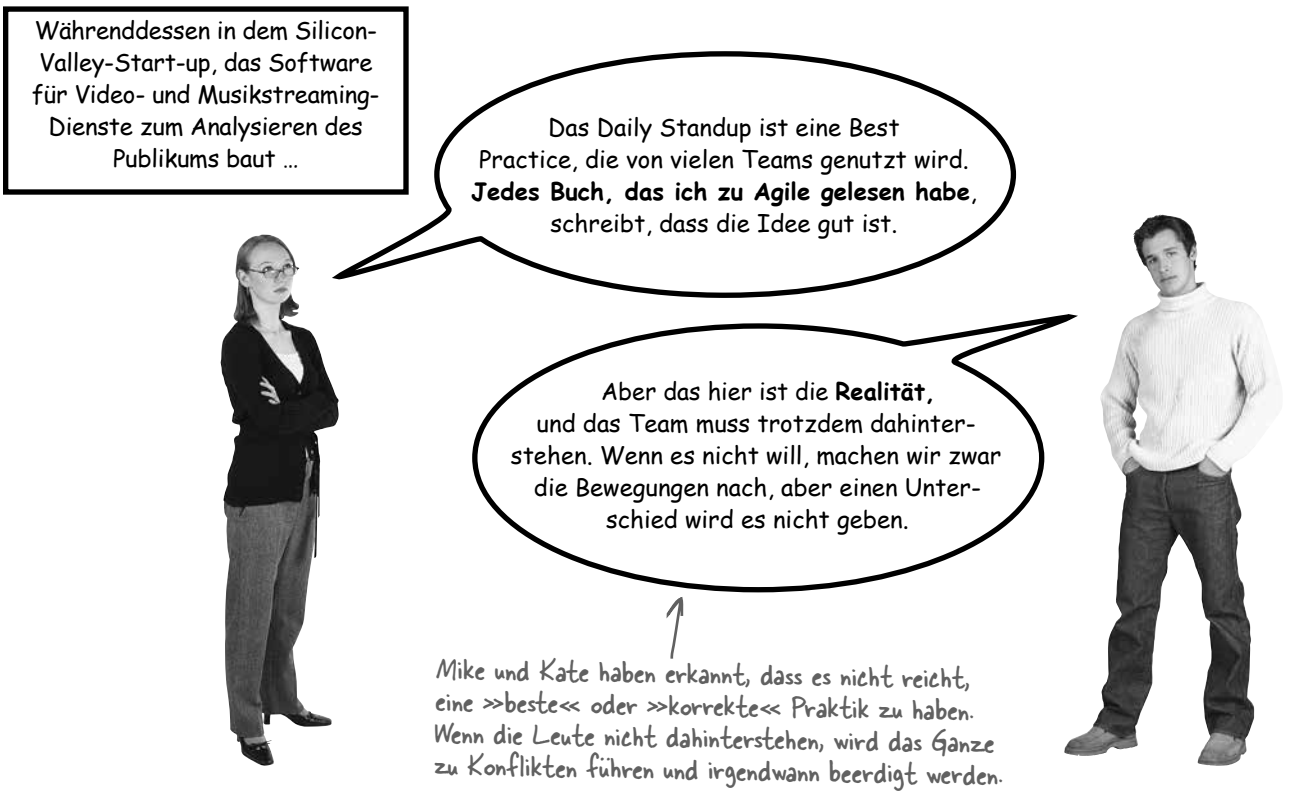
Die Idee, dass es keinen Königsweg für das Bauen von Software gibt, wurde erstmals in den 1980ern durch einen Pionier der Softwareentwickung, Fred Brooks, in einem Essay namens »No Silver Bullet« aufgebracht.

Sie haben nicht versucht, eine »vereinheitlichte« Vorgehensweise zu entwerfen.

Eine der grundlegenden Ideen moderner Softwareentwicklung ist, dass **es nicht den einen, besten Weg zum Bauen von Software gibt**. Das ist eine wichtige Idee, die in der Softwarebranche schon seit Jahrzehnten gärte. Das Agile Manifest ist effektiv, weil es ***Werte beschreibt, die Teams dabei helfen, ein agiles Mindset zu entwickeln***. Wenn jeder im Team diese Werte aufrichtig in seine Denkweise übernimmt, wird es tatsächlich dabei helfen, bessere Software zu bauen.

Es kann eine Herausforderung sein, Praktiken in der Realität umzusetzen

Teams suchen immer nach Wegen, besser zu werden. Wir haben gesehen, wie Praktiken dabei helfen können. Das gilt besonders für die schlanken Praktiken, die bei agilen Teams zum Einsatz kommen und die dazu gedacht sind, einfach, geradlinig und leicht umgesetzt werden zu können. Aber wir haben auch gesehen, dass das Mindset oder das Verhalten des Teams es viel schwieriger machen kann, sie erfolgreich umzusetzen – zum Beispiel als Kate erkannte, dass die Verhaltensweisen von ihr, Mike und dem Rest des Teams beim Versuch, ein Daily Standup durchzuführen, einen großen Einfluss hatte.



Die vier Werte des Agilen Manifests leiten das Team auf ihrem Weg zu einem besseren und effektiveren Mindset

Das Agile Manifest enthält vier »X mehr als Y«-Zeilen, die uns helfen, zu verstehen, was agile Teams wertschätzen. Jede dieser Zeilen sagt uns etwas Bestimmtes über die Werte, die einem agilen Mindset zugrunde liegen. Wir können sie einsetzen, um besser zu verstehen, was es für ein Team heißt, agil zu sein.

Schauen wir uns jeden dieser vier Werte genauer an. →

Individuen und Interaktion mehr als Prozesse und Werkzeuge

Agilen Teams ist klar, dass Prozesse und Werkzeuge wichtig sind. Sie haben schon ein paar Praktiken kennengelernt, die agile Teams nutzen: Daily Standups, User Stories, Taskboards, Burndown Charts, Refaktorieren und Retrospektiven. Das sind alles wertvolle Werkzeuge, die für ein agiles Team sehr hilfreich sein können.

Aber agile Teams schätzen Individuen und Interaktion noch höher ein als Prozesse und Werkzeuge, weil Teams immer dann am besten arbeiten, wenn Sie auf die menschliche Seite achten.

Sie haben schon ein Beispiel dafür gesehen – als Kate versuchte, Daily Standups zu etablieren, und dies zu einem Konflikt mit Mike und seinem Entwicklungsteam führte. Das liegt daran, dass ein Werkzeug, das einem Team sehr geholfen hat, bei einem anderen Team zu ernsthaften Problemen führen kann, wenn die Leute dort keinen Vorteil darin sehen und es ihnen nicht direkt dabei hilft, Software zu bauen.

Das nächste Mal, wenn ich versuche, ein neues Werkzeug oder einen neuen Prozess einzuführen, werde ich mit dem Team sprechen und versuchen, **seine Sichtweise zu verstehen**.

Gute Idee, Kate!

Prozesse und Tools sind wichtig, um das Projekt umzusetzen, und sie können sehr wertvoll sein. Aber die **individuellen Personen im Team** sind noch wertvoller, und jedes Werkzeug, das Sie einführen, muss dazu dienen, die **Interaktion** der Teammitglieder untereinander und mit ihren Anwendern und Stakeholdern zu verbessern.

Funktionierende Software mehr als umfassende Dokumentation

Was heißt »funktionierende« Software? Woher wissen Sie, ob Ihre Software funktioniert? Das ist tatsächlich eine schwerere Frage, als Sie vielleicht denken. Ein klassisches Wasserfall-Team beginnt ein Projekt damit, umfangreiche Anforderungsdokumente zu schreiben, um festzulegen, was das Team bauen wird. Diese Dokumentation wird anschließend zusammen mit den Anwendern und Stakeholdern begutachtet und dann an die Entwickler weitergereicht, damit diese die Software bauen.

Die meisten professionellen Softwareentwickler hatten schon furchtbare Meetings, in denen das Team die gerade entwickelte Software stolz vorstellen wollte, nur um festzustellen, dass sich ein Anwender darüber beschwert, dass ein wichtiges Feature fehlen oder etwas nicht ordentlich funktionieren würde. Häufig endet das in einer Diskussion wie der, die Ben und Mike führten, nachdem Mike ein Feature präsentiert hatte, an dem sein Team ein paar Monate gearbeitet hat:

Es gibt den alten Programmierer-Witz, dass Bugs schlicht undokumentierte Features seien. Viele Bugs treten auf, weil die Programmierer der Meinung waren, die Software sollte so funktionieren, während die Anwender sie anders erwarteten.

Viele versuchen, dieses Problem durch eine umfangreiche Dokumentation zu beseitigen, aber das kann die Situation tatsächlich noch schlimmer machen. Das Problem mit der Dokumentation ist, dass zwei Leute denselben Text lesen, diesen aber völlig unterschiedlich interpretieren können.

Darum wertschätzen agile Teams **funktionierende Software** mehr als umfangreiche Dokumentation – weil sich gezeigt hat, dass dies für den Anwender der effektivste Weg ist, herauszufinden, ob sich die Software wirklich sinnvoll einsetzen lässt.

KRAFT-TRAINING

Dieses kleine Rätsel sollte jedem vertraut sein, der *Head First PMP* gelesen hat!

Lisa testet die Firmwarekomponente für die Black Box 3000™. Das Produkt »funktioniert« nur in einem dieser Szenarien. Können Sie Lisa dabei helfen, herauszufinden, welche Version des Produkts die »funktionierende« Firmware hat?

Black Box 3000™

Ein kleiner Tipp: Eine wichtige Information haben wir Ihnen nicht gegeben. Ohne sie ist dieses Rätsel nur schwer zu lösen.

Manche würden sogar sagen, es ist unmöglich!

Szenario 1

Lisa drückt den Knopf, aber nichts passiert.

Also ... woher weiß ich, ob die Firmware in der Box funktioniert?

Szenario 2

Lisa drückt den Knopf, und eine Stimme sagt: »Sie haben den Knopf falsch gedrückt.«

Unsere Testerin Lisa testet die Firmware der Black Box 3000™, ist sich aber nicht sicher, auf was sie testen soll.

Szenario 3

Lisa drückt den Knopf, und die Box erhitzt sich auf 333 °C. Sie lässt die Box fallen, und diese zerspringt in tausend Teile.

Falls Sie den Begriff nicht kennen: **Firmware** ist Software, die im nur lesbaren Speicher einer Hardware abgelegt ist.

KRAFT-TRAINING

ERKLÄRUNG

Hier die entscheidende, aber fehlende Information, die wir Ihnen auf der vorigen Seite nicht gegeben haben: Die Black Box 3000™ ist das Heizelement für einen Industrieofen. Daher ist Szenario 3 dasjenige, das »funktionierende« Software demonstriert.

Der beste – und manchmal einzige! – Weg, herauszufinden, ob Software funktioniert, ist, sie in die Hände derjenigen zu geben, die sie einsetzen müssen. Wenn diese die Software dazu bringen können, das zu tun, was sie tun soll, funktioniert sie. Aber es ist nicht immer leicht, genau zu wissen, was »funktionierend« genau bedeutet. Deshalb schätzen agile Teams *auch* eine umfangreiche Dokumentation – nur ist ihnen die funktionierende Software wichtiger.

In diesem Fall wortwörtlich in Lisas Hände. Hoffentlich trug sie hitzebeständige Handschuhe!

Hier ein Beispiel für eine umfangreiche Dokumentation, die das Team tatsächlich als nützlich erachtet hat: die Spezifikation für die Black Box 3000™.

BLACK BOX 3000™
Spezifikation

Die BB3K™ ist ein Heizelement für einen Industrieofen.

Die BB3K™ muss sich in 0,8 Sekunden auf exakt 333 °C aufheizen.

Die BB3K™ muss einen großen, leicht zu drückenden Knopf haben.

Manchmal kann Dokumentation nützlich sein – wenn zum Beispiel nicht ganz klar ist, was die »funktionierende« Software tun soll.

Es scheint, dass Szenario 3 das mit der funktionierenden Software ist. Gut, dass ich die Dokumentation hatte, in der das stand … aber es ist für mich noch wichtiger, dass ich das **tatsächliche Produkt** in meinen Händen halten konnte.

Jetzt, da sie weiß, was für diese Software »funktionierend« heißt, kann Lisa sie wirklich testen.

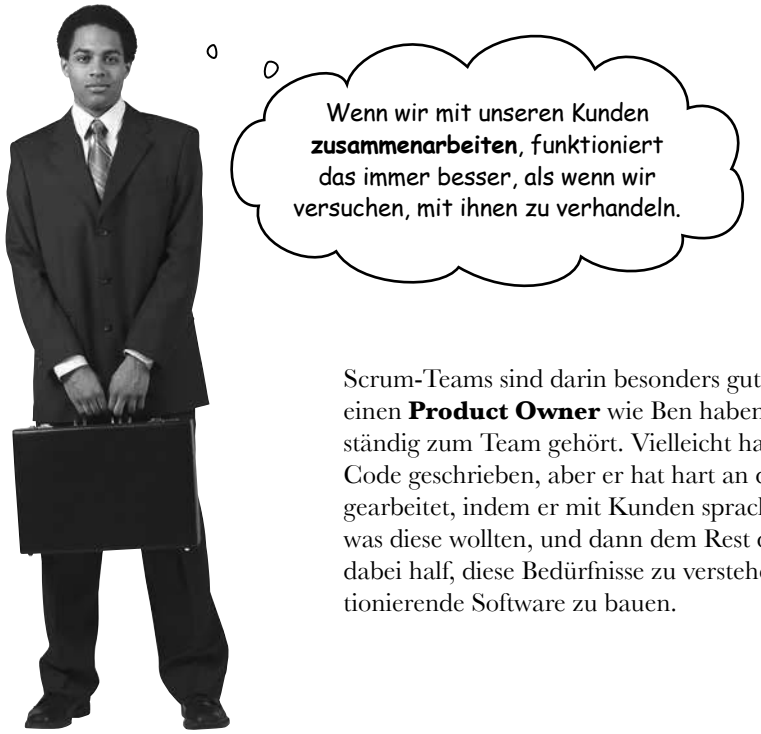
Zusammenarbeit mit dem Kunden mehr als Vertragsverhandlung

Nein, es geht *nicht* um Berater oder Beschaffungsteams, die mit Verträgen zu tun haben!

Wenn man in agilen Teams von Vertragsverhandlungen spricht, meint man häufig ein Verhalten gegenüber Anwendern, Kunden oder Personen in anderen Teams. Wenn Leute in einem Team eine »Vertragsverhandlungs«-Mentalität besitzen, sind sie der Meinung, sie müssten eine exakte Vereinbarung darüber treffen, was das Team bauen oder tun wird, bevor irgendeine Arbeit beginnen kann. Viele Firmen unterstützen diese Denkweise und fordern Teams auf, explizite »Vereinbarungen« darüber zu treffen (oft als Spezifikation dokumentiert und durch strikte Änderungssteuerungsprozesse überwacht), was sie wann liefern werden.

Agile Teams bewerten eine Zusammenarbeit mit dem Kunden höher als Vertragsverhandlungen. Sie wissen, dass sich Projekte ändern werden und dass die Beteiligten zu Beginn eines Projekts nie perfekt mit Informationen versorgt sind. Anstatt also zu versuchen, schon vor dem Start genau festzuhalten, was gebaut werden wird, **arbeiten sie mit ihren Kunden zusammen**, um bestmögliche Ergebnisse zu erhalten.

Vertragsverhandlungen sind notwendig, wenn Kunden nicht kooperieren wollen. Es ist sehr schwierig, wirklich gut mit jemandem zusammenzuarbeiten, der unvernünftig ist – wie zum Beispiel mit einem Kunden, der immer wieder den Scope des Projekts ändert, dem Team aber nicht genug Zeit geben will, diese Änderungen auch umzusetzen.



Scrum-Teams sind darin besonders gut, weil sie einen **Product Owner** wie Ben haben, der vollständig zum Team gehört. Vielleicht hat er keinen Code geschrieben, aber er hat hart an dem Projekt gearbeitet, indem er mit Kunden sprach, verstand, was diese wollten, und dann dem Rest des Teams dabei half, diese Bedürfnisse zu verstehen und funktionierende Software zu bauen.

Reagieren auf Veränderung mehr als das Befolgen eines Plans

Manche Projektmanager folgen dem Motto: »Plane die Arbeit, arbeite den Plan ab.« Und agile Teams erkennen durchaus an, dass Planung wichtig ist. Aber das Abarbeiten eines Plans, der problematisch ist, sorgt dafür, dass das Team ein Produkt mit Problemen baut.

Klassische Wasserfall-Projekte haben Vorgehensweisen, um Änderungen zu verwalten, aber diese beinhalten im Allgemeinen strikte und zeitaufwendige Prozeduren zur Änderungskontrolle. Das spiegelt eine Denkweise wider, in der Änderungen die Ausnahme sind und nicht die Regel.

Das Problem mit Plänen ist, dass sie zu Beginn eines Projekts erstellt werden – dann weiß das Team aber noch am wenigsten über das zu bauende Produkt. Daher **erwarten** agile Teams, dass sich **ihre Pläne ändern werden**.

Aus diesem Grund nutzen sie meist Vorgehensweisen und Werkzeuge, die ihnen helfen, fortlaufend auf Änderungen zu achten und auf diese zu reagieren. Sie haben schon solch ein Werkzeug gesehen: das Daily Standup.

In agilen Projekten wird Ihr Produkt Schritt für Schritt entwickelt, wobei jeder neue Schritt auf dem Wissen aus dem vorigen Schritt aufbaut. Wird auf diese Art und Weise ein Plan entwickelt (oder eine Anforderung oder irgendetwas anderes, das Sie in einem Projekt nutzen), nennt man das *fortschreitende Ausarbeitung*.

Bevor wir mit dem Daily Standup begonnen hatten, erfuhr ich von Problemen am Projektplan immer erst, wenn es zu spät war, um auf sie reagieren zu können.

Nachdem Kate und Mike ihre Meinungsverschiedenheiten hatten klären können, erkannten beide, dass das Daily Standup für alle eine Möglichkeit war, sich täglich den Plan anzuschauen und zusammen auf erforderliche Änderungen zu reagieren. Wenn das ganze Team gemeinsam auf Änderungen reagiert, kann es auch den Plan anpassen, an dem es gerade arbeitet, ohne das Projekt im Chaos versinken zu lassen.

Es ist wichtig, Ihr Projekt zu planen, aber es ist noch wichtiger, zu erkennen, dass sich diese Pläne ändern werden, sobald das Team beginnt, am Code zu arbeiten.

Die Reaktion auf Veränderung ist für agile Teams wichtig, aber dennoch ist es entscheidend, dass sie einem Plan folgen.

Schauen Sie sich nochmals den letzten Satz des Agilen Manifests an:

> Das heißt, obwohl wir die Werte auf der rechten Seite wichtig finden, schätzen wir die Werte auf der linken Seite höher ein.

Jeder der vier Werte des Agilen Manifests besteht aus zwei Teilen: etwas (auf der rechten Seite), das agile Teams wertschätzen, und etwas anderes (auf der linken Seite), das ihnen noch wichtiger ist.

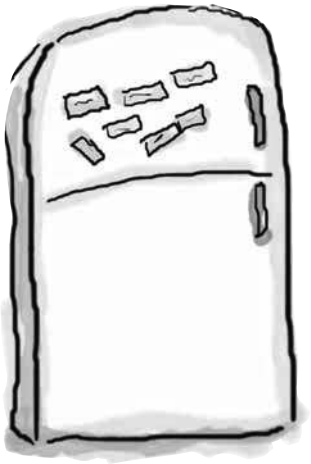
*Wenn also agile Teams sagen, dass es ihnen wichtiger ist, auf Veränderungen zu reagieren, als einem Plan zu folgen, heißt das **nicht**, dass sie eine Planung nicht wertschätzen – ganz im Gegenteil! Sie finden es toll, einem Plan zu folgen. Es ist ihnen nur noch wichtiger, auf Veränderungen zu reagieren.*

Tatsächlich planen Scrum-Teams sogar noch mehr als klassische Teams mit ihrem Wasserfall-Prozess! Aber da sie beim Reagieren auf Veränderungen so gut sind, fühlt es sich für die Teammitglieder gar nicht so an.

Punkt für Punkt

- Das **Agile Manifest** wurde 2001 von Menschen geschaffen, die zusammenkamen, um gemeinsame Grundlagen der verschiedenen »schlanken« Methoden, Vorgehensweisen und Ansätze zum Bauen von Software zu finden.
- Das Agile Manifest enthält **vier Werte**, die agilen Teams dabei helfen, das richtige Mindset aufzubauen.
- Agilen Teams sind **Prozesse und Werkzeuge wichtig**, weil sie dem Team dabei helfen, organisiert und effektiv zu arbeiten.
- Aber **Personen und Interaktion sind ihnen *wichtiger***, weil Teams dann am besten arbeiten, wenn auch der menschlichen Komponente Aufmerksamkeit geschenkt wird.
- Agilen Teams ist eine **umfassende Dokumentation wichtig**, weil es sich dabei um einen effektiven Weg handelt, komplexe Anforderungen und Ideen zu kommunizieren.
- Aber **funktionierende Software ist ihnen *wichtiger***, weil dies der effektivste Weg ist, Fortschritt zu kommunizieren und Feedback von Benutzern zu erhalten.
- Agilen Teams sind **Vertragsverhandlungen wichtig**, weil das manchmal der einzige Weg ist, in einer Firmenkultur zu arbeiten, in der Fehler bestraft werden.
- Aber **Zusammenarbeit mit dem Kunden ist ihnen wichtiger**, weil dies für das Bauen von Software viel effektiver ist als eine formale oder feindliche Kundenbeziehung.
- Agilen Teams **ist das Befolgen eines Plans wichtig**, weil komplexe Softwareprojekte ohne Planung aus der Spur geraten.
- Aber das **Reagieren auf Veränderung ist ihnen wichtiger**, weil ein Team, das dem falschen Plan folgt, auch die falsche Software bauen wird.

Manifest-Magneten



Hoppla! Eigentlich hatten Sie doch das Agile Manifest mithilfe von Kühlschrankmagneten perfekt zusammengebaut! Aber irgendwer hat die Tür zu kräftig zugeworfen, und sie sind heruntergefallen. Können Sie wieder alles zusammenbringen? Versuchen Sie es, möglichst ohne zurückzublättern.

Ein paar der Magneten sind hängen geblieben. Lassen Sie sie dort einfach kleben.

Manifest für Agile Softwareentwicklung

Wir erschließen bessere Wege, Software zu entwickeln,

indem wir es selbst tun und anderen dabei helfen.

Durch diese Tätigkeit haben wir diese

:

Machen Sie sich keine Gedanken um die Reihenfolge der Werte.

Die vier Werte im Agilen Manifest sind gleich wichtig, daher gibt es keine bestimmte Reihenfolge. Achten Sie einfach darauf, dass jeder Wert richtig zusammengesetzt ist (*X* mehr als *Y*).

mehr als

mehr als

mehr als

mehr als

Alle anderen Magneten sind abgefallen. Können Sie sie wieder richtig anbringen?

Das heißt, obwohl wir die | Werte | auf | der | Seite wichtig finden | ,

schätzen wir | die Werte auf der | Seite höher ein | .

Werte zu schätzen gelernt:

das Befolgen | Individuen | Veränderung | linken | Kunden | Werkzeuge | Plans

mit dem | Vertragsverhandlung

Prozesse | Funktionierende | und | und | auf | umfassende

Dokumentation | Software

rechten | Zusammenarbeit | eines

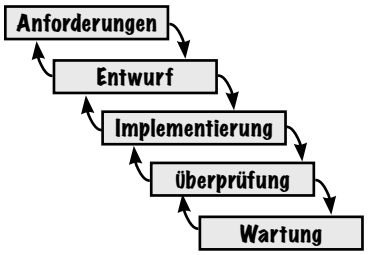
Interaktion | Reagieren

Antworten auf Seite 66

Es gibt keine Dummen Fragen

F: Mir ist immer noch nicht klar, was »Wasserfall« bedeutet.

A: »Wasserfall« ist der Name einer bestimmten Vorgehensweise, nach der Softwarefirmen traditionell Software gebaut haben. Sie haben ihre Projekte in Phasen unterteilt, die meist wie folgt in einem Diagramm abgebildet wurden:



Dieses Modell erhielt den Namen »Wasserfall« in den 1970ern von einem Forscher im Bereich der Softwareentwicklung, der es erstmals als einen weniger effektiven Weg zum Bauen von Software beschrieb. Es wird vom Team im Allgemeinen erwartet, dass es ein nahezu perfektes Anforderungsdokument und einen Entwurf liefert, bevor es mit dem Programmieren beginnt, weil es viel Zeit kostet, Anforderungen und/oder Design wieder zu ändern, wenn das Team darin Probleme erkennt.

Der Haken daran: Es gibt häufig keine Möglichkeit, vorherzusehen, ob die Anforderungen und der Entwurf richtig sind, bevor das Team nicht mit dem Programmieren angefangen hat. Bei einem Wasserfall-Team wird gerne davon ausgegangen, dass in der Dokumentation alles richtig ist, bis beim Coden des Designs ernsthafte Probleme auftauchen.

F: Warum sollte dann überhaupt jemand einen Wasserfall-Prozess genutzt haben?

A: Weil er funktioniert – oder zumindest funktionieren *kann*. Es gibt viele Teams, die einen Wasserfall-Prozess genutzt und großartige Software gebaut haben. Und natürlich ist es möglich, die Anforderungen und Entwürfe vorab so zu erstellen, dass es nur recht wenige Änderungen gibt.

Wichtiger noch: Es gibt viele Firmen, deren Kultur einen Wasserfall-Prozess fördert. Arbeiten Sie zum Beispiel für einen Chef, der Sie empfindlich bestraft, wenn er glaubt, dass Sie einen Fehler gemacht haben, kann es Ihnen helfen, Ihren Job zu behalten, wenn Sie sich von ihm persönlich die vollständig ausgearbeiteten Anforderungs- und Designdokumente genehmigen lassen, bevor Sie den Code schreiben. Aber egal wie Sie es machen – die Zeit, die erforderlich ist, um herauszufinden, wer für jede Entscheidung im Projekt verantwortlich ist, geht von der verfügbaren Zeit zum Umsetzen des Produkts ab.

F: Ist der Wasserfall nun gut oder schlecht? Und warum ist er weniger »schlank« als ein agiles Vorgehen oder ein Framework?

A: Der Wasserfall ist nicht »gut« oder »schlecht« – es handelt sich nur um einen bestimmten Weg, Dinge zu tun. Wie jedes Werkzeug hat er seine Stärken und Schwächen.

Aber viele Teams sind mit agilen Methoden wie Scrum erfolgreicher als mit einem Wasserfall-Prozess. Ein Grund dafür ist, dass sie einen Wasserfall-Prozess zu »schwerfällig« finden, weil er ihre Arbeit mit vielen Beschränkungen versieht: Sie müssen die kompletten Anforderungs- und Designphasen durchlaufen, bevor auch nur eine Zeile Code geschrieben werden kann. Im nächsten Kapitel werden Sie erfahren, wie Scrum-Teams ihre Sprints und Planungspraktiken nutzen, um schnell mit dem Bauen der Software zu beginnen, womit sie ihren Anwendern auch schneller funktionierende Software in die Hände geben können. Das fühlt sich für das Team viel »schlanker« an, weil alles, was sie tun, eine direkte Auswirkung auf den zu bauenden Code hat.

F: Wie sollte ich meine Projekte denn nun genau durchführen? Sollte mein Team eine Dokumentation erstellen oder nicht? Arbeiten wir mit einer vollständigen Spezifikation? Sollten wir die Dokumentation ganz wegschmeißen?

A: Dokumentation ist für agile Teams wichtig – aber vor allem, weil sie eine effektive Möglichkeit zum Bauen funktionierender Software sein kann. Dokumentation ist nur nützlich, wenn die Leute sie lesen, und die Wahrheit ist doch, dass viele sie einfach nicht lesen.

Wenn ich ein Anforderungsdokument schreibe und es Ihnen und Ihrem Team gebe, damit dieses den Code baut, ist das Dokument nicht wichtig. Wichtig ist, dass das, **was in meinem Kopf ist, zu dem passt, was in Ihrem Kopf ist** – und in den Köpfen aller anderen Teammitglieder. In manchen Fällen – zum Beispiel bei komplexen Berechnungen oder Abläufen – *kann ein Dokument eine sehr effektive Möglichkeit sein*, das **gemeinsame Verständnis zu schaffen**, das zu großartiger Software führt.

F: Ich bin immer noch nicht von der Idee überzeugt, dass Werte wichtig sind. Wie helfen sie mir und meinem Team dabei, tatsächlich Code zu schreiben?

A: Die Werte im Agilen Manifest helfen Ihnen und Ihrem Team dabei, eine Denkweise zu entwickeln, die Sie darin unterstützt, bessere Software zu bauen. Und Sie haben schon gelernt, dass Ihr Mindset – Ihr Verhalten bezüglich der eingesetzten Praktiken – einen großen Unterschied bedeuten kann.

Denken Sie an das Beispiel aus dem letzten Kapitel, in dem Kate und Mike ihre Probleme mit dem Daily Standup hatten. Kate bekam nur mittelmäßige Ergebnisse, als sie es als Weg nutzte, dem Team ihre Pläne aufzuoktroyieren und Statusinformationen einzufordern. Aber als sich alle kooperativer verhielten, wurden auch die Ergebnisse besser. Daran sieht man, dass das Mindset einen großen Einfluss auf die Ergebnisse haben kann.

Frage-Sprechstunde: Die Frage »Which is the BEST«

Eine tolle Möglichkeit, sich auf die Prüfung vorzubereiten, ist, die verschiedenen Arten von Fragen kennenzulernen und dann zu versuchen, eigene zu erstellen. Jede dieser Frage-Sprechstunden wird sich eine andere Art von Frage anschauen und Ihnen Tipps für das Schreiben eigener geben. Auch wenn Sie dieses Buch nicht nutzen, um sich auf die PMI-ACP-Zertifizierung vorzubereiten, setzen Sie sich ruhig einmal damit auseinander, denn es hilft Ihnen dabei, diese Konzepte in Ihren Kopf zu bekommen!

Da die Prüfung in englischer Sprache abgelegt wird, finden Sie in diesem Buch die Fragen und Antworten auf Englisch. Die Erläuterungen sind ins Deutsche übersetzt.

Bei vielen Prüfungsfragen müssen Sie sich entscheiden, welche der Antworten die **BESTE** ist. Das heißt im Allgemeinen, dass eine Antwort schon **ziemlich gut** ist, es aber eine andere gibt, die **besser** ist.

Es ist keine schlechte Idee, das Senior-Management einzubeziehen, aber das sollten Sie nur machen, wenn es um das Lösen ernsthafter Konflikte geht. Es ist für das Team besser, mit dem Anwender zusammenzuarbeiten und gemeinsam zu einer Lösung zu kommen, ohne auf Vorgesetzte zurückgreifen zu müssen.

82. A user asks the team for a new feature that's very important, but doing the work will make them miss a committed deadline for a different feature that they plan to work on. Which is the BEST thing the team should do first?

A. Call a meeting with senior management to get an official decision on relative priority

B. Follow the existing plan so they don't miss the deadline, and prioritize the new feature so they work on it next

C. Talk to to the user and figure out if the new feature is important enough for them to change direction

D. Initiate the change control process

Agilen Teams ist es wichtig, einem Plan zu folgen, daher ist das eine gute Idee ... aber noch wichtiger ist es, auf Änderungen zu reagieren. Gibt es vielleicht eine andere Antwort, die mehr den agilen Werten entspricht?

Das ist die BESTE Antwort! Agilen Teams ist es wichtiger, auf Veränderungen zu reagieren, als sich strikt an einen Plan zu halten. Daher ist es am besten, zügig beim Anwender rückzufragen, um alle notwendigen Informationen zu erhalten. Dann können alle zusammen überlegen, inwieweit der Plan betroffen ist.

Wenn Sie sich auf die PMP-Prüfung vorbereiten, könnte dies die richtige Antwort sein, da nämlich viele agile Teams in Firmen mit einem Change Control Process arbeiten. Aber in der Frage ging es darum, was das Team als Erstes machen sollte, und das hier würde erst später drankommen (oder auch gar nicht!).

Die »Which is the BEST«-Frage bietet mehr als eine gute Antwort, aber nur eine **BESTE**.

KOPF NUSS

Füllen Sie die freien Bereiche aus, um eine eigene »Which is the BEST«-Frage dazu zu basteln, wie agile Teams die Zusammenarbeit mit dem Kunden stärker wertschätzen können als Vertragsverhandlungen.

Sie sind Entwickler in einem _____ -Projekt. Ein _____
(eine Branche) (ein Anwendertyp)

möchte, dass Sie _____, aber Sie müssen _____ .
(etwas, das Sie für den Anwender tun sollen) (etwas anderes, das Sie tun müssten)

Welcher Weg ist der BESTE, um diese Situation zu meistern?

A. _____
(eine offensichtlich falsche Antwort, die gar nichts mit dieser Frage zu tun hat)

B. _____
(eine gute Antwort, die eine gute Idee ist, aber für diesen Wert keine Relevanz hat)

C. _____
(eine bessere Antwort, die zum Wertschätzen von Vertragsverhandlungen passt)

D. _____
(die BESTE Antwort, die dazu passt, dass die Zusammenarbeit mit Kunden wichtig ist)

LADIES UND GENTLEMEN, WIR KEHREN NUN ZU KAPITEL 2 ZURÜCK.

Sie glauben, einen Volltreffer gelandet zu haben ...

Mike hat jetzt zusammen mit seinem Entwicklungsteam seit fast einem Jahr am neuesten Killer-Feature gearbeitet, und er freut sich wirklich darüber, es endlich fertig zu haben.

… aber es ist ein Flop!

Hoppla. Es sieht so aus, als hätten Mike und sein Team ein Jahr für ein Produkt verschwendet, das keiner haben will. Was ist passiert?

Es wäre super gewesen, wenn wir es **vor einem Jahr** ausgeliefert hätten. Aber jetzt nutzt uns das nichts mehr!

Mike: Was? Wir haben ein ganzes Jahr lang daran gearbeitet. Willst du mir damit sagen, wir hätten nur unsere Zeit verschwendet?

Ben: Ich weiß nicht, was du mit deiner Zeit angestellt hast. Aber jetzt sage ich dir, dass ich mir ***keinen*** unserer Kunden damit vorstellen kann.

Mike: Aber was ist mit der großen Präsentation, die wir letztes Jahr auf der Konferenz gezeigt haben? Jeder Kunde, mit dem wir gesprochen hatten, war ganz begeistert von der Idee, genau zu wissen, wer seine Zuhörer sind, und sie direkt ansprechen zu können.

Ben: Stimmt. Und vor neun Monaten wurden drei unserer Kunden wegen Verletzung des Datenschutzes angeklagt. Jetzt wird sich da keiner mehr heranwagen.

Mike: Aber … aber das ist eine riesige Innovation! Du kannst dir gar nicht vorstellen, wie viele technische Probleme wir lösen mussten. Wir haben sogar extra eine spezialisierte KI-Beratungsfirma ins Boot geholt, die uns dabei half, die ausgefeilte Kundenanalyse umzusetzen!

Ben: Schau mal, Mike. Ich weiß nicht, was ich dir sagen soll. Vielleicht kannst du den Code für etwas anderes einsetzen?

Mike: Wir müssen sehen, dass wir retten, was möglich ist. Aber ich kann jetzt schon sagen: Immer wenn wir Codeteile herausnehmen, werden wir Fehler bekommen.

Ben: Echt jetzt? Ich wünschte, du hättest früher darüber mit mir gesprochen.

Das ist nicht abwegig! Eine Hauptfehlerquelle ist die Überarbeitung oder das Anfassen von Code, der schon gebaut wurde, um ihn einem anderen Zweck zuzuführen.

KOPF-NUSS

Wenn agilen Teams Änderungen wichtig sind, Veränderungen aber oft zu Überarbeitungen führen – wie können sie dann dafür sorgen, dass dadurch nicht dauernd neue Fehler entstehen?

Die Prinzipien hinter dem Agilen Manifest

Die vier Werte im Agilen Manifest verdeutlichen wunderbar den Kern des agilen Mindset. Sie vermitteln Ihnen ein sehr gutes Grundverständnis des »agilen Denkens« – dennoch muss ein Softwareteam tagtäglich auch jede Menge Entscheidungen treffen. Daher gibt es neben den vier Werten auch noch **zwölf Prinzipien hinter dem Agilen Manifest**, die Ihnen helfen sollen, das agile Mindset *wirklich* zu verstehen.

Manifest für Agile Softwareentwicklung

Wir erschließen bessere Wege, Software zu entwickeln,
indem wir es selbst tun und anderen dabei helfen.
Durch diese Tätigkeit haben wir diese Werte zu schätzen gelernt:

Individuen und Interaktion mehr als Prozesse und Werkzeuge
Funktionierende Software mehr als umfassende Dokumentation
Zusammenarbeit mit dem Kunden mehr als Vertragsverhandlung
Reagieren auf Veränderung mehr als das Befolgen des Plans

Das heißt, obwohl wir die Werte auf der rechten Seite wichtig finden,
schätzen wir die Werte auf der linken Seite höher ein.

???

```
public object Convert
    (object value, Type targetType,
                object parameter, string language)
{
  double parsedValue;
  if ((value != null)
      && double.TryParse(value.ToString(),
                         out parsedValue)
      && (parameter != null))
      switch (parameter.ToString()) {
          case "Hours":
              return parsedValue * 30;
              case "Minutes":
              case "Seconds":
                  return parsedValue * 6;
          }
      return 0;
}
```

Wenn Sie Software bauen, ist es nicht so leicht, die direkte Verbindung zwischen den agilen Werten und der täglichen Arbeit zu erkennen. Hier kommen die Prinzipien hinter dem Agilen Manifest ins Spiel.

Hinter den Kulissen

Die Gruppe konnte in Snowbird schnell die vier Werte beschreiben, aber es dauerte dann ein paar Tage mit langen Diskussionen, sich auf die zwölf Prinzipien hinter dem Agilen Manifest zu einigen – und selbst als sie Utah verließen, war der Text noch nicht fertig. Die erste vorgestellte Version sieht ein bisschen anders aus als die auf der nächsten Seite dargestellte finale Version (Sie finden sie auch auf *http://www.agilemanifesto.org*, der offiziellen Website des Agilen Manifests). Aber auch wenn sich der Text in den ersten paar Jahren noch ein wenig geändert hat – die Ideen, die in den zwölf Prinzipien stecken, sind gleich geblieben.

Prinzipien hinter dem Agilen Manifest

Wir folgen diesen Prinzipien:

- Unsere höchste Priorität ist es, den Kunden durch frühe und kontinuierliche Auslieferung wertvoller Software zufriedenzustellen.
- Heißen Sie Anforderungsänderungen selbst spät in der Entwicklung willkommen. Agile Prozesse nutzen Veränderungen zum Wettbewerbsvorteil des Kunden.
- Liefern Sie funktionierende Software regelmäßig innerhalb weniger Wochen oder Monate aus und bevorzugen Sie dabei die kürzere Zeitspanne.
- Fachexperten und Entwickler müssen während des Projekts täglich zusammenarbeiten.
- Errichten Sie Projekte rund um motivierte Individuen. Geben Sie ihnen das Umfeld und die Unterstützung, die sie benötigen, und vertrauen Sie darauf, dass sie die Aufgabe erledigen.
- Die effizienteste und effektivste Methode, Informationen an ein Entwicklungsteam und auch innerhalb des Teams zu übermitteln, ist das Gespräch von Angesicht zu Angesicht.
- Funktionierende Software ist der wichtigste Maßstab des Fortschritts.
- Agile Prozesse fördern nachhaltige Entwicklung. Die Auftraggeber, Entwickler und Benutzer sollten ein gleichmäßiges Tempo auf unbestimmte Zeit halten können.
- Ein ständiges Augenmerk auf technische Exzellenz und gutes Design fördert Agilität.
- Einfachheit – die Kunst, die Menge nicht getaner Arbeit zu maximieren – ist essenziell.
- Die besten Architekturen, Anforderungen und Entwürfe entstehen durch selbstorganisierte Teams.
- In regelmäßigen Abständen reflektiert das Team, wie es effektiver werden kann, und passt sein Verhalten entsprechend an.

Die agilen Prinzipien helfen Ihnen dabei, Ihr Produkt auszuliefern

Die ersten drei Prinzipien drehen sich darum, Software an Ihre Anwender auszuliefern. Und der effektivste Weg für das Ausliefern der bestmöglichen Software ist, sicherzustellen, dass sie **wertvoll** ist. Aber was bedeutet »Wert«? Wie stellen wir also sicher, dass wir beim beim Bauen von Software die Interessen unserer Nutzer, Stakeholder und Kunden berücksichtigen? Diese Prinzipien helfen uns beim Verständnis.

Ein Product Owner wie Ben arbeitet mit Anwendern und Kunden zusammen, um wirklich zu verstehen, was sie bei der Software benötigen. Er kann meist erkennen, dass ein Feature gar nicht verwendet wird … was häufiger vorkommt, als Sie vielleicht glauben!

- Unsere höchste Priorität ist es, den Kunden durch frühe und kontinuierliche Auslieferung wertvoller Software zufriedenzustellen.

Was heißt das jetzt genau? Es heißt, dass frühe Auslieferung und kontinuierliche Auslieferung für zufriedene Anwender sorgen:

Frühe Auslieferung

Die erste Version der Software sollte so früh wie möglich an die Benutzer gehen, um auch frühzeitig Feedback zu erhalten

Kontinuierliche Auslieferung

\+ Aktualisierte Versionen gehen immer wieder an die Anwender, sodass diese dem Team dabei helfen können, Software zu bauen, die ihre wichtigsten Probleme löst

---

Zufriedene Anwender

= Die Anwender helfen dem Team dabei, auf dem richtigen Weg zu bleiben, indem sie dafür sorgen, dass die wichtigsten Features als Erstes eingebaut werden

So verhalten sich viele Teams, wenn sich jemand eine größere Änderung am Code wünscht. Das ist verständlich, weil eine Änderung, die frühzeitiger hätte bekannt sein können, viel Arbeit verursacht, die langwierig und nervig sein kann.

- Heißen Sie Anforderungsänderungen selbst spät in der Entwicklung willkommen. Agile Prozesse nutzen Veränderungen zum Wettbewerbsvorteil des Kunden.

Wie reagiert das Team, wenn jemand darauf hinweist, dass es eine Änderung vornehmen muss, die viel Code betrifft? Jeder Entwickler kennt das, und es kann viel (oft schwierige) Arbeit bedeuten. Wie reagiert das Team nun? Es ist ganz natürlich, sich einer großen Änderung zu widersetzen. Aber wenn das Team einen Weg finden kann, diese Änderung nicht nur zu akzeptieren, sondern **willkommen zu heißen**, sorgt das dafür, dass ihm die ***langfristigen Anforderungen der Benutzer wichtiger sind als der eigene kurzfristiger Ärger***.

>>Anforderungen<< heißt schlicht, was die Software tun soll ... Aber manchmal ändern sich die Bedürfnisse der Anwender, oder die Programmierer haben missverstanden, was gebraucht wird, und das kann zu sich ändernden Anforderungen führen.

Wenn das Team Software früh und häufig an die Anwender, Stakeholder und Kunden ausliefert, haben alle die Möglichkeit, notwendige Änderungen frühzeitig zu erkennen, wenn sie sich noch leicht umsetzen lassen.

- Liefern Sie funktionierende Software regelmäßig innerhalb weniger Wochen oder Monate aus und bevorzugen Sie dabei die kürzere Zeitspanne.
- Funktionierende Software ist der wichtigste Maßstab des Fortschritts.

Wenn sich Entwickler gegen Änderungen wehren, ist das keine irrationale Reaktion: Haben sie viele Monate mit der Arbeit an einem Feature verbracht, kann es sich beim Ändern dieses Codes um einen langsamen, schmerzhaften und fehlerbehafteten Prozess handeln. Ein Grund dafür ist, dass beim **Überarbeiten** durch das Team (wenn also bestehender Code für etwas Neues angepasst wird) fast immer Fehler auftreten – gerne solche, die sich nur ausgesprochen schwer finden und beheben lassen.

Wie vermeidet das Team dann ein Überarbeiten? **Liefern Sie funktionierende Software regelmäßig aus**. Baut das Team ein Feature, das nicht nützlich ist oder gar nicht gebraucht wird, werden die Anwender das frühzeitig erkennen, und das Team kann die Änderung vornehmen, bevor zu viel Code geschrieben wurde … und durch das Vermeiden von Überarbeitung werden dann auch Bugs vermieden.

Das klingt ja **auf dem Papier** alles gut und schön, aber ich sehe noch nicht, wie diese Prinzipien in der Realität den Unterschied ausmachen können.

Prinzipien sind in der Praxis am sinnvollsten.

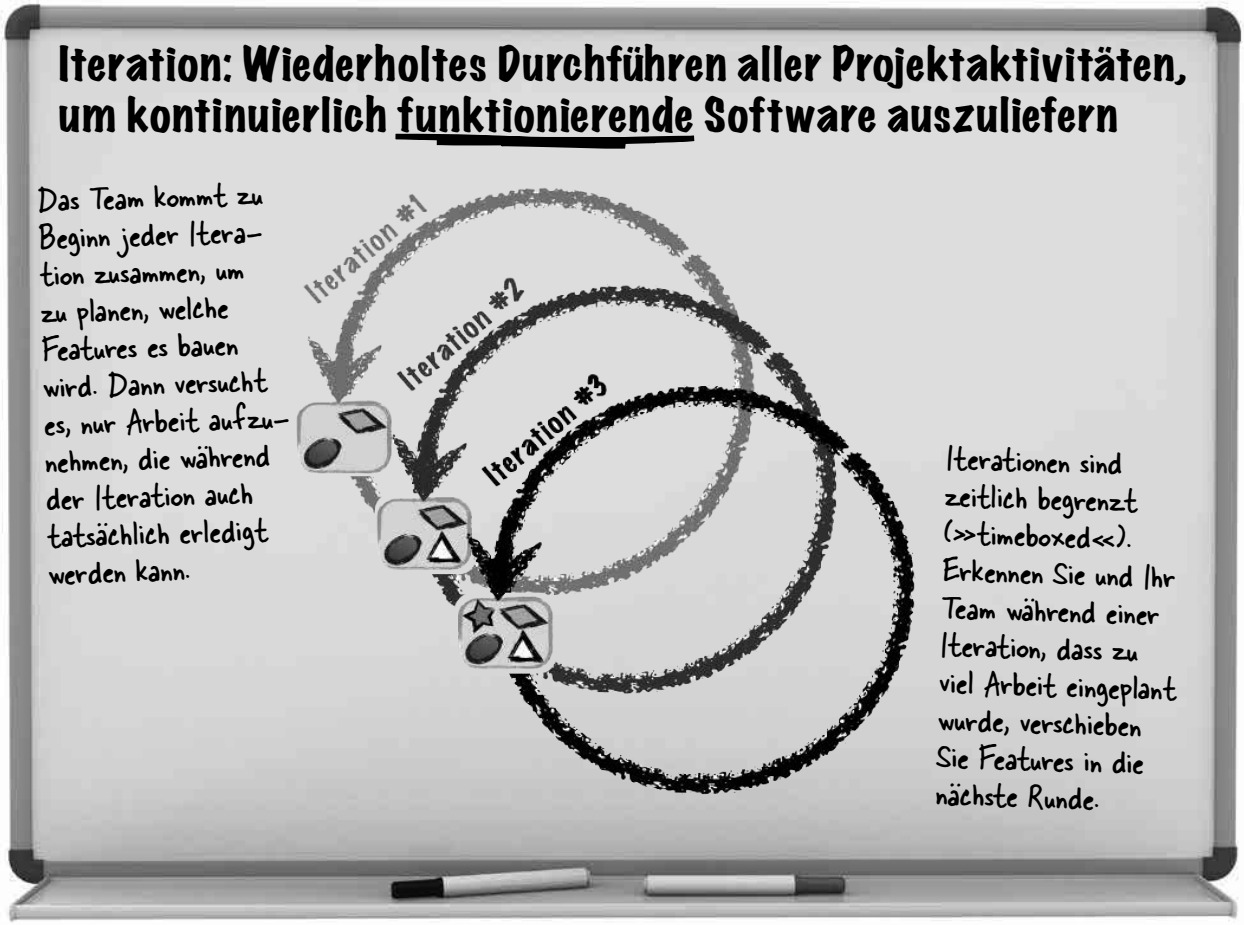
Viele von uns waren schon in Teams, die irgendwann einmal ins Straucheln kamen. Meist wurden dann neue Praktiken eingeführt. Aber manche Praktiken, die für die einen Teams richtig gut funktionieren, sorgen bei anderen Teams nur für winzige Verbesserungen – so wie wir das mit dem Daily Standup in Kapitel 1 gesehen haben.

Was macht also den Unterschied zwischen den Teams aus, die nur so mittelprächtige Ergebnisse erhalten, und solchen, bei denen es großartig funktioniert? Meist hat es viel mit dem *Mindset des Teams* zu tun und ihrer Einstellung gegenüber der Praktik. Und darum geht es bei den Prinzipien: Sie helfen Teams dabei, die beste Denkweise zu finden, die ihre Praktiken so effektiv wie möglich machen.

Sie werden auf der nächsten Seite ein Beispiel dafür finden … ⟶

Prinzipien in der Praxis

In den ersten drei Prinzipien, die hinter dem Agilen Manifest stehen, geht es um frühes und kontinuierliches Ausliefern von Software, das Willkommenheißen von Änderungsanforderungen und das Ausliefern funktionierender Software in kurzen Zeitabständen. Aber wie machen Teams das in der Realität? Mit guten Praktiken, wie zum Beispiel **Iteration** und dem Einsatz eines **Backlogs**.



AUS DEM WÖRTERBUCH

timeboxed, Adjektiv

Eine feste Deadline für eine Aktivität setzen, bis zu der sie fertig sein soll, und den Scope dieser Aktivität anpassen, um die Deadline zu erfüllen.

Das Team konnte nicht alle angeforderten Features in der aktuellen ***timeboxed*** *Iteration umsetzen, daher hat es sich auf die wertvollsten konzentriert.*

Wenn Sie unser Buch *Learning Agile* gelesen haben, werden Sie diese Bilder zu Iteration und Backlog wiedererkennen!

He, Moment mal! Wir haben schon gelernt, dass Scrum-Teams ein Backlog nutzen. Heißt das, **Scrum-Sprints** sind eine Form von **Iteration**?

Genau! Scrum nutzt einen iterativen Ansatz.

Die Scrum-Praktik der Sprints ist ein klassisches Beispiel dafür, wie Teams Iteration im echten Leben einsetzen, um funktionierende Software früh und regelmäßig auszuliefern. Das Scrum-Team hat einen Product Owner, der mit den Anwendern und Stakeholdern zusammenarbeitet, um deren Bedürfnisse zu verstehen. Alle lernen mit jeder neuen Version der funktionierenden Software dazu, und der Product Owner nutzt dieses Wissen, um Features zum Backlog hinzuzufügen oder daraus zu entfernen.

Wir werden uns über Scrum noch mehr im nächsten Kapitel unterhalten.

Kamingespräche

Thema heute: **Praktik trifft auf Prinzipien**

Prinzip:

Ich habe mir diese Diskussion jetzt schon eine Weile angeschaut.

Das ist er wieder. Immer mit der gleichen Leier: »Nichts passiert ohne Praktiken.«

Ja? Okay. Dann reden wir doch mal über diese Praktiken. Wie zum Beispiel das Daily Scrum …

Genau! Und es ist nicht nur das Daily Scrum. Reden wir über Iteration.

Das ist sie. Aber was passiert, wenn die Leute im Team nicht wirklich und wahrhaftig an das Prinzip des regelmäßigen Auslieferns funktionierender Software glauben?

Schon. Aber wird das Team wirklich funktionierende Software ausliefern? Oder wird es einfach zu viel wegschneiden, nur um den Kunden irgendetwas vor die Tür zu werfen, bevor die Iteration endet? Wird es ein Feature wirklich in die nächste Iteration verschieben, weil die Zeit nicht mehr reicht? Oder wird das Hinzufügen von Iterationen zum Projekt jedem im Team das Gefühl geben, das einfach nur der Form halber zu tun?

Waren Sie schon mal in einem Team, das versucht hat, agil zu werden, aber letztendlich nur mittelmäßige Ergebnisse erzielt hat? Wenn das so ist, erinnert Sie das vielleicht an Ihre eigenen Erfahrungen.

Praktik:

Ich bin nicht sicher, ob das wirklich eine große Diskussion ist, wenn Sie meine Meinung hören wollen.

Na ja, Sie müssen zugeben, dass das ein ziemlich gutes Argument ist. Denn was wäre ein Ansatz wie Scrum schließlich ohne mich? Nehmen Sie die Sprints, Backlogs, Retrospektiven, Sprint Reviews, Daily Scrums und Sprint-Planungen weg. Was bleibt? Chaos!

Moment, ich weiß schon, was Sie als Nächstes sagen werden. Es geht doch darum, dass das Daily Scrum nur zu »mittelmäßigen« Ergebnissen führt, wenn das Team die Prinzipien nicht »verstanden« hat.

Eine fantastische Praktik, vielen Dank.

Es wird weiterhin Iterationen geben! Und wissen Sie was? Es wird trotzdem besser sein als vor dem Einsatz dieser Praktik.

Das stimmt! Auch wenn das Team die Prinzipien nicht wirklich versteht, führt das Hinzufügen von Iterationen normalerweise trotzdem zu einer Verbesserung – einer nicht sehr großen, aber doch lohnenswerten.

Na ja, zumindest wird es etwas zu zeigen haben. Selbst wenn es nur um eine kleine Verbesserung geht, ist das besser als nichts!

wort-wissen

Hier finden Sie ein paar Definitionen für Wörter, die Ihnen in diesem Kapitel über den Weg gelaufen sind. Können Sie sie den entsprechenden Definitionen zuordnen?

_____, Nomen

Vom Team geleistete Arbeit, um zuvor geschriebenen Code einem anderen Zweck zuzuführen oder anders arbeiten zu lassen, wird von Teams aufgrund der erhöhten Bug-Anfälligkeit oft als riskant angesehen.

_____, Adjektiv

Eine harte Deadline für das Fertigstellen einer Aktivität setzen und dann den Scope der Aktivität anpassen, damit die Deadline gehalten wird.

_____, Nomen

Eine von Teams genutzte Praktik, bei der das Team mit Anwendern, Kunden und/oder Stakeholdern zusammenarbeitet, um eine Liste mit zukünftigen Features zu erstellen – häufig so geordnet, dass das wertvollste Feature ganz oben steht.

_____, Adjektiv und Nomen

Ein Projekt-Artefakt (wie zum Beispiel einen Plan) schrittweise entwickeln und dabei das Wissen aus dem vorigen Schritt nutzen, um es zu verbessern.

_____, Adjektiv

Eine Art von Vorgehensweise, bei der Teams das Projekt in kleinere Einheiten unterteilen, am Ende jeder dieser Einheiten funktionierende Software ausliefern und, abhängig vom Feedback, eventuell die Richtung mit Blick auf funktionierende Software anpassen.

_____, Nomen

Ein Typ eines Modells, Prozesses oder einer Methode für das Bauen von Software, in der das gesamte Projekt in aufeinanderfolgende Phasen unterteilt wird, oft in Verbindung mit einem Prozess zum Änderungsmanagement, bei dem Änderungen ein Projekt in eine frühere Phase zurückwerfen.

→ Antworten auf Seite 67

Es gibt keine Dummen Fragen

F: **Gehört jedes Prinzip zu genau einer Praktik? Gibt es also eine Eins-zu-eins-Beziehung?**

A: Ganz und gar nicht. Die ersten drei Prinzipien hinter dem Agilen Manifest betonen das frühe und kontinuierliche Ausliefern von Software, das Willkommenheißen von sich ändernden Anforderungen und das regelmäßige Ausliefern funktionierender Software. Und wir haben zwei Praktiken (Iteration und Backlog) genutzt, um Ihnen dabei zu helfen, die Prinzipien auf einer tieferen Ebene zu verstehen. Aber das heißt nicht, dass es eine Eins-zu-eins-Beziehung zwischen den Praktiken und den Prinzipien gibt.

Tatsächlich ist das Gegenteil der Fall. Sie können Prinzipien ohne Praktiken und Praktiken ohne Prinzipien haben.

F: **Ich bin nicht sicher, wie das funktioniert. Was genau meinen Sie mit »Praktiken ohne Prinzipien«?**

A: Hier ein Beispiel dafür, wie es aussieht, wenn ein Team eine Praktik umsetzt, ohne die agilen Prinzipien wirklich zu verstehen oder sie zu verinnerlichen: Scrum-Teams halten am Ende jedes Sprints eine **Retrospektive** ab, sodass sie darüber sprechen können, was gut lief und was noch verbessert werden kann.

Aber schauen Sie noch einmal auf das letzte Prinzip in der Liste:

> In regelmäßigen Abständen reflektiert das Team, wie es effektiver werden kann, und passt sein Verhalten entsprechend an.

Was, wenn sich das Team diese Idee nicht wirklich zu Herzen genommen hat? Sie werden das Retrospektiven-Meeting abhalten, weil die Scrum-Regeln das so vorgeben. Und sie werden vermutlich über ein paar der aufgetretenen Probleme sprechen, was sicherlich auch zu kleinen Verbesserungen führen kann.

Das Problem liegt darin, dass das neue Meeting zwar irgendetwas bewirkt hat, es sich aber dennoch »leer« oder überflüssig anfühlt. Die Leute im Team haben das Gefühl, dass ihnen Zeit für die »richtigen« Aufgaben weggenommen wird, die sie zu erledigen haben. Und schließlich werden sie darüber reden, es durch etwas »Effizienteres« zu ersetzen, zum Beispiel eine E-Mail-Liste oder eine Wiki-Seite. Viele Teams haben diese Erfahrung gemacht, wenn sie Praktiken umsetzten, ohne die Prinzipien dahinter zu berücksichtigen.

F: **Okay, ich glaube, ich verstehe, wie Praktiken funktionieren, ohne dass wirklich Prinzipien dahinterstehen. Aber wie können Sie Prinzipien ohne Praktiken haben?**

A: Viele Menschen haben so ihre Zweifel, wenn sie das erste Mal mit der Idee eines »agilen Mindset« in Kontakt kommen, die von Prinzipien gesteuert wird.

Wie sieht es also aus, wenn das Team das agile Prinzip des Reflektierens über ein effektiveres Arbeiten sehr ernst nimmt, aber keine Praktik für das Umsetzen dieser Reflexion nutzt? Das ist in *sehr effektiven* agilen Teams tatsächlich stark verbreitet. Jeder hat die Haltung, häufig zu reflektieren, und wenn jemand das Gefühl hat, es sei an der Zeit, ein Review darüber abzuhalten, wie das Projekt vorankommt und was für Änderungen vorzunehmen sind, schnappt sich diese Person meist ein paar Kollegen und führt eine informelle Retrospektive durch. Wenn dabei etwas Gutes herauskommt, reden sie darüber und setzen diese Änderung um.

Ein Team, das sich an einem Framework mit sehr spezifischen Regeln, wie etwa Scrum, ausrichtet, empfindet das als unorganisiert, chaotisch oder auch extrem gechillt. Das ist ein Grund dafür, warum sich Teams gerne an einem standardisierten Satz von Praktiken orientieren – alle haben dadurch eine gemeinsame Regelgrundlage.

F: **Warum haben Sie unten auf Seite 47 eigentlich den Product Owner so hervorgehoben?**

A: Während viele Leute zwar den Jobtitel »Product Owner« tragen, bezieht sich diese Hervorhebung auf eine spezifische Rolle mit bestimmten Verantwortlichkeiten, die durch die Scrum-Regeln definiert sind. Sie werden darüber mehr im nächsten Kapitel erfahren.

Wenn das Team Praktiken umsetzt, ohne das dazu passende durch die Prinzipien gesteuerte Mindset zu besitzen, fühlt sich das oft »leer« oder überflüssig an – so als folgten sie lediglich den Regeln. Dann wird sich das Team Alternativen suchen, die weniger Aufwand erfordern.

Wow, Kate!
Ich fühle mich bei diesem Projekt jetzt viel wohler. Jedes Mal, wenn ich einen neuen Build bekomme, kann ich genau sehen, **welchen Fortschritt** das Team gemacht hat.

Es gibt aber immer noch etwas, das mich stört.

Ben: Gerade noch habe ich mich gut gefühlt. Ich wünschte, du wärst nicht so negativ. Welche schlechten Nachrichten hast du denn?

Kate: Ich versuche gar nicht, negativ zu sein. Ich freue mich wirklich über den Fortschritt, den wir alleine schon dem Einsatz der Iteration zu verdanken haben.

Ben: Genau! Ich bin mit den frühen Builds zurück zu den Anwendern gegangen, und sie haben alle möglichen Änderungen aufgezählt, die wir frühzeitig umsetzen könnten, ohne viel überarbeiten zu müssen.

Kate: Ja, und das ist großartig. Aber wir haben weiterhin Probleme.

Ben: Und die wären …?

Kate: Nun, zum Beispiel das Meeting letzten Mittwoch. Den ganzen Nachmittag haben wir über die Dokumentation gestritten.

Ben: Geht das schon wieder los? Du und Mike fragt die ganze Zeit nach Spezifikationen mit allen zu bauenden Details.

Kate: Ja, denn dann kann ich dem Team dabei helfen, das Projekt durchzuplanen, und Mike und das Team wissen ganz genau, was sie zu bauen haben.

Ben: Aber das ist nicht so einfach! Diese Specs lassen sich echt schwer schreiben. Und selbst wenn wir eine Spec für eine Iteration schreiben, wird sie ziemlich lang werden.

Kate: Gut, wenn du eine bessere Idee hast, wie wir das Team dazu kriegen können, die richtige Software zu bauen, dann nur her damit.

KOPF-NUSS

Viele Teams scheinen Probleme damit zu haben, sehr detaillierte Spezifikationen zu schreiben und zu lesen. Fällt Ihnen ein, wie ein Product Owner dem Team effektiver dabei helfen kann, genau zu verstehen, was die Anwender gebaut haben müssen?

Die agilen Prinzipien helfen Ihrem Team dabei, zu kommunizieren und zusammenzuarbeiten

Moderne Software wird von Teams gebaut, und obwohl die einzelnen Personen für jedes Team ausgesprochen wichtig sind, funktioniert es am besten, wenn alle zusammenarbeiten – was bedeutet, dass Entwickler nicht nur untereinander zusammenarbeiten, sondern auch mit den Anwendern, Kunden und Stakeholdern. Darum geht es bei diesen nächsten Prinzipien.

- Fachexperten und Entwickler müssen während des Projekts täglich zusammenarbeiten.
- Errichten Sie Projekte rund um motivierte Individuen. Geben Sie ihnen das Umfeld und die Unterstützung, die sie benötigen, und vertrauen Sie darauf, dass sie die Aufgabe erledigen.

Entwickler drücken sich gerne vor Meetings mit Anwendern, denn dabei kommen häufig Änderungen heraus, die zu Überarbeitungen führen können, die immer wieder schwierig und frustrierend sind. Aber wenn das Team ein besseres, agileres Mindset besitzt, weiß es, dass **Meetings mit Anwendern häufig** dafür sorgen, auf dem neuesten Stand zu sein, und damit solche Änderungen tatsächlich zu verhindern.

Ein agileres Mindset würde Mike dabei helfen, zu erkennen, dass die Zusammenarbeit mit seinen Anwendern solche Änderungen eher zu vermeiden hilft.

Sich mit den Anwendern **noch häufiger** treffen?! Aber die wollen doch immer nur Änderungen.

Teams funktionieren am besten, wenn die Leute darin **motiviert** sind. Leider hatten wir alle schon Chefs oder Kollegen, die irgendwie dazu bestimmt waren, all die schöne Motivation aus uns herauszusaugen. Wenn die Mitarbeiter das Gefühl haben, dass sie keine Fehler machen dürfen, ohne drastische Strafen zu befürchten, dazu genötigt werden, viele Überstunden zu machen, und ganz allgemein spüren, dass man ihnen beim Erledigen ihrer Arbeit nicht traut, geht es mit Quantität und Qualität ihrer Arbeit bergab. Teams mit einem agileren Mindset wissen, dass sie gedeihen können, wenn allen vertraut wird und die Arbeitsumgebung angenehm ist.

Es ist mir egal, ob das Team 70-Stunden-Wochen hat. Ein Fehlschlag steht nicht zur Diskussion, und **Fehler werden gemeldet**.

Das ist eine 1A-Methode, Ihr ganzes Team zu demotivieren und dafür zu sorgen, dass es schlechte Arbeit abliefert. Ben ist nicht einmal der Chef! Aber er kann trotzdem eine Atmosphäre der Angst und des Misstrauens um ihn herum erzeugen.

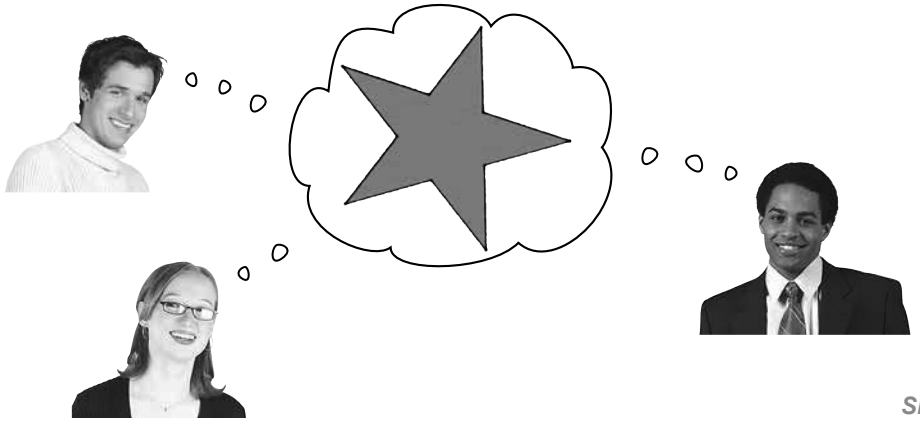
- Die effizienteste und effektivste Methode, Informationen an ein Entwicklungsteam und auch innerhalb des Teams zu übermitteln, ist das Gespräch von Angesicht zu Angesicht.

Wasserfall-Teams erstellen im Allgemeinen zuerst eine Spezifikation, und dann entwerfen sie die Software basierend auf diesen Anforderungen. Das Problem ist, dass drei Leute die gleiche Spec lesen und dann drei verschiedenen Vorstellungen davon existieren, was das Team zu bauen hat. Das klingt ein bisschen überraschend – sollten Spezifikationen nicht präzise genug sein, allen die gleichen Vorstellungen zu vermitteln?

In der Realität gibt es dabei zwei Probleme: Das Schreiben technischer Dokumente ist schwer und das Lesen ist noch schwerer. Auch wenn die Person, die die Spec schreibt, ihre Aufgabe perfekt erledigt und beschreibt, was gebaut werden muss (was nur selten geschieht), werden die Leute, die sie lesen, den Text sehr unterschiedlich interpretieren. Wie können Sie dieses Problem dann vermeiden?

Seien wir ehrlich – wir lesen nicht immer jedes Wort in der Anleitung, bevor wir ein neues Spielzeug in Betrieb nehmen. Warum sollten wir dann glauben, dass das die Leute mit den Specs machen?

Die Antwort ist überraschend einfach: **direkte Gespräche**. Kommt das Team zusammen und redet darüber, was es bauen muss, ist das *wirklich* der effizienteste und effektivste Weg, das zu kommunizieren, was gebaut werden muss … und dazu Status, Ideen und andere Informationen.



Es gibt keine Dummen Fragen

F: Wollen Sie behaupten, dass Leute absichtlich schlechte Arbeit abliefern, wenn sie demotiviert sind?

A: Nein, nicht absichtlich. Aber es ist sehr schwer, innovativ und kreativ zu sein oder die mental herausfordernden Aufgaben zu erledigen, die in einem Softwareteam erforderlich sind, wenn Sie sich in einer demotivierenden Umgebung befinden. Und es ist erstaunlich einfach, einem Team die Motivation zu nehmen: Sie sinkt, wenn Ihnen nicht zugetraut wird, Ihren Job zu erledigen, wenn Sie hart bestraft oder öffentlich gedemütigt werden, wenn Sie einen Fehler machen (jeder macht Fehler!) oder wenn es unhaltbare Deadlines gibt, über die Sie keinerlei Kontrolle oder auf die Sie keinen Einfluss haben – alles Dinge, die wiederholt dafür verantwortlich waren, Softwareteams herunterzuziehen und sie weniger produktiv zu machen.

F: Moment, noch mal zurück zu der Stelle mit dem Fehler. Wir haben darüber geredet, Änderungen willkommen zu heißen. Aber wenn Sie eine Änderung vornehmen, heißt das doch auch, dass jemand zuvor einen Fehler gemacht hat, der jetzt korrigiert werden muss, oder?

A: Es ist gefährlich, Änderungen als Fehler anzusehen, insbesondere wenn Sie Iteration nutzen. Häufig sind sich Teammitglieder, die Anwender und Stakeholder darin einig, dass die Software so gebaut werden soll, dass sie etwas Bestimmtes tut. Aber wenn die Anwender dann am Ende der Iterationsrunde damit arbeiten, erkennen sie, dass sie geändert werden muss – nicht, weil sie zuvor einen Fehler gemacht haben, sondern weil sie jetzt über Informationen verfügen, die sie zu Beginn der Iteration nicht besaßen. Das ist sogar ein sehr effektiver Weg, Software zu bauen. Aber er funktioniert nur, wenn die Leute kein Problem damit haben, Änderungen vorzunehmen, und wenn sie hier keinen Fehler sehen oder jemandem die »Schuld« daran geben, dass er die Notwendigkeit zur Änderung erkannt hat.

F: Brauchen wir Spezifikationen nicht für mehr als nur die Kommunikation? Was, wenn Sie sich später darauf beziehen müssen? Oder wenn sie an mehr als an ein paar Leute verteilt werden muss?

A: Sicher, und das sind auch gute Gründe dafür, Dinge aufzuschreiben. Darum schätzen agile Teams eine umfassende Dokumentation – aber funktionierende Software ist ihnen trotzdem wichtiger.

Bedenken Sie aber noch Folgendes: Wenn Sie eine Dokumentation schreiben, um sich darauf beziehen zu können oder um sie über das Softwareteam hinaus an eine größere Leserschaft zu verteilen, ist eine Softwarespezifikation dafür vielleicht nicht die richtige Art von Dokument. Die Dokumentation ist ein Werkzeug, um eine Aufgabe erledigen zu können, und Sie wollen immer das richtige Werkzeug für eine Aufgabe nutzen. Die Informationen, die das Team benötigt, um Software zu bauen, unterscheiden sich im Allgemeinen von denen, die ein Anwender oder Manager braucht, nachdem die Software gebaut wurde. Daher kann es sein, dass ein Dokument für beide Zwecke nicht sonderlich hilfreich ist.

F: So, das Kapitel ist fast zu Ende und Sie haben noch gar nicht alle zwölf Prinzipien behandelt. Warum nicht?

A: Weil die agilen Prinzipien kein isoliertes Thema sind, das Teams einmal behandeln und dann fortfahren. Sie sind wichtig, weil sie Ihnen dabei helfen, zu verstehen, wie agile Teams über ihre Arbeitsweise beim Bauen von Software denken. Darum sind die Werte und Prinzipien des Agilen Manifests so entscheidend.

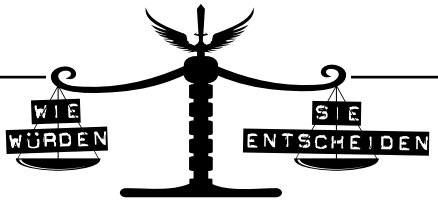
Wir werden das Thema des agilen Mindset oder der Prinzipien hier nicht beenden, auch wenn wir uns im nächsten Kapitel den Vorgehensweisen zuwenden. Da sie Ihnen das Verständnis für die Vorgehensweisen erleichtern (etwa dass Scrum-Teams sich selbst organisieren und dass XP-Teams Einfachheit wichtig ist), werden wir wieder auf sie zurückkommen.

Punkt für Punkt

- Software ist **wertvoll**, wenn sie das tut, was Anwender, Kunden und Stakeholder von ihr brauchen.
- Um sicherzustellen, dass Software wertvoll ist, sollten Teams eine **frühe** Version an die Benutzer liefern und dann weiter **kontinuierlich** ausliefern.
- Agile Teams **heißen Veränderungsanforderungen willkommen,** und das frühzeitige Finden solcher Änderungen hilft dabei, Überarbeitungen zu verhindern.
- Die beste Möglichkeit, solche Änderungen frühzeitig zu finden, ist, **regelmäßig funktionierende Software an die Anwender zu liefern.**
- Dokumente sind hilfreich, aber der *effektivste* Weg, Informationen zu verbreiten, sind **direkte Gespräche.**
- Entwickler arbeiten in agilen Teams **jeden Tag mit Fachexperten zusammen**, auch mit Anwendern und Stakeholdern.
- **Iteration** ist eine Praktik, bei der Teams die Software häufig und »timeboxed« ausliefern.
- Ein **Backlog** ist eine Praktik, bei der Teams eine Liste mit Features verwalten, die in zukünftigen Iterationen gebaut werden sollen.

Scrum-Teams verwalten tatsächlich zwei Backlogs: eines für den aktuellen Sprint und eines für das gesamte Produkt.

Darüber werden Sie mehr im nächsten Kapitel erfahren.



Es ist nicht immer leicht, sich ein agileres Mindset anzueignen! Manchmal klappt es problemlos, aber manchmal ist ein wenig Aufwand nötig. Hier sind ein paar Dinge, die wir Mike, Kate und Ben haben sagen hören. Ziehen Sie eine Linie von jeder Sprechblase hin zu KOMPATIBEL oder INKOMPATIBEL und dann zu dem agilen Prinzip, mit dem die Aussage entweder kompatibel oder inkompatibel ist.

Warum fragt ihr mich etwas? Ich habe doch schon alles, was die Anwender haben wollen, in der Spezifikation aufgeschrieben.

KOMPATIBEL

INKOMPATIBEL

Funktionierende Software ist der wichtigste Maßstab des Fortschritts.

Ich habe gerade entdeckt, dass unser Algorithmus zum Berechnen der Zuhöreranzahl nicht funktioniert. Wir müssen dieses Feature in die nächste Iteration verschieben.

KOMPATIBEL

INKOMPATIBEL

Heißen Sie Anforderungsänderungen selbst spät in der Entwicklung willkommen. Agile Prozesse nutzen Veränderungen zum Wettbewerbsvorteil des Kunden.

Okay, wer von euch Idioten hat diesen fehlerstrotzenden Spaghetti-Code geschrieben? Es ist dein Fehler, dass wir nicht im Zeitplan liegen.

KOMPATIBEL

INKOMPATIBEL

Liefern Sie funktionierende Software regelmäßig innerhalb weniger Wochen oder Monate aus und bevorzugen Sie dabei die kürzere Zeitspanne.

Ich lasse den letzten Build laufen, aber ich dachte, wir wären mit dem Analytics-Feature schon weiter. Gibt es ein Problem, von dem ich nichts weiß?

KOMPATIBEL

INKOMPATIBEL

Errichten Sie Projekte rund um motivierte Individuen. Geben Sie ihnen das Umfeld und die Unterstützung, die sie benötigen, und vertrauen Sie darauf, dass sie die Aufgabe erledigen.

Antworten auf Seite 68

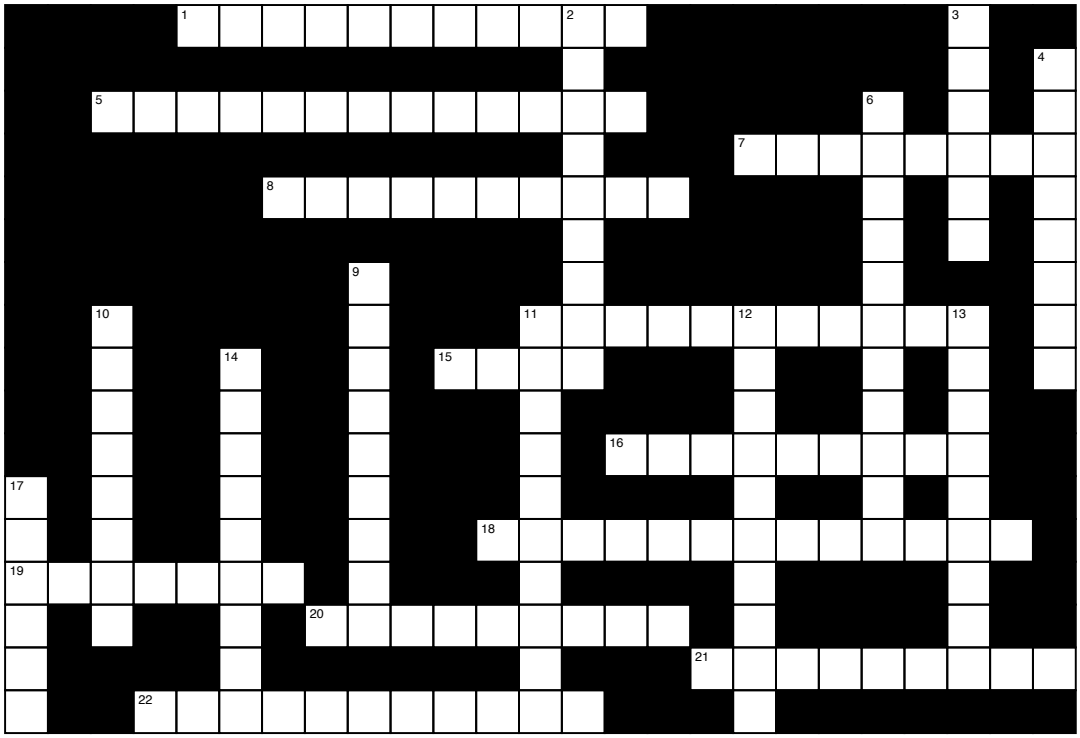
Das neue Produkt ist ein Hit!

Kate und Mike haben ein großartiges Produkt ausgeliefert, das außerordentlich erfolgreich ist.

Tatsächlich lief es so gut, dass Ben fantastische Neuigkeiten für alle hat. Gute Arbeit, Team!

Mindset-Kreuzworträtsel

Wie gut haben Sie die agilen Werte und Prinzipien verstanden? Können Sie dieses Kreuzworträtsel lösen, ohne im Kapitel zurückzublättern?



Waagerecht

1. Wann Software geliefert werden soll.
5. Das tut Software am Ende jeder Iteration.
7. Was Business-Mitarbeiter und Entwickler tun müssen.
8. Klassischer, aber oft weniger effektiver Weg, Software zu bauen.
11. Funktionierende Software ist die beste Messmethode für _____.
15. Auch agile Teams folgen einem _____.
16. _____ legen genau fest, was zwischen Team und Kunden vereinbart ist.
18. Ein effektiver Weg, um komplexe Anforderungen und Ideen zu kommunizieren.
19. Wie oft wird geliefert?
20. Sie müssen dem Team _____ damit die Arbeit erledigt wird.
21. Die höchste Priorität sind Kunden, die _____ sind.
22. In regelmäßigen _____ reflektiert das Team, wie es effektiver werden kann.

Senkrecht

2. Wenn Teams all ihre Projekt-Aktivitäten in kleinen Einheiten wiederholen.
3. Es gibt nicht den einen »_____« Weg, Software zu bauen-
4. Teams arbeiten am besten, wenn man den _____ Aufmerksamkeit widmet.
6. Ihr Team wird _____, wenn Sie eine Kultur der Angst aufbauen.
9. Sehr nützlich für agile Teams, weil sie dabei helfen, die Arbeit zu erledigen.
10. Was ein Team nach einer Retrospektive mit seinem Verhalten tut.
11. Die effektivste und effizienteste Methode, Informationen einzuholen (ohne Leerzeichen angeben).
12. Agile Teams versuchen, _____ Delivery zu erreichen.
13. Wenn eine Deadline fest gesetzt ist und der Scope daran angepasst wird.
14. Die Leute müssen _____ sein, damit Projekte funktionieren.
17. Etwas, für das ein motiviertes Team nicht bestraft werden sollte.

→ Antworten auf Seite 69

Prüfungsfragen

Diese Übungsprüfungsfragen werden Ihnen dabei helfen, das Material in diesem Kapitel Revue passieren zu lassen. Versuchen Sie, sie auch dann zu beantworten, wenn Sie sich mit diesem Buch nicht auf die PMI-ACP-Zertifizierung (englischsprachig) vorbereiten wollen. So finden Sie schnell heraus, was Sie wissen – und was nicht. Die Informationen gelangen dadurch schneller in Ihren Kopf.

1. You're a project manager on a team building network firmware for embedded systems. You've called a meeting to give a demo of the latest version of code the team has been working on for a control panel interface to a very technical group of business users and customers. This is the fifth time that you've called a meeting to do a demo like this. And for the fifth time, the users and customers asked for specific changes. The team will now go back and work on a sixth version, and you'll repeat the process again.

Which of the following BEST describes this situation?

A. The team does not understand the requirements
B. The users and customers don't know what they want
C. The project needs better change control and requirements management practices
D. The team is delivering value early and continuously

2. Which of the following is NOT a Scrum role?

A. Scrum Master
B. Team Member
C. Project Manager
D. Product Owner

3. Joaquin is a developer, and his software team is in the process of adopting agile. One of the project's users wrote a brief specification that describes exactly what she wants for a new feature, and Joaquin's manager assigned him to work on that feature. What should Joaquin do next?

A. Demand a meeting with the user, because agile teams recognize that face-to-face conversation is the most efficient and effective method of conveying information
B. Read the specification
C. Ignore the specification, because agile teams value customer collaboration over comprehensive documentation
D. Start writing code immediately, because the team's highest priority is to satisfy the customer through early delivery of valuable software

4. Which of the following is TRUE about working software?

A. It does what the users need it to do
B. It meets the requirements in its specification
C. Both A and B
D. Neither A nor B

Prüfungsfragen

5. Which of the following statements BEST describes the Agile Manifesto?

A. It outlines the most effective way to build software
B. It contains practices that many agile teams use
C. It contains values that establish an agile mindset
D. It defines rules for building software

6. Scrum projects are divided into:

A. Phases
B. Sprints
C. Milestones
D. Rolling wave planning

7. You are a developer at a social media company working on a project to build a new feature to create a private site for a corporate client. You need to work with your company's network engineers to determine a hosting strategy, and come up with a set of services and tools that the engineers will use to manage the site. The network engineers want to host all of the services internally on your network, but you and your teammates disagree and feel that the services should be hosted on the client's network. Work on the project has come to a halt while everyone tries to come to an agreement. Which agile value BEST applies to this situation?

A. Individuals and interactions over processes and tools
B. Working software over comprehensive documentation
C. Customer collaboration over contract negotiation
D. Responding to change over following a plan

8. Donald is a project manager on a team that follows separate phases for each project, starting with a requirements phase followed by a design phase. Some work can begin on the code before the requirements and design are finished, but the team typically doesn't consider any work to be complete until those phases are finished. Which term BEST describes Donald's projects?

A. Iterative
B. Rolling wave planning
C. Waterfall
D. Scrum

Prüfungsfragen

9. Keith is the manager of a software team. He's made it clear that mistakes are not to be tolerated. A developer spent several hours building "proof of concept" code to test a possible approach to a complex problem. When he eventually discovered from the experiment that the approach wouldn't work, Keith yelled at him in front of the whole team and threatened to fire him if he did it again.

Which agile principle BEST applies to this situation?

A. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

B. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

C. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

D. Continuous attention to technical excellence and good design enhances agility.

10. What's the highest priority of an agile team?

A. Maximizing the work not done

B. Satisfying the customer by delivering valuable software early and often

C. Welcoming changing requirements, even late in development

D. Using iteration to effectively plan the project

11. Which of the following statements is NOT true about the daily standup?

A. The length is kept short by having everyone stand for the duration of the meeting

B. It's the same thing as a status meeting

C. It is most effective when everyone listens to each other.

D. It's an opportunity for every team member to get involved in planning the project.

12. Which of the following BEST describes the agile mindset with respect to simplicity?

A. Maximizing the work not done

B. Satisfying the customer by delivering valuable software early and often

C. Welcoming changing requirements, even late in development

D. Using iteration to effectively plan the project

13. A'ja is a project manager on a team that is just starting their agile adoption. The first change they made to the way they work was to start holding daily standup meetings. Several team members have approached her to say that they don't like attending. And despite the fact that she's getting some valuable information from the team at each standup, A'ja is concerned that the extra lines of communication might not be worth damaging the team cohesion.

Prüfungsfragen

What is the BEST thing for A'ja to do?

A. Stop holding the daily standup and find another way to adopt agile.
B. Make and enforce a rule that every attendee must put away his or her phone and pay attention.
C. Follow up with people individually after the meeting to get more detailed status.
D. Work with the team on changing their mindset.

14. You're a developer on a software team. A user has approached your team about building a new feature, and has provided requirements for it in the form of a specification. She is very certain of exactly how the feature will work, and promises there will be no changes. Which agile value BEST applies to this situation?

A. Individuals and interactions over processes and tools
B. Working software over comprehensive documentation
C. Customer collaboration over contract negotiation
D. Responding to change over following a plan

15. Which of the following is NOT a benefit of welcoming changing requirements?

A. It gives the team a way to explain a missed deadline
B. The team builds more valuable software when customers aren't pressured not to change their minds
C. There's more time and less pressure so the team can make better decisions
D. Less code is written before changes happen, which minimizes unnecessary rework

16. Which of the following is NOT part of an agile team's mindset toward working software?

A. It contains the final version of all features
B. It is the primary measure of progress
C. It is delivered frequently
D. It is an effective way to get feedback

17. Which of the following is NOT true about iteration?

A. The team must finish all planned work by the end of an iteration
B. Iterations have a fixed deadline
C. The scope of work performed during an iteration may change by the time it ends
D. Projects typically have multiple sequential iterations

Prüfungsfragen antworten

Hier finden Sie die Antworten zu den Übungsprüfungsfragen aus diesem Kapitel. Wie viele haben Sie richtig? Haben Sie eine falsch beantwortet, ist das auch okay – es lohnt sich, zurückzublättern und den entsprechenden Teil des Kapitels nochmals zu lesen, damit Sie verstehen, worum es geht.

1. Antwort: D

Klang diese Situation negativ? So als wäre etwas so richtig schiefgelaufen? Wenn ja, sollten Sie vielleicht über Ihr eigenes Mindset nachdenken! Das war nämlich tatsächlich eine ziemlich genaue Beschreibung eines sehr erfolgreichen agilen Projekts, das eine iterative Vorgehensweise verfolgte. Es klingt nur so, als hätte das Projekt Probleme, wenn Sie es aus einer Haltung heraus angehen, bei der Veränderung und Iteration Fehler statt hilfreiche Aktivitäten sind. Wenn Sie das Projekt so sehen, sind Sie vielleicht dazu verleitet, dem Team die »Schuld« daran zu geben, dass es die Anforderungen nicht verstanden hat, oder den Anwendern, nicht zu wissen, was sie wollen, oder dem Prozess, weil er keine adäquaten Steuerelemente zum Verhindern und Verwalten von Veränderung hat. Agile Teams denken nicht so. Sie wissen, dass die beste Möglichkeit, herauszubekommen, was die Anwender brauchen, ein frühes und häufiges Ausliefern funktionierender Software ist.

2. Antwort: C

Projektmanager sind sehr wichtig, aber es gibt in Scrum keine spezifische Rolle mit dem Namen »Projektmanager«. Bei Scrum gibt es drei Rollen: Scrum Master, Product Owner und Teammitglied. Der Projektmanager wird in einem Projekt, das Scrum nutzt, eine dieser Rollen übernehmen, häufig aber weiterhin als Jobtitel nur »Projektmanager« tragen.

Wenn Ihr Team eine agile Vorgehensweise verfolgt, die bestimmte Rollen nutzt, entspricht die von Ihnen ausgefüllte Rolle nicht immer der auf Ihrer Visitenkarte, insbesondere dann, wenn Ihr Team gerade erst damit beginnt, die Vorgehensweise zu übernehmen.

3. Antwort: B

Es stimmt, dass agilen Teams die Zusammenarbeit mit Kunden wichtig ist, dass sie daran glauben, dass direkte Gespräche die effektivste Methode zum Vermitteln von Informationen sind und dass die oberste Priorität das Ausliefern von Software ist. Aber der Anwender hat sich die Zeit genommen, die Spezifikation zu schreiben, und die Information darin könnte sehr hilfreich sein – beim Schreiben von Code oder bei späteren direkten Gesprächen.

Wenn sich jemand die Zeit nimmt, Informationen aufzuschreiben, die seiner Meinung nach wichtig sind, wäre es sehr UN–kooperativ, sie zu ignorieren.

4. Antwort: D

Wenn agile Teams von funktionierender Software sprechen, meinen sie Software, die sie als »Done« betrachten und die den Anwendern vorgeführt werden kann. Aber es gibt keine Garantie, dass sie die Bedürfnisse der Anwender oder die Anforderungen in einer Spezifikation erfüllt. Tatsächlich ist der effektivste Weg, Software zu bauen, die den Benutzern wirklich hilft, ein häufiges Ausliefern funktionierender Software. Das liegt daran, dass die frühen Versionen funktionierender Software im Allgemeinen **die Anforderungen der Benutzer nicht komplett erfüllen,** und die einzige Möglichkeit, das herauszufinden, ist die Übergabe in die Hände der Anwender, sodass diese Feedback dazu geben können.

Das ist der Grund dafür, dass agilen Teams eine frühe und kontinuierliche Auslieferung funktionierender Software wichtig ist.

Prüfungsfragen antworten

5. Antwort: C

Das Agile Manifest enthält zentrale Werte, die von effektiven agilen Teams geteilt werden. Es definiert keinen »besten« Weg, um Software zu bauen, und auch kein Regelwerk, das alle Teams befolgen sollten. Denn agile Teams wissen, dass es keinen »one-size-fits-all«-Ansatz gibt, der für alle Teams funktioniert.

6. Antwort: B

Scrum-Teams arbeiten in Sprints, die meist (aber nicht immer) 30 Tage lang sind. Sie planen die Arbeit der nächsten 30 Tage (sofern dies die Sprintlänge ist) am Anfang des Sprints. Am Ende des Sprints demonstrieren sie den Anwendern funktionierende Software und halten eine Retrospektive ab, um zu besprechen, was gut lief und wie sich etwas verbessern ließe.

7. Antwort: C

Das Projekt hat Probleme, weil das Team nicht gut mit seinem Kunden zusammenarbeitet. Hier sind die Netzwerktechniker die Kunden, weil diese die Software einsetzen werden. In dieser Situation wäre es einfach, auf Vertragsverhandlungen zurückzugreifen, spezifische Bedingungen festzulegen und Dokumente zu erstellen, um zu beschreiben, was gebaut werden wird, sodass mit der Softwareentwicklung begonnen werden kann. Aber es ist effektiver, wirklich mit dem Kunden zusammenzuarbeiten, um die beste technische Lösung herauszufinden.

8. Antwort: C

Ein Wasserfall-Projekt ist in Phasen unterteilt, meist beginnend mit der Anforderungs- und Designphase. Viele Wasserfall-Teams beginnen schon mit »Vorarbeiten« am Code, wenn die Anforderungen und das Design halbwegs stabil (aber noch nicht abgeschlossen) sind. Das ist jedoch definitiv nicht das Gleiche wie Iteration, weil das Team den Plan nicht abhängig von seinen Erfahrungen beim Bauen und Vorführen funktionierender Software anpasst.

9. Antwort: B

Agile Projekte entstehen durch motivierte Teammitglieder. Keith macht Dinge, die die Motivation des Teams untergraben, indem er ein Teammitglied niedermacht, der ein sinnvolles Risiko eingeht und wirklich versucht, das Projekt besser zu machen.

Prüfungsfragen antworten

10. Antwort: B

Blättern Sie zurück und lesen Sie nochmals das erste Prinzip von Agile: »Unsere höchste Priorität ist es, den Kunden durch frühe und kontinuierliche Auslieferung wertvoller Software zufriedenzustellen.« Der Grund dafür, dass dies die höchste Priorität hat, ist, dass sich agile Teams vor allem auf das Ausliefern von wertvoller Software konzentrieren. Alle anderen Dinge im Projekt – Planen, Designen, Testen, Meeten, Diskutieren, Dokumentieren – sind wirklich, wirklich wichtig, aber alle dienen nur dem Liefern wertvoller Software an unsere Kunden.

11. Antwort: B

Manche Teams behandeln das Daily Standup zwar als Status-Meeting, in dem jedes Teammitglied ein Update an einen Chef oder Projektmanager gibt, aber das ist eigentlich nicht Sinn des Ganzen. Am besten funktioniert es, wenn jeder den anderen zuhört und das Meeting nutzt, um das Projekt im Team zu planen.

12. Antwort: A

Agilen Teams ist Einfachheit wichtig, weil man mit einfachen Designs und einfachem Code leichter arbeiten kann und er sich leichter warten und ändern lässt als eine komplexe Variante. Einfachheit wird häufig beschrieben als »die Kunst, die nicht zu erledigende Arbeit zu maximieren«, denn – insbesondere bei Software – ist weniger zu tun der effektivste Weg, etwas einfach zu halten.

13. Antwort: D

Das Team ist beim Daily Standup deshalb nicht aufmerksam, weil es ihm egal ist oder weil es das Meeting nicht als effektives Werkzeug ansieht. Vor allem wollen die Leute, dass es so schnell wie möglich wieder abgeschafft wird, damit sie schnell zurück an ihre »richtige« Arbeit können. Wenn Teams diese Haltung haben, werden sie das Meeting irgendwann nicht mehr besuchen, und die Übernahme agiler Vorgehensweisen wird sehr viel weniger wahrscheinlich erfolgreich sein. Das Daily Standup wird effektiver sein, wenn das Team versteht, wie das Meeting allen hilft – sowohl jedem einzelnen als auch dem Team als Ganzes. Dieser Mentalitätswechsel kann nur durch offene und ehrliche Gespräche darüber erreicht werden, was funktioniert und was nicht. Das ist der Grund dafür, dass eine Zusammenarbeit mit dem Team zum Ändern ihres Mindset die beste Vorgehensweise in dieser Situation ist.

14. Antwort: B

Natürlich ist es sinnvoll, die Spezifikation zu lesen und zu verstehen. Aber am effektivsten stellt das Team sicher, dass es ihre Wünsche verstanden hat, indem es ihr funktionierende Software ausliefert, sodass sie selbst sehen kann, wie die von ihr dokumentierten Anforderungen interpretiert wurden. Dann kann sie mit dem Team zusammen herausfinden, was gut funktioniert und was geändert werden muss.

Prüfungsfragen antworten

15. Antwort: A

Es gibt viele gute Gründe dafür, dass agile Teams sich ändernde Anforderungen willkommen heißen. Wenn Kunden darin unterstützt werden, ihre Meinung zu ändern (statt sie davon abzuhalten), geben diese bessere Informationen an das Team, was wiederum zu besserer Software führt. Selbst wenn die Leute in Bezug auf Änderungen ihren Mund halten, werden diese so gut wie immer schließlich doch aufgedeckt, und das Team hat mehr Zeit, darauf zu reagieren, wenn sie diese Informationen frühzeitig erhält – und je früher die Änderungen bekannt werden, desto weniger Code muss umgearbeitet werden.

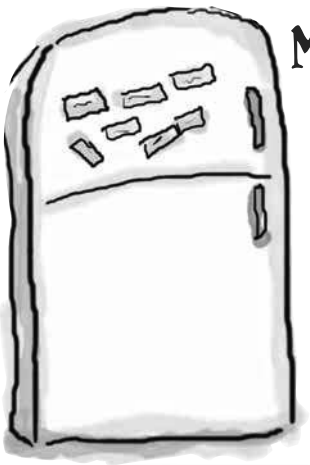
Trotzdem sind Änderungen niemals eine Ausrede für schlechte Planung oder verpasste Deadlines. Effektive agile Teams haben im Allgemeinen eine Vereinbarung mit ihren Anwendern: Das Team heißt sich ändernde Anforderungen von Anwendern, Kunden und Managern willkommen, dafür wird ihm nicht die Schuld daran gegeben, dass es Zeit braucht, darauf zu reagieren. Denn jeder erkennt an, dass es sich auf diesem Weg trotzdem um die schnellste und effektivste Möglichkeit handelt, Software zu bauen. Darum sieht niemand das Begrüßen von Änderungen als Möglichkeit für das Team an, sich bei verpassten Deadlines herauszureden, denn die Deadlines sollten immer mit Rücksicht auf die Änderungen angepasst werden.

16. Antwort: A

Funktionierende Software wird häufig ausgeliefert, damit das Team auch häufig Rückmeldung bekommt und Änderungen frühzeitig vornehmen kann. Funktionierende Software sollte also niemals als finale Version einer Anforderung angesehen werden. Deswegen heißt sie »funktionierende« Software, nicht »abgeschlossene« Software.

17. Antwort: A

Iterationen sind timeboxed, die Deadline ist also fest, und der Scope wird so angepasst, dass sie gehalten werden kann. Das Team beginnt jede Iteration mit einem Planungsmeeting, um zu entscheiden, welche Arbeit erledigt werden wird. Wenn sich dann herausstellt, dass der Plan nicht gut war und es länger dauert als erwartet, wird alle Arbeit, die nicht erledigt werden kann, zurück ins Backlog geschoben und neu priorisiert (und oft in die nächste Iteration übertragen).



Manifest-Magneten, Lösung

Manifest für Agile Softwareentwicklung

Wir erschließen bessere Wege, Software zu entwickeln,
indem wir es selbst tun und anderen dabei helfen.

Durch diese Tätigkeit haben wir diese Werte zu schätzen gelernt:

Individuen und Interaktion mehr als Prozesse und Werkzeuge

Funktionierende Software mehr als umfassende Dokumentation

Zusammenarbeit mit dem Kunden mehr als Vertragsverhandlung

Reagieren auf Veränderung mehr als das Befolgen eines Plans

Das heißt, obwohl wir die Werte auf der rechten Seite wichtig finden,
schätzen wir die Werte auf der linken Seite höher ein.

wort-wissen LÖSUNG

Hier finden Sie ein paar Definitionen für Wörter, die Ihnen in diesem Kapitel über den Weg gelaufen sind. Können Sie sie den entsprechenden Definitionen zuordnen?

Wir haben »Überarbeitung« schon zuvor im Kapitel genutzt: Es ist eine wichtige Quelle für Fehler.

Überarbeitung, Nomen

Vom Team geleistete Arbeit, um zuvor geschriebenen Code einem anderen Zweck zuzuführen oder anders arbeiten zu lassen, wird von Teams aufgrund der erhöhten Bug-Anfälligkeit oft als riskant angesehen.

Es kann auch als Verb auftreten: »Wir mussten dieses Stück Code überarbeiten, um es einem neuen Zweck zuzuführen.«

timeboxed, Adjektiv

Eine harte Deadline für das Fertigstellen einer Aktivität setzen und dann den Scope der Aktivität anpassen, damit die Deadline gehalten wird.

Das kann man auch als Verb nutzen: »Lasst uns die Arbeit an diesem Feature auf sechs Stunden timeboxen.«

Backlog, Nomen

Eine von Teams genutzte Praktik, bei der das Team mit Anwendern, Kunden und/oder Stakeholdern zusammenarbeitet, um eine Liste mit zukünftigen Features zu erstellen – häufig so geordnet, dass das wertvollste Feature ganz oben steht.

fortschreitende Ausarbeitung, Adjektiv und Nomen

Ein Projekt-Artefakt (wie zum Beispiel einen Plan) schrittweise entwickeln und dabei das Wissen aus dem vorigen Schritt nutzen, um es zu verbessern.

iterativ, Adjektiv

Eine Art von Vorgehensweise, bei der Teams das Projekt in kleinere Einheiten unterteilen, am Ende jeder dieser Einheiten funktionierende Software ausliefern und, abhängig vom Feedback, eventuell die Richtung zu funktionierender Software anpassen.

Wasserfall, Nomen

Ein Typ eines Modells, Prozesses oder einer Methode für das Bauen von Software, in der das gesamte Projekt in aufeinanderfolgende Phasen unterteilt wird, oft in Verbindung mit einem Prozess zum Änderungsmanagement, bei dem Änderungen ein Projekt in eine frühere Phase zurückwerfen.

Ein Beispielsatz: »Brian hat bisher in einer Firma gearbeitet, die einen Wasserfall-Prozess nutzt, daher freut er sich wirklich darüber, jetzt einen agilen Prozess wie Scrum ausprobieren zu können.«

Es ist nicht immer leicht, sich ein agileres Mindset anzueignen! Manchmal klappt es problemlos, aber manchmal ist ein wenig Aufwand nötig. Hier sind ein paar Dinge, die wir Mike, Kate und Ben haben sagen hören. Ziehen Sie eine Linie von jeder Sprechblase hin zu KOMPATIBEL oder INKOMPATIBEL und dann zu dem agilen Prinzip, mit dem die Aussage entweder kompatibel oder inkompatibel ist.

Warum fragt ihr mich etwas? Ich habe doch schon alles, was die Anwender haben wollen, in die Spezifikation geschrieben.

KOMPATIBEL

INKOMPATIBEL

Nach Kates Entdeckung hätten sie entweder eine Software ausliefern können, die nicht funktioniert, oder die Auslieferung verzögern können, um den Fehler zu beheben. Aber das Verschieben dieses Features in den nächsten Iterationszyklus ist die bessere Wahl, weil so trotzdem eine funktionierende Software mit den restlichen Features ausgeliefert wird.

Ich habe gerade entdeckt, dass unser Algorithmus zum Berechnen der Zuhöreranzahl nicht funktioniert. Wir müssen dieses Feature in die nächste Iteration verschieben.

KOMPATIBEL

INKOMPATIBEL

Okay, wer von euch Idioten hat diesen fehlerstrotzenden Spaghetti-Code geschrieben? Es ist dein Fehler, dass wir nicht im Zeitplan liegen.

KOMPATIBEL

INKOMPATIBEL

Wenn sich Kate nur auf ihren Zeitplan verlassen hätte, um den Fortschritt zu messen, würde sie vielleicht denken, dass das Projekt gut läuft. Das Berücksichtigen lauffähiger Software als zentrales Fortschrittsmaß hilft ihr dabei, Probleme frühzeitig zu entdecken (und hoffentlich zu beheben!).

Ich lasse den letzten Build laufen, aber ich dachte, wir wären mit dem Analytics-Feature schon weiter. Gibt es ein Problem, von dem ich nichts weiß?

KOMPATIBEL

INKOMPATIBEL

Funktionierende Software ist der wichtigste Maßstab des Fortschritts.

Es ist nicht fair, Anwender zu Beginn des Projekts nach Anforderungen zu fragen und es ihnen dann zu verwehren, die Meinung zu ändern (solange sie verstehen, dass das Team für die Änderungen Zeit braucht).

Heißen Sie Anforderungsänderungen selbst spät in der Entwicklung willkommen. Agile Prozesse nutzen Veränderungen zum Wettbewerbsvorteil des Kunden.

Liefern Sie funktionierende Software regelmäßig innerhalb weniger Wochen oder Monate aus und bevorzugen Sie dabei die kürzere Zeitspanne.

Technikorientierte Menschen wie Mike sind häufig direkt. Aber auch wenn das Team eine Kultur pflegt, in der es in Ordnung ist, Leute herauszufordern oder sogar zu beleidigen, ist das Verantwortlichmachen einer Person für Verzögerungen oder Qualitätsprobleme echt demotivierend.

Errichten Sie Projekte rund um motivierte Individuen. Geben Sie ihnen das Umfeld und die Unterstützung, die sie benötigen, und vertrauen Sie darauf, dass sie die Aufgabe erledigen.

Mindset-Kreuzworträtsel, Lösung

			[1]F	R	U	E	H	Z	E	I	T	[2]I	G							[3]B				
												T								E		[4]M		
		[5]F	U	N	K	T	I	O	N	I	E	R	E	N					[6]D		S		E	
												R				[7]A	R	B	E	I	T	E	N	
						[8]W	A	S	S	E	R	F	A	L	L					M		E		S	
												T								O		N		C	
								[9]W				I								T				H	
			[10]A					E				[11]F	O	R	T	S	[12]C	H	R	I	T	[13]T		E	
			N			[14]M		R			[15]P	L	A	N			O			V		I		N	
			P			O		K					C				N			I		M			
			A			T		Z					E		[16]V	E	R	T	R	A	E	G	E		
[17]F			S			I		E					T					I			R		B		
E			S			V		U				[18]D	O	K	U	M	E	N	T	A	T	I	O	N	
[19]H	A	E	U	F	I	G		G					F					U					X		
L			N			E		[20]V	E	R	T	R	A	U	E	N		O					E		
E						R							C				[21]Z	U	F	R	I	E	D	E	N
R			[22]I	N	T	E	R	V	A	L	L	E	N					S							