PRO

BUSINESS

Sander Hoogendoorn

Das kleine **Agile-Buch**

ALWAYS LEARNING

PEARSON

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.dnb.de abrufbar.

Autorisierte Übersetzung der niederländischen Originalausgabe mit dem Titel Dit is Agile, von Sander Hoogendoorn, erschienen bei Pearson Benelux, ein Imprint von Pearson Education, Inc., Copyright 2012

10 9 8 7 6 5 4 3 2 1

15 14 13

ISBN 978-3-8273-3232-5 (Print); 978-3-86324-685-3 (PDF); 978-3-86324-250-3 (ePUB)

Fachlektorat: Monika B. Paitl, www.communications9.com
Einbandgestaltung: Marco Lindenbeck, webwo GmbH, m.lindenbeck@webwo.de
Übersetzung und Fachlektorat: Sonja Willner, willner-dolmetschen.de
Illustrationen: Sander Hoogendoorn
Lektorat: Boris Karnikowski, bkarnikowski@pearson.de
Herstellung: Martha Kürzl-Harrison, mkuerzl@pearson.de
Korrektorat: Judith Klein, Siegen
Satz: Gerhard Alfes, mediaService, Siegen (www.mediaservice.tv)
Druck und Verarbeitung: GraphyCems, Villatuerta
Printed in Spain

Kapitel 3

Kurze Iterationen

@AncientProverbs: Wir sind, was wir wiederholt tun. Hervorragende Leistung ist dann nicht eine Tat, aber eine Gewohnheit. – Aristoteles

Exkurs

Ich führte bei einem großen Projekt einer internationalen Bank ein Audit durch. An das erste Gespräch kann ich mich noch gut erinnern. Der Projektleiter erzählte mir stolz, sein Projekt laufe in Iterationen. Das fand ich bewundernswert, zumal ich wusste, dass er ein starker Verfechter von Wasserfall war. Während des Gesprächs dämmerte mir, dass er unter Iterationen etwas ganz anderes verstand als ich. „Wie lange dauern denn die Iterationen?" fragte ich. Als ob das vollkommen selbstverständlich sei, antwortete er ohne zu zögern: „Sechs Monate. Wieso?"

In der Theorie sind Iterationen von sechs Monaten auch Iterationen. Agile Iterationen sind jedoch meistens kürzer. Viel kürzer. In der Blütezeit von DSDM, den neunziger Jahren des vergangenen Jahrhunderts, waren Iterationen von sechs Wochen üblich. Seither sind Iterationen kürzer geworden und ich habe in Projekten mit vierwöchigen, dreiwöchigen und sogar einwöchigen Iterationen mitgearbeitet. In der Regel bin ich für Iterationen von zwei Wochen. Aber ehrlich gesagt unterscheidet sich die ideale Iterationslänge von Projekt zu Projekt.

3.1 Der Aufbau einer Iteration

Auch wenn sich die Bezeichnung, die Arbeitseinheiten und sogar die Iterationslänge in den verschiedenen Ansätzen unterscheiden, sind sich die Schriftgelehrten doch einig, *wie* eine Iteration aussieht.

Jede Iteration besteht aus drei Teilen:

- **Kickoff.** Jede Iteration beginnt mit einem Kickoff, in dem vom ganzen Team festgelegt wird, welche Work Items in dieser Iteration angegangen werden. Dabei werden so viele Work Items ausgewählt, dass das Team diese umsetzen kann. Nicht zu viel und nicht zu wenig. Das Team gibt dazu ein Commitment ab.
- **Arbeit.** Der größte Teil der Iteration besteht aus der Umsetzung der besprochenen Work Items. Die Umsetzung beinhaltet die Analyse, den Entwurf, Coden, Testen und auch die Abnahme der Work Items. Der Arbeitsfortschritt wird täglich im sogenannten Stand-Up festgestellt.
- **Evaluation.** Am Ende jeder Iteration findet eine Evaluation statt. Mit dem ganzen Team. Und den betroffenen Stakeholdern. Die gelieferten Work Items werden evaluiert. Oft gibt es dazu eine Demo. Aber zusätzlich wird auch die Arbeitsweise des Teams unter die Lupe genommen. Ist die Qualität ausreichend? Sind wir schnell genug? Können wir noch schneller werden? Verbesserungen der Arbeitsweise werden direkt in der nächsten Iteration in Angriff genommen. Und bei der Evaluation der nächsten Iteration werden diese Verbesserungen wieder evaluiert.

3.2 Warum Iterationen?

Iteratives Arbeiten ist eine wirkungsvolle Waffe. Viele Nachteile des Wasserfallmodells werden damit abgefangen. Die wichtigsten Vorteile auf einen Blick:

- **Risiken mindern.** Natürlich gibt es bei Agile die gleichen Risiken wie bei Wasserfall. Auch in agilen Projekten kann man an die Grenzen der Architektur stoßen, für die man sich entschieden hat, ist die Benutzeroberfläche anders als gewünscht, lässt die Performance manchmal zu wünschen übrig, funktioniert in der Produktiv-Umgebung doch nicht alles und sind die Schnittstellen mit anderen Systemen, Services und Middleware kompliziert. Der große Unterschied ist, dass in Iterationen die *gesamte* Arbeit an den geplanten Work Items durchgeführt wird. So erkennt man Risiken nicht erst

am Ende des Projekts, sondern schon während der ersten Iterationen. Es ist genau dieses Vorziehen der Risiken, das die möglichen Folgen deutlich abfedert.

- **Umgang mit Veränderungen.** Zu Beginn jeder Iteration werden die Work Items ausgewählt, die zu diesem Zeitpunkt am wichtigsten sind. Für die Entscheidung kann es eine Vielzahl von Gründen geben. Wenn während einer Iteration neue Work Items identifiziert werden oder bestehende verändert werden, werden diese schlicht und einfach dem Backlog hinzugefügt. Wenn sie sich als wichtig herausstellen, dann können sie bereits für die nächste Iteration ganz nach vorne geholt werden. In Wasserfall werden dafür *Change Requests* erfasst. Diese werden gesammelt und im günstigsten Fall am Ende des Projekts noch umgesetzt. Wenn sich bei näherer Betrachtung herausstellt, dass bereits erfasste Work Items unwichtig sind, verschwinden sie bei agilem Vorgehen, noch bevor viel Arbeit investiert worden ist.
- **Zusammenarbeit der Rollen.** Während jeder Iteration werden die ausgewählten Work Items vollständig umgesetzt. Analyse, Entwurf, Coden, Testen und die Abnahme der Work Items folgen aufeinander wie bei einem Staffellauf, der im Idealfall innerhalb eines Tages beendet ist. Die verschiedenen Rollen arbeiten also nicht in verschiedenen Phasen des Projekts, wie bei Wasserfall, sondern gleichzeitig. Alle Rollen profitieren so von dem Wissen und der Erfahrung der anderen. Nehmen wir zum Beispiel das Erstellen eines Entwurfs für ein Work Item. Daran sind in einem Workshop der Fachexperte, der Entwickler und der Tester beteiligt. Im Idealfall trägt auch noch der Endanwender dazu bei. Mit so einem rollenübergreifenden Workshop kann man Fehler vermeiden, bevor sie gemacht werden. Dadurch steigt die Qualität der Work Items und auch die Geschwindigkeit, mit der sie umgesetzt werden, enorm.
- **Timeboxes.** Wie lange es tatsächlich dauert, ein Work Item umzusetzen, kann man nicht bis ins letzte Detail planen. Manchmal dauert die Umsetzung einfach länger. Manchmal gibt es technische Probleme. Es ist allerdings unvernünftig, Iterationen so lange laufen zu lassen, bis alle Work Items umgesetzt sind. Iterationen werden in *Timeboxes* durchgeführt. Sie haben eine feste Länge. Work Items, die zum Ende der Timebox nicht fertig sind, kommen zurück ins Backlog. Mir der Arbeit in Timeboxes verhindert man Endlositerationen.

- **Messen heißt wissen.** Die Umsetzung einzelner Work Items erleichtert die Messung des Fortschritts. Ein Work Item ist fertig oder nicht. Ziemlich binär. Zu jedem Zeitpunkt des Projekts kann man also feststellen, wie viele Work Items fertig sind, häufig in Punkten. Diese genaue Zahl macht es möglich, früh und prompt zu reagieren, wenn es zu Verzögerungen kommt.

Exkurs

In einem meiner Projekte sind momentan 758 der 1408 Smart-Use-Case-Punkte umgesetzt. Wir haben also gerade die Hälfte geschafft.

3.3 Das Kickoff

Jede Iteration beginnt mit einem Kickoff. Dabei ist das wichtigste Ziel, festzustellen, welche Work Items in der nächsten Iteration umgesetzt werden. Jeder, der am Projekt beteiligt ist, nimmt an diesem Kickoff teil. Auftraggeber, Endanwender, Projektleiter, Fachexperten, Entwickler und Tester.

Die Work Items werden aus der Gesamtliste der noch verbleibenden Work Items, dem Projektbacklog, ausgewählt. Der Kunde trifft diese Wahl. Er bezahlt schließlich für das Projekt. Oft delegiert der Kunde diese Entscheidung auch an die Endanwender oder an die Fachexperten. Sie haben täglich mit dem Projekt zu tun und kennen die Materie, das Fach, wie ihre Westentasche. Manchmal wird auch jemand eigens für diese Rolle ausgewählt. In Scrum heißt diese Person *Product Owner*. Die Liste der gewählten Work Items für eine Iteration heißt *Iterationsbacklog* und in Scrum *Sprintbacklog*.

Doch selbst für diejenigen, die tagtäglich mit dem Projekt zu tun haben, ist es nicht immer einfach, diese Wahl zu treffen. Viele Aspekte müssen berücksichtigt werden:

- **Wert.** Viele agile Ansätze sehen vor, dass die Work Items, die für den Kunden den meisten Wert schöpfen, als erste umgesetzt werden müssen. Leider ist es oft nicht einfach, festzustellen, welche Work Items das denn genau sind.
- **Zusammenhang.** Die meisten Work Items stehen nicht für sich, sondern im Zusammenhang mit anderen. Deshalb gibt es in Smart *Smart Use Cases* als Arbeitseinheit. Mehrere Smart Use Cases stellen einen Geschäftsprozess dar. Aus diesem Grund ist es nicht immer

einfach, einzelne Work Items auszuwählen. Die Wahl eines Work Items bedingt auch die Wahl eines anderen.

- **Risiken.** Es ist vernünftig, Work Items mit hohen Risiken frühzeitig einzuplanen. Dazu gehören zum Beispiel die Verfügbarkeit von Entwicklungsumgebungen, Spezialisten, die mit einbezogen werden müssen, die Performance, das Layout der Benutzeroberfläche oder die Verfügbarkeit von Webservices, die von anderen Projekten oder Unternehmen zur Verfügung gestellt werden müssen. Umso eher man Risiken angeht, desto geringer die Auswirkungen.
- **Balance.** Denken Sie an die Balance zwischen den verschiedenen Rollen. Sorgen Sie dafür, dass allen Rollen ausreichend Arbeit zugeteilt wird.

Exkurs

In einem Projekt für eine Versicherung mussten vier verschiedene Technologien verknüpft werden: drei externe Lösungen und Java. Also bestand das Team aus vier Entwicklerarten. Die Spezialisten für die externen Technologien waren rar und teuer und mussten somit laut Projektleiter „gut beschäftigt bleiben".

Um das klarzustellen: Es ist *nicht* der Projektleiter, der entscheidet, welche Work Items umgesetzt werden, wie das traditionell oft der Fall ist. Es ist auch nicht das Team, das die Wahl trifft. Natürlich werden Projektleiter und Team mit einbezogen und beraten bei der Entscheidung. Aber der Kunde sucht aus.

Damit die Auswahl der Work Items während des Kickoffs effizient verläuft, ist es vernünftig, vorab eine Vorauswahl zu treffen.

Exkurs

Ein Projekt bei einem Logistikunternehmen arbeitete mit Smart Use Cases als Arbeitseinheit. Schon vor dem Kickoff der folgenden Iteration hatte der Product Owner alle Diagramme ausgedruckt. Auf den Diagrammen hatte er festgehalten, welche Use Cases schon umgesetzt waren, welche er unbedingt und welche er vielleicht in der nächsten Iteration umsetzen wollte. Das Iterationsbacklog stand schnell – genauer gesagt in fünf Minuten.

Nun bleibt noch zu klären, *wie viele* Work Items in der folgenden Iteration geliefert werden können. Darüber entscheidet alleine das Team. Auch dabei muss ein Gleichgewicht gefunden werden. Nicht zu viel und nicht zu wenig:

- **Zu viel.** Es ist besser, nicht zu viele Work Items für eine Iteration zu wählen. Das macht es schwieriger, alle Work Items zu schaffen. An sich ist das nicht schlimm. Allerdings verlieren meiner Erfahrung nach Teams, die mehrere Iterationen hintereinander die geplanten Work Items nicht schaffen, schnell die Motivation. Teams, die gerade erst begonnen haben, agil zu arbeiten, zweifeln dann schnell an der agilen Arbeitsweise.

Exkurs

Im ersten agilen Projekt eines Unternehmens wussten wir nicht, wie schnell das Team arbeiten würde. Deshalb stellten wir die Ausgangsgeschwindigkeit des Teams in einer Stichprobe fest. Für einige relevante Work Items schätzten wir die Arbeitsstunden, die voraussichtlich für die Umsetzung benötigt würden. Dividiert durch die Anzahl geschätzter Punkte kamen wir so auf einen Wert für Stunden pro Punkt. Auf dieser Basis planten wir die Work Items für die erste Iteration. Schon nach drei Iterationen zeigten unsere Kennzahlen, dass wir immer wieder zu viele Work Items eingeplant hatten. Das frischgebackene agile Team hatte uns das schon signalisiert: Sie fanden die Iterationen zu kurz.

- **Zu wenig.** Es ist besser, nicht zu wenig in eine Iteration einzuplanen. Hier treten andere Phänomene auf. Eins davon ist das *Parkinsonsche Gesetz*. Es besagt: *„Arbeit dehnt sich in genau dem Maß aus, wie Zeit für ihre Erledigung zur Verfügung steht."* Auch wenn sich das Parkinsonsche Gesetz ursprünglich auf die Zunahme der Bürokratie bezog, ist dieses Gesetz doch auch auf Agile anwendbar. Wenn zu wenige Work Items ausgewählt werden, besteht das Risiko, dass diese bis ins Letzte in die Länge gezogen und *besser* geliefert werden, als nötig gewesen wäre. In dem Zeitraum, in dem sechs Work Items umgesetzt wurden, hätte man auch locker acht umsetzen können. In Agile gilt, dass gut gut genug ist. Oder wie ein Entwickler einmal schön sagte: „Gut genug ist besser als perfekt".
  Ein anderes Phänomen ist das *Studentensyndrom*. Das ist das Phänomen, dass wenn genügend Zeit für eine Arbeit vorhanden zu sein scheint, Menschen so spät wie möglich damit anfangen. Wer studiert hat, dem dürfte das bekannt vorkommen. Auch in Agile funktioniert das Studentensyndrom. Wenn die Zeit für die Umsetzung der geplanten Work Items vollkommen auszureichen scheint, neigt das Team dazu, erst andere Aufgaben anzugehen.

Exkurs

Ich coachte ein Projekt, in dem jedes Teammitglied diverse andere Aufgaben hatte. Immer wieder waren sie für das Projekt nicht verfügbar, weil sie sich um allerlei andere Probleme kümmern mussten. Die Folge dessen war, dass die Work Items immer in den letzten Tagen der Iteration und somit in Hast und Eile umgesetzt wurden.

Um festzustellen, wie viele Work Items umgesetzt werden können, ist es wichtig, festzuhalten, wie viele Work Items in früheren Iterationen umgesetzt worden sind. Die Zahl der Work Items für Iteration zwanzig entspricht ungefähr der durchschnittlichen Anzahl Work Items, die in den ersten neunzehn Iterationen umgesetzt worden sind. Nehmen Sie diese Zahl also auch als Richtwert. Es ist schließlich unwahrscheinlich, dass die Produktivität in Iteration zwanzig plötzlich ganz anders ist.

Nicht alle Work Items sind gleich groß oder gleich schnell umzusetzen. Ich rate dazu, alle Work Items auf einer *relativen* Punkteskala zu schätzen und mithilfe dieser Punkte festzustellen, wie viel in eine Iteration hineinpasst. So kann die Geschwindigkeit des Projekts (die *Velocity*) in der Menge an Zeit ausgedrückt werden, die erforderlich ist, um einen einzelnen Punkt umzusetzen. Das ist einfach auszurechnen, indem man die Zahl der insgesamt geleisteten Stunden durch die insgesamt abgenommenen Punkte teilt. Eine zweite wichtige Kennzahl ist die Iterationsgeschwindigkeit oder Iteration Velocity. Das ist die Anzahl der Punkte, die im Verlauf des gesamten Projekts im Schnitt pro Iteration umgesetzt worden sind.

Exkurs

In einem meiner Projekte liegt die Velocity bei 10,7 Stunden pro Smart-Use-Case-Punkt. Die Iteration Velocity liegt über 42 Iterationen bei 25,2 Smart-Use-Case-Punkten.

Mithilfe der Iterationsgeschwindigkeit kann der Kunde während des Kickoffs *ungefähr* so viele Punkte aus den verbleibenden Work Items auswählen. So wird für die beginnende Iteration nicht zu viel, aber auch nicht zu wenig Arbeit eingeplant.

Wichtig ist auch, unter welchen Voraussetzungen die Work Items als fertig angesehen werden. Diese Voraussetzungen bezeichnet man als die *Definition of Done*. Der Use Case ist dokumentiert. Der Code ist

geschrieben. Alle Unit Tests sind erfolgreich. Der Tester hat sein Okay gegeben. Der Endanwender hat das Work Item abgenommen.

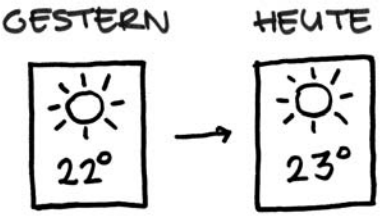
Solange mindestens eine dieser Voraussetzungen nicht erfüllt ist, ist das Work Item nicht fertig. Erst wenn alle Voraussetzungen erfüllt sind, ist das Work Item fertig und das Team *bekommt* die Punkte. Dabei ist zu bedenken, dass es bei sehr unterschiedlichen Work Items länger dauert, für die einzelnen Work Items die Definition of Done festzustellen. Wählen Sie möglichst homogene Arbeitseinheiten innerhalb eines Projekts. So kann für die meisten Work Items dieselbe Definition of Done verwendet werden. Das erspart viel Arbeit. Außerdem ist dieser Ansatz eingängiger.

Kurzum, es ist wichtig, die Zahl der Work Items, die innerhalb eines Releases oder einer Iteration umgesetzt müssen werden, so sorgfältig, realistisch und ehrlich wie möglich festzustellen. Dafür müssen die Work Items realistisch und auf einer einheitlichen Skala eingeschätzt werden und es muss von Tag eins an im Projekt gemessen werden, mit welcher Geschwindigkeit tatsächlich gearbeitet wird.

3.4 Yesterday's Weather

Beim Planen von Work Items ist oft der Wunsch der Vater des Gedankens. Selbst wenn ein Team schon seit mehreren Iterationen mit einer bestimmten Geschwindigkeit arbeitet, kann der Projektleiter auf besorgniserregende Ideen kommen: „Wenn wir jetzt in den kommenden Iterationen mehr Punkte hinbekommen, schaffen wir es genau pünktlich zur Deadline." Das ist ein gefährlicher Gedanke. Das Team arbeitet schließlich mit einer *erwiesenen* Geschwindigkeit.

Es ist nicht realistisch, für kommende Iterationen mit einer höheren Geschwindigkeit zu kalkulieren. Das schadet der Motivation des Teams. Außerdem weckt es beim Kunden falsche Erwartungen. Dieser Projektleiter tut gut daran, mit dem Prinzip *Yesterday's Weather* zu arbeiten. Dabei geht man davon aus, dass man die zuverlässigste Wettervorhersage für den heutigen Tag bekommt, wenn man sich das Wetter von gestern anschaut.



Übertragen auf Agile: Am besten kann man die Geschwindigkeit der folgenden Iterationen vorhersagen, wenn man die Geschwindigkeit der gerade abgeschlossenen Iterationen zur Hand nimmt. Planen Sie Iterationen anhand dessen, was erwiesenermaßen möglich ist, nicht anhand dessen, was Sie *hoffen*. Yesterday's Weather gilt übrigens auch für das Erstellen von Angeboten. Aber allzu oft will der jeweilige Account Manager den Angebotspreis runter kriegen, indem er die Produktivität für das neue Projekt höher einschätzt, als man auf der Basis früherer vergleichbarer Projekte erwarten könnte. Mein Tipp: *Keine* gute Idee, lieber Account Manager.

3.5 Die Evaluation

Wenn ich die Unterschiede zwischen Wasserfall und Agile erkläre, stelle ich immer wieder die Frage: Wer hat an der Abschlussevaluation seines letzten Projekts teilgenommen? Die Abschlussevaluation ist ein interessantes Phänomen. Das Ziel einer Evaluation ist, aus seinen Handlungen zu lernen und es beim nächsten Mal besser zu machen. Vom netten Bierchen mal abgesehen ist die Abschlussevaluation für das fertige Projekt selbst vollkommen sinnlos. Es ist schließlich abgeschlossen. Der Grund, warum man sie dennoch veranstaltet, ist, dass man für kommende Projekte etwas lernen will. Da tun wir uns meistens schwer. Trotz der Evaluationen machen wir in den nächsten Projekten einfach wieder die gleichen Fehler.

In Agile ist das anders. In Agile wird am Ende jeder Iteration eine Evaluation durchgeführt. Es kostet natürlich mehr Zeit, nach jeder Iteration zu evaluieren, aber es bewirkt auch viel. Ganz klar dann, wenn mögliche Verbesserungen früh in einem Projekt identifiziert werden. Von möglichen Verbesserungen, die in einer zweiten Iteration bemerkt werden, profitiert das Team schließlich schon ab der dritten Iteration. Ein chinesisches Sprichwort sagt: *„Verbesserung ist leichter als Perfektion."* Agile Projekte sind lernende Projekte.

Copyright