# Agile Scrum Handbuch

3. Ausgabe

Nader K. Rad

Van Haren PUBLISHING

Agile Scrum Handbuch

## Andere Ausgaben bei Van Haren Publishing

Van Haren Publishing (VHP) ist auf Veröffentlichungen über Best Practices, Methoden und Standards in den folgenden Domänen spezialisiert:

- IT und IT Management
- Enterprise-Architektur
- Business Management und
- Projektmanagement

Van Haren Publishing ist außerdem der Herausgeber von führenden Institutionen und Unternehmen darunter: ASLBiSL Foundation, BRMI, CA, Centre Henri Tudor, CATS CM, Gaming Works, IACCM, IAOP, IFDC, Innovation Value Institute, IPMA-NL, ITSqc, NAF, KNVI, PMI-NL, PON, The Open Group, The SOX Institute.

Die Themen der einzelnen Domänen sind:

### IT und IT Management

ABC of ICT
ASL® CMMI®
COBIT®
e-CF
ISO/IEC 20000
ISO/IEC 27001/27002
ISPL
IT4IT®
IT-CMF™
IT Service CMM
ITIL®
MOF
MSF
SABSA
SAF
SIAM™
TRIM
VeriSM™

### Enterprise-Architektur

ArchiMate®
GEA®
Novius Architectuur Methode
TOGAF®

### Projektmanagement

A4-Projektmanagement
DSDM/Atern
ICB / NCB
ISO 21500
MINCE®
M_o_R®
MSP®
P3O®
*PMBOK*®
*Guide* Praxis®
PRINCE2®

### Business Management

*BABOK*® *Guide* BiSL® und BiSL® Next
BRMBOK™
BTF
CATS CM®
DID®
EFQM
eSCM
IACCM
ISA-95
ISO 9000/9001
OPBOK
SixSigma SOX
SqEME®

Die neuesten Informationen zu VHP-Publikationen finden Sie auf unserer Website: www.vanharen.net.

# Agile Scrum Handbuch

## 3. Ausgabe

**Nader K. Rad**

Van Haren PUBLISHING

# Impressum


| | |
|---|---|
| Titel: | Agile Scrum Handbuch – 3. Ausgabe |
| Autor: | Nader K. Rad |
| Lektor: | Stephen Brightman |
| Übersetzung aus dem Englischen: | Andrea Kaufer-Ehm |
| Herausgeber: | Van Haren Publishing, 's-Hertogenbosch-NL, www.vanharen.net |
| DTP: | Coco Bookmedia, Amersfoort |
| ISBN Hard copy: | 978 94 018 0844 6 |
| ISBN ebook (pdf): | 978 94 018 0845 3 |
| ISBN ePUB: | 978 94 018 0846 0 |
| Auflage: | Dritte Aufsgabe, erste Auflage, März 2022 |
| Copyright: | Nader K. Rad & Van Haren Publishing |

Weitere Informationen zu Van Haren Publishing erhalten Sie per E-Mail unter: info@vanharen.net.

# Inhalt

# 1. Das Agile Konzept

Um Agile ranken sich viele Mythen und Legenden. Dies gilt nicht zuletzt auch für die Antwort auf die grundlegendste aller Fragen: **Was versteht man unter Agile?**

Die Antworten darauf sind häufig missverständlich wie: „Agile ist eine Einstellung, ein „Mindset“. Aber diese Aussage ist falsch. Agile ist kein Mindset. Richtig ist dagegen, dass Agile ein bestimmtes Mindset voraussetzt. Bezeichnet man Agile als Mindset, so führt dies in der Praxis lediglich dazu, dass manche Mitarbeiter: innen machen, was sie wollen und es Agile nennen, weil dies gerade in Mode ist.

Ein anderes auf diesem Gebiet weit verbreitetes Problem ist die Illusion des externen Feinds. Wer weiß, wie autoritäre Systeme funktionieren, weiß auch, dass ein solches System immer ein Feindbild braucht, um von den Schwächen im eigenen System abzulenken. Ein Feindbild schafft ein gemeinsames Ziel und täuscht darüber hinweg, dass es keine echten, erreichbaren internen Ziele gibt. Viele Agile-Fachleute gehen leider ähnlich vor, in der Regel zum persönlichen Vorteil einiger weniger Führungskräfte.

Die beste Strategie im Berufsleben ist jedoch, für alles offen zu sein und von allen Ansätzen und Methoden zu lernen. Etwas zum Kult zu erheben ist nie förderlich. Dies deckt sich auch mit dem ersten Grundsatz im Leitfaden Nearly Universal Principles of Projects: https://nupp.guide

Sehen wir uns nun also an, was Agile wirklich bedeutet.

## 1.1 Die verschiedenen Vorgehensweisen in der Softwareentwicklung

In der Softwareentwicklung werden folgende Schritte entweder für einzelne Features oder für die gesamte Softwarelösung auf die eine oder andere Weise durchgeführt:

- Analyse
- Design (Entwurf)
- Realisierung
- Integration
- Test

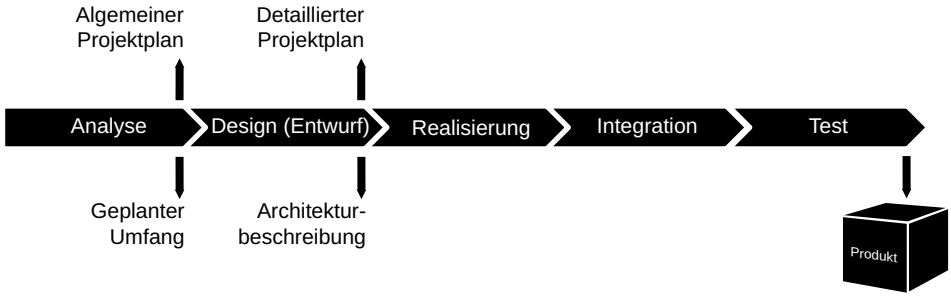
Selbstverständlich können diese Schritte auch anders bezeichnet, in weniger Schritte zusammengefasst oder in mehr Schritte aufgeteilt werden. Alles ist möglich. Zur Abgrenzung von Managementprozessen, wie Planung und Überwachung, werden diese Schritte auch als **Delivery-Prozesse** bezeichnet**.**

Wie also werden Sie diese Prozesse organisieren und durchführen? Überlegen Sie sich ein paar Möglichkeiten, bevor Sie den Rest dieses Kapitels lesen.

### 1.1.1 Der prädiktive Ansatz

Wahrscheinlich haben Sie schon ein paar Möglichkeiten im Kopf, die alle zu einer der zwei generischen Formen gehören, die wir im Folgenden erörtern werden. Diese Optionen bezeichnet man übrigens auch als **Lebenszyklus der Entwicklung oder Entwicklungsverfahren**.

Die Abbildung unten zeigt einen generischen Lebenszyklus der Entwicklung.

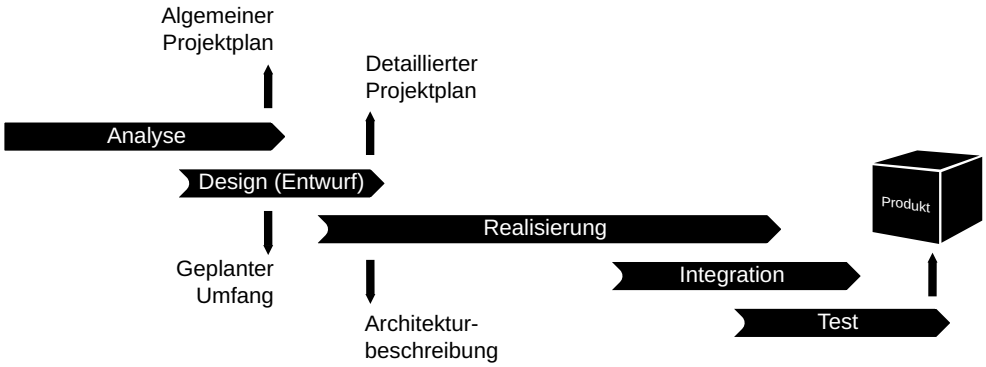



In diesem Lebenszyklus wird jeder Prozess abgeschlossen, bevor man zum nächsten Prozess übergeht:

1. Zuerst werden alle Anforderungen vollständig analysiert und festgelegt, was die Lösung beinhalten muss.
2. Danach entwirft man die Architektur der Lösung und untersucht, wie sich die Features am besten gestalten lassen.

3. Im Anschluss daran beginnen die Programmierer mit der Realisierung der Einheiten.
4. Die Einheiten werden dann in einer Lösung zusammengeführt.
5. Abschließend wird die gesamte Lösung getestet und eventuelle Fehler werden behoben.

Natürlich können sich diese Schritte auch überlappen. So kann beispielsweise mit der Integration und dem Testen begonnen werden, bevor alle Einheiten vollständig vorliegen. Ein solcher Lebenszyklus mit Überlappungen sieht dann wie folgt aus:




Dieser Lebenszyklus unterscheidet sich nicht grundlegend vom vorherigen Lebenszyklus, da auch in diesem Fall eine sequenzielle Abfolge von Entwicklungsprozessen vorliegt.

Grundvoraussetzung für diese Art von Lebenszyklus ist die anfängliche Analyse, die uns zeigt, was wir bauen müssen. Die Spezifikation, das Design und folglich auch der Plan (Entwurf) liegen bereits im Vorfeld vor. Man spricht daher auch von einer **plangesteuerten Entwicklung.** Darüber hinaus versuchen wir vorherzusagen bzw. zu prognostizieren, was wir benötigen und wie sich dies realisieren lässt. Deshalb spricht man auch häufig von **prädiktiver Entwicklung**.

Der prädiktive Lebenszyklusansatz ist bei vielen Projekten, die normale und geeignete Form der Projektentwicklung. So zum Beispiel bei Bauvorhaben. Zuerst erstellt man einen Rohplan oder -entwurf und erst danach folgen dann die optimierten und detaillierten Pläne und Entwürfe. Für andere Projekte dagegen ist diese Vorgehensweise nicht optimal. So zum Beispiel bei typischen Projekten in der IT-Entwicklung. Hier investiert man unter Umständen viel Zeit in die Spezifikation und in die Analyse der Anforderungen, auf denen dann alles weitere aufbaut. Und was passiert dann häufig? Die Kunden sind mit dem Ergebnis nicht zufrieden. Sie wünschen Veränderungen. Aber Veränderungen sind bei diesem Lebenszyklusansatz kostspielig, da möglicherweise alles, was bis zu diesem Zeitpunkt erstellt wurde, geändert werden muss.

In der IT-Branche gilt es als offenes Geheimnis, dass viele Kunden erst wissen, was sie wollen, wenn sie das Produkt sehen. Aber wann bekommen die Kunden das Produkt bei

einem prädiktiven Lebenszyklusansatz zu sehen? Richtig erst gegen Ende des Projekts und damit zu einem Zeitpunkt, an dem die Veränderungskosten am höchsten sind.

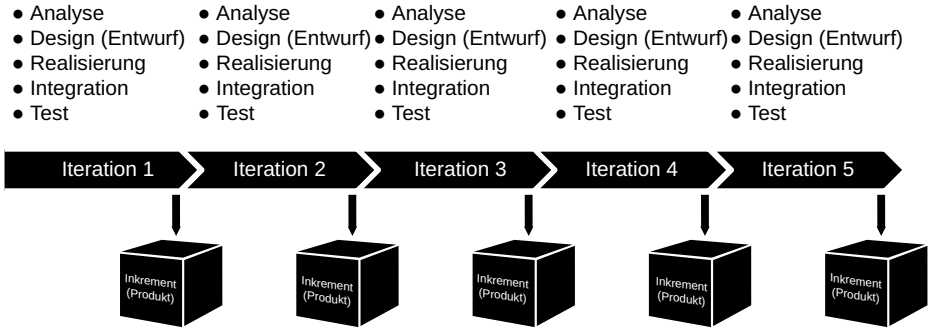
Die Agile Community bezeichnet prädiktive System in der Regel als **Wasserfallsysteme**. Diese Bezeichnung ist inzwischen jedoch negativ konnotiert und sollte daher nicht mehr verwendet werden, da dies in einer an sich rationalen Unterhaltung über verschiedene Entwicklungsansätze eine gewisse Voreingenommenheit zur Folge hätte.

### 1.1.2 Der adaptive Ansatz

Die Probleme, die prädiktive Lebenszyklen bei IT-Entwicklungsprojekten verursachen, lassen sich überwinden, indem wir den Komfort und die Struktur eines prädiktiven Systems zugunsten eines anderen Lebenszyklusansatz opfern, bei dem das Produkt inkrementell entwickelt und dabei immer wieder gemeinsam mit den Kunden und Usern (Benutzern und Benutzerinnen) überprüft wird. Diesen Luxus bietet uns die IT-Entwicklung im Gegensatz zu vielen anderen Bereichen. Denken Sie nur einmal an ein Bauprojekt. Bei einem Bauprojekt gibt es keine sinnvollen Inkremente und das Produkt ist erst am Ende des Projekts einsatzfähig.

Fairerweise muss hier jedoch gesagt werden, dass sie bei einem Bauvorhaben zwar keine Inkremente entwickeln können, dafür aber beim Bau eines Krankenhauses, ganz gleich wie viele Änderungen sie auch vornehmen, am Ende immer ein Krankenhaus entsteht und nicht zum Beispiel ein Freizeitpark. In der IT-Entwicklung dagegen, kann es durchaus vorkommen, dass sie im übertragenen Sinn mit dem Bau eines Krankenhauses beginnen und am Ende kommt eine Art Freizeitpark heraus.

Machen wir uns also mit einem Lebenszyklusansatz wie in der Abbildung unten dargestellt die Tatsache zu Nutze, dass bei IT-Entwicklungsprojekten Inkremente geliefert werden können




Bei diesem Lebenszyklus gibt es keine wirklichen Prognosen. Anstatt das Produkt detailliert zu planen (und sich auf den Plan oder Entwurf zu verlassen), werden in kurzen Iterationen Inkremente des Produkts erstellt. Jede Iteration konzentriert sich dabei auf ein paar vielversprechend erscheinende Features. Wir bauen ein Inkrement,

stellen es den Kunden und Usern vor, holen ihr Feedback dazu ein und entscheiden dann, was wir in der nächsten Iteration tun. Wir treffen also keine Vorhersagen mehr, sondern setzen das Projekt fort, indem wir es an das Feedback anpassen (adaptieren). Dies entspricht einem **adaptiven Lebenszyklusansatz.** „Agile" ist die populäre Bezeichnung für adaptive Systeme.

Um ein Inkrement zu erstellen, müssen alle Entwicklungsprozesse innerhalb eines bestimmten Zeitraums ausgeführt werden. Im nächsten Zeitraum wird das Ganze dann wiederholt bzw. iteriert. Daher bezeichnet man diese Zeitspannen auch als **Iterationen** und diese Art der Entwicklung als **iterative Entwicklung**. In der iterativen Entwicklung wird jeder Prozess (wie z. B. das Design) nicht nur einmal für das Produkt ausgeführt, sondern für verschiedene Elemente des Produkts wiederholt, d. h. mehrfach ausgeführt.

Die Entwicklung in Iterationen und die Lieferung in Inkrementen erfolgen in der Regel in Kombination.

#### 1.1.2.1 Iterationen mit festgelegtem Umfang versus Iterationen mit festgelegter Dauer

Was ist Ihrer Meinung nach vorzuziehen, Iterationen mit festgelegtem Umfang oder Iterationen mit festgelegter Dauer?

Theoretisch ist beides möglich, aber in der Praxis haben sich Iterationen mit festgelegter Dauer bewährt, denn bei Iterationen mit festem Umfang:

- verschwendet man möglicherweise zu viel Zeit auf einzelne Features und auf Schnickschnack. Bei Iterationen mit festgelegter Dauer dagegen ist man kontinuierlich angehalten, sich zuerst auf die Aspekte zu konzentrieren, die am meisten Wert bringen.
- benötigt man in der Regel mehr Zeit als erwartet, um den Umfang abzuschließen. Dies wiederum führt zu längeren Iterationen und reduziert die Zahl der Feedback-Schleifen. Weniger Feedback bedeutet aber auch weniger Adaption.

Daher nutzen fast alle Agilen Methoden Iterationen von fester Dauer, sogenannte **Timeboxes**, und bestehen meist auf deren Einhaltung. Eine Timebox ist eine Zeitspanne mit einer maximalen (oder festgelegten) Zeitdauer, die unter keinen Umständen verlängert wird (denn wenn man sie einmal verlängert, wird dies gerne zur Gewohnheit).

#### 1.1.2.2 Länge der Iterationen

Wir haben also festgestellt, dass Iterationen zeitlich beschränkt, d. h. timeboxed, sind. Welche Dauer sollte eine solche Iteration demnach haben?

Wir können zwar jederzeit Feedback einholen, aber maßgeblich ist das strukturierte Feedback am Ende jeder Iteration. Bei kürzeren Iterationen erhalten wir demnach häufiger strukturiertes Feedback und somit mehr Möglichkeiten zur Adaption. Andererseits sollte eine Iteration ausreichend Zeit bieten, um eine Reihe von Features zu erstellen, damit sich ein ernsthaftes Review gemeinsam mit dem Kunden lohnt. Mit anderen Worten, eine Iteration sollte auch nicht zu kurz sein.

In den Anfangszeiten der Agilen Systeme galten 4 bis 8 Wochen als ideale Länge für eine Iteration. Inzwischen haben sich jedoch kürzere Iterationen durchgesetzt. Heute gilt in den meisten Systemen eine maximale Länge von vier Wochen und sogar eine Länge von nur einer Woche erscheint angesichts der modernen Technologien als praktikabel.

#### 1.1.2.3 Gleiche oder unterschiedliche Längen?

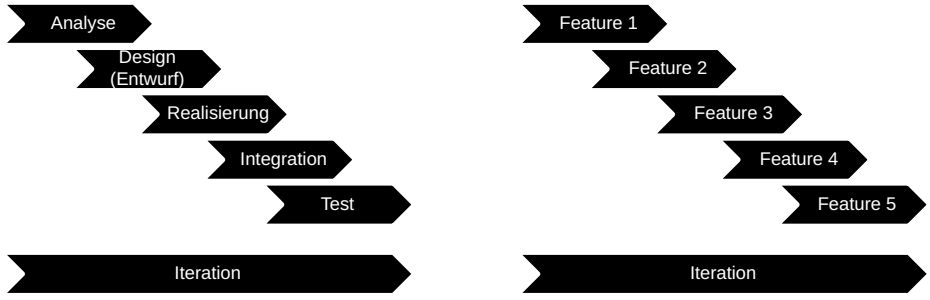
Was meinen Sie? Ist es besser, wenn alle Iterationen gleich lang sind, oder sollte die Länge flexibel sein?

Iterationen von gleicher Länge sorgen für mehr Disziplin und Regelmäßigkeit. In den meisten Fällen ist es nicht erforderlich, die Dauer jeder einzelnen Timebox separat festzulegen. Deshalb setzen die meisten Systeme für alle Iterationen die gleiche Länge an. In diesem Fall kann man die timeboxed (zeitbegrenzte) Länge einer Iteration ändern, ohne die Länge jeder einzelnen Iteration separat festlegen zu müssen.

#### 1.1.2.4 Was passiert in den Iterationen?

Eine Iteration ist eine Zeitspanne, in der wir die Entwicklungsprozesse wiederholen. Aber wie machen wir das?

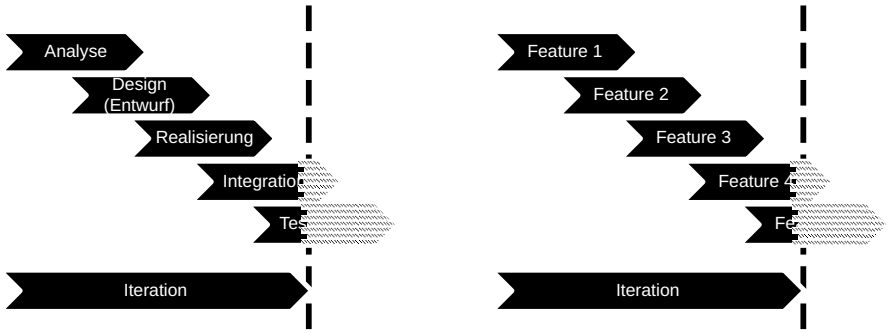
Die Abbildung unten zeigt zwei Möglichkeiten:




Bei der Möglichkeit links konzentrieren wir uns auf die einzelnen Entwicklungsprozesse und führen diese für alle Features einer Iteration durch. Man könnte von einem leicht prädiktiven Lebenszyklusansatz sprechen.

Bei der Möglichkeit rechts konzentriert man sich auf ein Feature oder mehrere Features und führt für diese(s) alle Entwicklungsprozesse durch. Dies könnte man als fast nicht prädiktiven Ansatz bezeichnen.

Wir bevorzugen die zweite, Möglichkeit, die auf Features basiert, weil sie mit den zeitlich begrenzten (timeboxed) Iterationen kompatibel ist.




Ist die maximale Länge einer Iteration begrenzt, kann es vorkommen, dass am Ende der Iteration nicht alles fertiggestellt ist. Bei dem auf Features basierenden Ansatz bedeutet dies, dass ein paar Features nicht fertiggestellt sind. Bei dem anderen Ansatz hingegen ist bei allen Features mindestens ein Entwicklungsprozess nicht abgeschlossen, sodass am Ende der Iteration kein brauchbares Ergebnis vorliegt, das wir vorstellen und für das wir Feedback einholen können.

#### 1.1.2.5 Inkrement versus Ergebnis

Jedes Inkrement ist ein Ergebnis, aber nicht jedes Ergebnis ist ein Inkrement.

Mit Inkrement bezeichnen wir die Inkremente eines Produkts. Bei der IT-Entwicklung sind dies die verschiedenen Versionen einer funktionierenden Software. Jedes neue Inkrement ist eine einsatzfähige Version desselben Produkts mit mehr Features. Die Einsatzfähigkeit muss sichergestellt sein, um zuverlässiges Feedback zu ermöglichen.

Ein Ergebnis dagegen kann so gut wie alles sein, was im Rahmen eines Projekts erstellt wird. In einem prädiktiven Projekt beispielsweise sind der vorläufige Entwurf und der vorläufige Plan zwar Ergebnisse, aber keine Inkremente des Produkts.

Da Agiles Vorgehen in Mode ist, bezeichnen manche ihre Ergebnisse als Inkremente und behaupten dann, sie seien Agile.

#### 1.1.2.6 Iterationen versus Zyklen

Jede Iteration ist ein Zyklus, aber nicht jeder Zyklus ist eine Iteration.

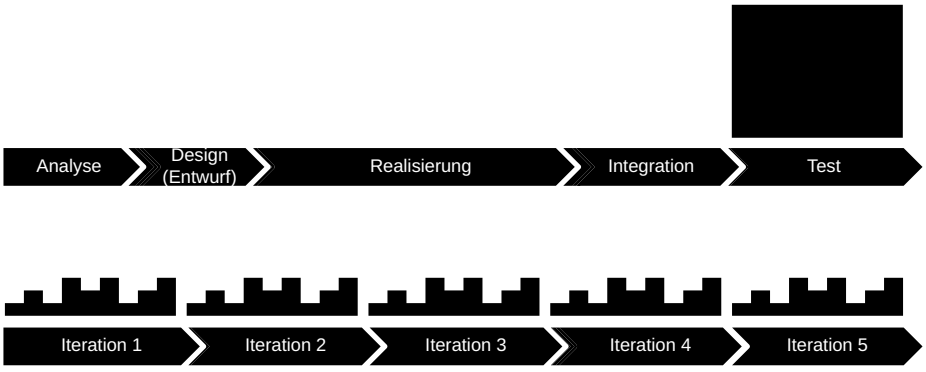
Eine Iteration ist eine besondere Art von Zyklus, in dem sich sowohl **Entwicklungsprozesse** als auch **Managementprozesse** wiederholen. Viele Systeme bestehen aus Zyklen, in denen jedoch nur die Management- und nicht auch die Entwicklungsprozesse wiederholt werden. Beispiele hierfür sind der große monatliche Zyklus und der kleine wöchentliche Zyklus in P3.express, die Phasen bei PRINCE2®, und die Phasen im PMBOK®-Guide.

Arbeiten wir den Unterschied heraus. Stellen Sie sich einen Zyklus vor, der über eine eigene Planung, Überwachung sowie Steuerung und einen eigenen Abschluss verfügt. Die Tatsache, dass diese Managementprozesse wiederholt werden, ist der Grund für die Bezeichnung als Zyklus. Stellen Sie sich nun einmal vor, es handele sich um ein prädiktives Projekt. In einem Zyklus geht es um die Spezifikation der Anforderungen, im nächsten um den Entwurf des Produkts und so weiter. Dies wäre ein zyklisches System ohne Iterationen.

Viele denken nun, dass sie, sobald erkennbare Zyklen vorliegen, ihre Projekte als iterative und daher Agile Projekte bezeichnen könnten. Das ist jedoch nicht korrekt. Noch verwerflicher als Managementprozesse mit Entwicklungsprozessen zu verwechseln ist jedoch, wenn einfach irgendwelche Zeitspannen in Projekten als Iterationen bezeichnet werden, wie z. B. wöchentliche Iterationen, obwohl in diesen Zeiträumen überhaupt keine Prozessiteration stattfindet.

#### 1.1.2.7 Testen und Qualität bei Agilen Modellen

Das nachfolgende Diagramm zeigt stark vereinfacht, wie das Testen in den jeweiligen Lebenszyklusansätzen erfolgt.




Die Testaktivitäten erfolgen meist am Ende eines prädiktiven Projekts, d.h. wenn wir wahrscheinlich schon etwas in Verzug und unter großem Druck sind, das Projekt schnellstmöglich abzuschließen. Dieser Druck kann dazu führen, dass Tests ausgelassen und bei der Qualität Kompromisse gemacht werden.

Wie sieht es hier bei den adaptiven Systemen aus?

Nun, bei adaptiven Lebenszyklen existiert dieses Problem nicht, da kontinuierlich getestet wird. Es ist vollkommen gleichgültig, wann wir das Projekt beenden, wir haben stets das genau richtige Maß an Tests durchgeführt.

Natürlich gibt es auch andere Unterschiede. So sind automatisierte Tests aufgrund des Wesens von adaptiven Systemen quasi unumgänglich. Zwar decken automatisierte Tests möglicherweise nicht jede einzelne Zeile des entwickelten Codes ab, aber es gibt eine optimale **Codeabdeckung**, die wir in einem Projekt erfüllen müssen. Mit Codeabdeckung bezeichnen wir das Verhältnis von Codezeilen, die in automatisierten Tests überprüft werden, zur Gesamtzahl an Zeilen.

## 1.2 Die Wahl des Vorgehens bei der Entwicklung

Prädiktive und adaptive Lebenszyklen haben jeweils Vor- und Nachteile. Die richtige Wahl hängt von vielen verschiedenen Faktoren ab. Der wichtigste Faktor ist dabei jedoch die Art des Produkts.

Bevor Sie sich entscheiden, welchen Lebenszyklus Sie für Ihr Produkt brauchen, sollten Sie sich zwei Fragen stellen:

1. **Muss das Produkt adaptiv sein?** Benötigen Sie kein adaptives Produkt, dann ist ein prädiktiver Lebenszyklus wahrscheinlich einfacher, strukturierter und *planbarer*. Ein adaptives System ist immer dann erforderlich, wenn ein gewisses Risiko besteht, dass Sie mit der Idee beginnen, ein Krankenhaus zu bauen und letztendlich ein Freizeitpark herauskommt.
2. **Kann das Produkt adaptiv sein?** Diese Frage ist vielleicht noch wichtiger als die vorherige Frage. Sie können nur adaptiv vorgehen, wenn Sie die Möglichkeit haben, in Iterationen zu entwickeln und Inkremente bereitzustellen, um Feedback einzuholen und Ihr Produkt entsprechend anzupassen. Denken wir noch einmal an ein Bauvorhaben: Kann man ein Gebäude in Iterationen entwerfen? Können Sie beispielsweise das Fundament eines Gebäudes entwerfen, ohne das restliche Gebäude zu planen und zu wissen, welche Last auf dem Fundament ruhen wird? Die Antwort ist ein klares „Nein“. Eine Entwicklung in Iterationen (entsprechend unserer Definition) ist bei einem Bauvorhaben nicht möglich. Auch die Bereitstellung in Inkrementen ist in den meisten Fällen nicht möglich, weil die Einzelteile eines Gebäudes einerseits nicht einsatzbereit sind und das zu einem Einzelteil generierte Feedback andererseits vielleicht nicht auf das restliche Gebäude anwendbar ist. Ein adaptiver Lebenszyklus lässt sich also nicht auf den Bau eines Gebäudes anwenden. (Aber Achtung dies ist nicht mit der Innenausstattung, Gestaltung oder auch Renovierung eines Gebäudes zu verwechseln, wo adaptive Systeme durchaus möglich sind).

Die wichtigste Botschaft in diesem Zusammenhang lautet, dass die Entscheidung zwischen einem prädiktiven und einem adaptiven Ansatz keine Entscheidung zwischen Gut und Böse ist, sondern von vielen Faktoren abhängt. Beides sind valide Vorgehensweisen, die sich jeweils für eine bestimmte Art von Produkten eignen.

Stellen Sie sich zur Übung folgende IT-Projekte vor: Ein Projekt, bei dem in einer Organisation die Betriebssysteme von 300 Computern aktualisiert werden müssen und ein Projekt, bei dem für eine sehr große Organisation mit Niederlassungen an sechs Standorten eine Netzwerkinfrastruktur zu erstellen ist. Welche Art von Lebenszyklus der IT-Entwicklung eignet sich Ihrer Meinung nach besser für diese beiden Projekte?

## 1.3 Eignet sich Agile nur für Entwicklungsprojekte in der IT?

Die meisten Beispiele in diesem Buch und in anderen Materialien über Agile stammen aus Entwicklungsprojekten in der IT. Bedeutet dies, dass Agile Methoden auf IT-Entwicklungsprojekte beschränkt sind?

### 1.3.1 Projekte

Es gibt Leute, die behaupten, eine Agile Vorgehensweise eigne sich für jede Art von Projekt. Diese Leute behaupten in der Regel auch, es gebe nur eine korrekte Form der Projektdurchführung. In der Regel handelt es sich dabei um Menschen, die bislang nur Erfahrung mit unkritischen IT-Entwicklungsprojekten haben. Denn tatsächlich gibt es viele Arten von Projekten, bei denen ein adaptives Vorgehen entweder nicht nötig oder nicht möglich ist, weil wir sie nicht in Iterationen entwickeln und in Inkrementen liefern können.

Abgesehen von der simplen Tatsache, dass Agile nicht die einzig wahre Lösung ist und nicht in jedem Projekt genutzt werden kann, können wir uns trotzdem anschauen, welche Projekte von einem adaptiven System profitieren können. Sind diese Projekte auf die IT-Entwicklung beschränkt oder gibt es weitere geeignete Projektarten?
Es mag zwar möglich sein, Agile in anderen Projekttypen zu verwenden, aber dies bedarf professioneller und strukturierter Bemühungen, die bislang anscheinend noch nicht unternommen wurden. Zwar gibt es außerhalb der IT einige Projekte, die behaupten Agile zu sein, aber in der Regel legen diese den Begriff Agilität falsch aus und sind Opfer des sogenannten **Cargo-Kult-**Effekts, d. h. sie ahmen ein Verhalten nach, ohne den eigentlichen Sinn zu verstehen. Dessen ungeachtet eignen sich Projekte im Bereich der IT-Entwicklung wahrscheinlich nach wie vor am besten für ein adaptives Vorgehen.

### 1.3.2 Programme

Bei allen Ausführungen bisher ging es um **Projekte**. Bei **Programmen** gilt jedoch etwas anderes. Laut MSP®, einer Programm-Management-Methode aus der gleichen

Familie wie PRINCE2® und ITIL®, können Projekte sowohl adaptiv als auch prädiktiv sein, wohingegen Programme stets adaptiv sein müssen. Der Grund hierfür ist, dass es bei Projekten um Produkte und bei Programmen um Ergebnisse geht. Wir können zwar vorhersagen, wie Produkte zu erstellen sind, nicht aber wie Ergebnisse erzielt werden können.

### 1.3.3 Betrieb

Projektmanagementmethoden beginnen immer mit der Definition, was ein Projekt ist. Der Grund hierfür ist, dass diese Methoden nur für Projekte und nicht für Programme, das Portfolio oder alltägliche Geschäft (den Betrieb) gelten. Bei Agilen Systemen dagegen ist dies nie zur Tradition geworden. Sie bestehen nicht darauf, in Projekten eingesetzt zu werden und wurden durchaus schon manchmal im Betrieb eingesetzt. Dies geht zurück auf die IT-Entwicklung, in der die Grenzen zwischen Projekten mit größeren Veränderungen und Betrieb, in dem kleinere Veränderungen auf das Produkt angewendet werden, verschwimmen. In seiner extremen Form zeigt sich dies in der Philosophie von DevOps, bei dem die Projektseite (Entwicklung) mit dem Alltagsgeschäft (Betrieb) verschmilzt.

## 1.4 Ist Agile schneller?

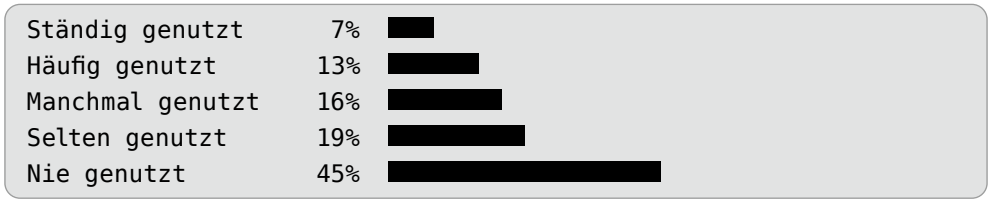
Die Bezeichnung „Agile" bzw. „agil" impliziert, dass diese Methoden schneller sind. Diese Hypothese zu bestätigen oder zu widerlegen ist äußerst schwierig, aber es gibt ein Konzept, das bei Agilen Projekten wirklich hilft. Dabei geht es jedoch nicht um die Geschwindigkeit, in der wir entwickeln, sondern um die Features, die wir entwickeln müssen (den Umfang).

Denken Sie an ein IT-Projekt, das mit Hilfe einer prädiktiven Methode entwickelt werden soll. In diesem Fall wären ein oder ein paar Kundenvertreter:innen dafür verantwortlich, die Anforderungen zu identifizieren und zu kommunizieren. Ihnen ist klar, dass Anforderungen zu übersehen kostspielig und die nachträgliche Ergänzung von Anforderungen aufwändig ist. Die Kundenvertreter:innen neigen daher dazu, übermäßig kreativ zu werden und Anforderungen zu ergänzen, die keine ausreichende Wertschöpfung bieten. Diese Extra-Features erfordern mehr Zeit und Ressourcen und machen das Produkt komplexer, was für die künftige Instandhaltung und künftigen Erweiterungen ein ernsthaftes Problem darstellen kann.

In einem adaptiven System dagegen sind die Kundenvertreter:innen nicht gezwungen, alle Anforderungen von vornherein festzulegen. Damit sinkt die Wahrscheinlichkeit, dass seltsame Anforderungen ergänzt werden. Sollte es doch solche Anforderungen geben, so hilft ein ordentliches adaptives Entwicklungssystem den Kundenvertretern und -vertreterinnen zumindest, deren Wert zu verstehen, so dass sie diese erst am Schluss realisieren oder ganz weglassen können.

Ein ordnungsgemäß durchgeführtes Agiles Projekt bietet in der Praxis die Chance auf einen kleineren Umfang. Damit wird das Projekt schneller und weniger kompliziert.

Die Standish Group beispielsweise meldete 2002 bezüglich vier interner Applikationen folgende Zahlen zur Nutzung der Features:




Stellen Sie sich jetzt einmal vor, wie viel schneller die Projekte und wie viel einfacher die Produkte hätten sein können ohne diese nie oder nur selten genutzten Features. Zwar ist dies nur ein Beispiel von einigen wenigen Applikationen in einem Unternehmen, aber der Trend insgesamt ist wahrscheinlich gar nicht so viel anders.

## 1.5 Ist Agile etwas Neues?

Die Agile Methode wird in der Regel als neue Vorgehensweise propagiert. Was sicherlich neu ist, ist die Bezeichnung „Agile" für adaptive Lebenszyklen. Wie aber sieht es mit dem Lebenszyklus selbst aus?

Es ist nur schwer vorstellbar, dass die Menschheit in ihrer langen Geschichte, mit ihren vielen Projekten und Programmen, ganz ohne adaptive Lebenszyklen ausgekommen ist. Denken Sie nur einmal an eine früher weit verbreitete Unternehmung bzw. ein durchaus populäres Projekt oder Programm: Krieg führen. Lässt sich mit einem prädiktiven Ansatz ein Krieg führen. Lässt sich alles von Anfang an planen und festlegen? Ganz bestimmt nicht. Es gibt zwar möglicherweise einen übergeordneten Plan (wobei es sich wahrscheinlicher eher um eine Strategie als einen Plan handelt), aber den Krieg an sich müssen Sie Schlacht um Schlacht (d. h. Iteration um Iteration) führen und die weitere Kriegsführung jeweils an den Ausgang der einzelnen Schlachten anpassen. Es ist kein „nettes" Beispiel, aber eines das sehr deutlich zeigt, dass adaptive Lebenszyklen gar nicht so neu sind.

Was also ist neu? Neu sind hauptsächlich die Nutzung adaptiver Systeme in der IT-Entwicklung und die Bezeichnung „Agile". Früher waren IT-Entwicklungsprojekte ganz anders. Sie forderten eine präzise prädiktive Methode. Im Laufe der Zeit veränderten sich dann jedoch das Wesen und das Publikum dieser Projekte. Prädiktive Systeme stellten zwar meist nicht mehr die beste Wahl dar, aber die Experten und Expertinnen setzten sie weiterhin ein, bis die adaptive Methode schließlich von einer Gruppe von Experten, die in diese Projekte involviert war, neu erfunden wurde.