

# 1 Einführung

*Großer Stein ist schwer zu werfen.*

— Deutsches Sprichwort

Stellen Sie sich vor, Sie müssen ein Programm entwickeln, das Mitarbeiter von Finanzdienstleistern beim Handeln mit Finanztiteln unterstützt. Die Händler müssen in der Lage sein, die Produkte zu kaufen, zu verkaufen und dabei die dazugehörigen Risiken einzukalkulieren. Ein Problem dabei ist, dass die Menge möglicher Produkte häufigen Änderungen unterliegt. Schaut man sich nur mal den Optionsmarkt an (der z.B. mit Derivaten handelt), kann man beobachten, dass mit jedem neuen Tag neue Arten von Optionen auftreten, die unterschiedlich gehandelt werden und deren Risiken auf eine andere Art bewertet werden müssen. Fragt man den Händler deshalb heute nach seinen Anforderungen an das Programm, wird man sicherlich eine andere Antwort erhalten, als wenn man morgen fragt. Hinzu kommt, dass der Händler sagen wird, dass jeder Tag, an dem eine neue Option nicht mit dem Programm gehandelt werden kann, einen finanziellen Verlust darstellt, der sich auch auf die Provision des Händlers auswirkt und für den Finanzdienstleister als Ganzes durchaus eine Größenordnung von mehreren 100.000 Euro betragen kann. Natürlich wird der Händler auch darauf hinweisen, dass das Konkurrenzprogramm diese Option schon lange handeln kann, was die Sache nicht besser macht. Auf der anderen Seite wird nicht unbedingt eine vollständige Unterstützung des Handels mit neuen Optionen verlangt. Jede Verbesserung, die manuelle Arbeit abnimmt, hilft. Natürlich wäre eine teilweise Unterstützung nicht optimal und eine vollständige Unterstützung auf jeden Fall anzustreben.

Entsprechende Anfragen können im Bereich der Telekommunikation oder auch in allen Bereichen des E-Business-Umfelds auftreten. Der große Unterschied zu den bisher traditionell geforderten Programmen ist dabei, dass es immer wichtiger wird, *schnell* mit passenden Lösungen auf dem Markt zu sein. Ansonsten besteht die Gefahr, dass das Programm schon bei der Fertigstellung veraltet ist oder die Firma des Kunden schon gar nicht mehr existiert, weil das Programm zu spät kam. Manchmal ist es sogar wichtiger, die *wichtigsten* Anforderungen des Kunden zunächst einmal mit einer ersten Version *schnell* abzudecken, als später (oder zu spät) mit einer Version für alle Anforderungen herauszukommen.

Schwergewichtige Prozesse aus den achtziger und neunziger Jahren haben Schwierigkeiten mit derartigen neuen Anforderungen umzugehen. Wenn überhaupt, dann waren sie gelegentlich in Bereichen mit *stabilen* Anforderungen erfolgreich. In solchen Bereichen kann von Anfang an alles formalisiert und ein

detaillierter Plan erstellt werden. Darüber hinaus können Folgeaktionen »blind« dem Plan folgen und am Ende des Projekts ist der Plan immer noch korrekt. Beispiele für derartige Projekte sind Projekte im militärischen Bereich oder in vergleichbaren Bereichen wie dem Flugzeugbau oder bei Kraftwerken. Neben stabilen Anforderungen zeichnen sich solche Projekte außerdem noch durch scheinbar unbegrenzte Mengen von Zeit und Geld aus, was bedeutet, dass es wichtiger ist, *alle* Anforderungen zu erfüllen, als nur einen Teil in einem gewissen zeitlichen oder finanziellen Rahmen zu liefern. Es zeichnet sich allerdings ab, dass man sich das mittlerweile auch in diesen Bereichen nur noch selten leisten kann. So werden Prozesse, die Änderungen bei den Anforderungen unterstützen, auch im militärischen Bereich immer wichtiger.

Agile Prozesse versprechen, flexibel auf sich immer wieder ändernde Anforderungen zu reagieren. Aus diesem Grund werden agile Prozesse inzwischen als Allheilmittel für erfolgreiche Softwareentwicklung betrachtet. Fast immer werden agile Prozesse dabei allerdings nur für kleine Teams und kleine Projekte empfohlen, womit den großen Projekten und großen Teams, die auch auf schnelle Änderungen der Anforderungen reagieren müssen, nicht wirklich geholfen ist.

Dieses Buch beschäftigt sich daher mit agilen Prozessen in großen Projekten. Doch bevor wir in das Thema detailliert einsteigen, möchte ich den Fokus und die Zielgruppe des Buchs noch genauer definieren. Dazu ist es zunächst notwendig, Begriffe wie *groß*, *agil* und *agiler Prozess* zu präzisieren und im Kontext dieses Buchs abzugrenzen.

## Große Projekte

Softwareentwickler stellen Softwareentwicklung im Großen grundsätzlich in Frage. Dies liegt nicht nur daran, dass die meisten der agilen Prozesse im Wesentlichen nur auf kleine Teams beschränkt sind, sondern auch daran, dass die meisten fehlgeschlagenen Projekte richtig große Projekte sind. (Es kann natürlich auch sein, dass man über die vielen fehlgeschlagenen kleinen Projekte gar nicht spricht, da sie nicht wichtig genug sind.)

Der häufigste Grund, weshalb große Projekte fehlschlagen, ist der Mangel an Kommunikation: Kommunikation zwischen den Teammitgliedern, zwischen dem Team und den Managern, zwischen dem Team und dem Kunden und so weiter. Kommunikation ist jedoch eines der wichtigsten Elemente agiler Prozesse. Somit stellt sich die Frage, ob Kommunikation überhaupt grundsätzlich erfolgreich in großen Teams etabliert werden kann. Dies ist ein Aspekt, der dazu führt, dass man bei großen Teams gern die Empfehlung gibt, diese »gesund« zu schrumpfen: »Wenn du in deinem Team hundert Entwickler hast, werfe mindestens achtzig raus und arbeite mit den restlichen zwanzig (oder weniger) weiter. Damit steigen die Erfolgsaussichten für dein Projekt signifikant.«

Große Projekte lassen sich aber nicht grundsätzlich vermeiden. Es kann Umstände geben, die dazu führen, dass man ein großes Projekt mit einem großen Team durchführen muss. So gibt es zum Beispiel Projekte, die einen derartig großen Umfang haben, dass dieser im geplanten Zeitraum unmöglich mit einem kleinen Team realisiert werden kann.

Will man nun von den Vorteilen agiler Prozesse profitieren, stellen sich einige Fragen: Können agile Prozesse skaliert, das heißt vergrößert oder aufgestockt werden, so dass sie große Projekte unterstützen? Und welche Probleme treten auf, wenn ein Unternehmen sich entscheidet, einen agilen Prozess in einem großen und möglicherweise kritischen Projekt einzusetzen? Dieses Buch beantwortet diese und viele andere Fragen in diesem Umfeld. Bevor wir ins Detail gehen, möchte ich allerdings noch klären, was mit »großen« Projekten eigentlich gemeint ist.

## Was bedeutet »groß«?

Man kann ein Projekt in verschiedener Hinsicht als *groß* betrachten. So können zum Beispiel die Kosten, der Umfang, die Dauer, die Anzahl der beteiligten Personen und die Risiken groß sein. Diese verschiedenen »Dimensionen« von Größe sind nicht unabhängig voneinander. Manche Dimensionen sind primär durch die Anforderungen und Randbedingungen vorhanden. Andere sind sekundär aus den primären abgeleitet.

Die einzelnen Dimensionen von Größe und deren Abhängigkeiten stellen sich wie folgt dar:

- ❑ **Umfang:** Der Umfang ist eine primäre Dimension von Größe, die sich unmittelbar aus der Menge und Komplexität der Anforderungen ergibt. Hat ein Projekt einen großen Umfang, kann man damit unterschiedlich umgehen: Man kann die Zeitdauer und/oder die Anzahl der Mitarbeiter des Projekts erhöhen. Das bedeutet, dass die Dimension Umfang die Dimensionen Dauer und Anzahl der Personen beeinflusst.
- ❑ **Dauer:** Die Dauer ist eine typische sekundäre Dimension. Kein Projekt wird mit dem Selbstzweck gestartet, 20 Jahre zu dauern und dabei zuzusehen, wie die Zeit vergeht. Wenn ein Projekt lange dauert, ist das immer eine Folge davon, dass eine andere Dimension angewachsen ist. Wenn z.B. das Risiko eines Projekts groß ist, weil viele unerfahrene Leute eingesetzt werden, kann als Folge von umfangreichen Schulungsmaßnahmen und Fehlschlägen die Dauer eines Projekts sehr groß werden. Manche Projekte können mitunter sogar unendlich lange dauern, weil keiner sich mehr traut, die Projekte abubrechen.
- ❑ **Kosten:** Kosten sind ebenfalls eine sekundäre Dimension; das heißt, hohe Kosten sind ebenfalls eine Konsequenz aus dem Wachstum anderer Dimensionen. Ich habe zumindest noch kein Projekt erlebt, das nur gestartet wurde,

weil man einfach zu viel Geld hatte und dieses Geld los werden wollte. Ich habe allerdings schon oft gesehen, dass Unsummen ausgegeben wurden, ohne dies wirklich zu hinterfragen. Dies war aber immer eine Folge einer der anderen Dimensionen. So entstehen nicht selten unnötige Kosten durch den Einsatz von vielen Personen, ohne dass unbedingt hinterfragt wurde, ob die Anzahl an Personen überhaupt notwendig ist.

- **Personenanzahl:** Die Anzahl an Mitarbeitern ist ein anderer Fall. Zum einen kann es sich um eine Konsequenz eines großen Umfangs oder Risikos handeln. Manchmal werden Projekte aber auch einfach nur mit vielen Mitarbeitern ausgestattet, um die Wichtigkeit des Projekts oder des Projektmanagers auszudrücken. Im schlimmsten Fall geschieht dies sogar gleich von Anfang an. Die Anzahl der Projektmitarbeiter bezieht sich dabei nicht nur auf die Anzahl der an der Entwicklung direkt beteiligten Personen, sondern z.B. auch auf die Anzahl der Kunden. Je mehr Kunden am Projekt beteiligt sind, desto höher sind zum Beispiel die Auswirkungen von widersprüchlichen Anforderungen.
- **Risiken:** Risiken sind viel komplizierter als die anderen Dimensionen, da sie sich auf so ziemlich alles beziehen können. Die Teamgröße selbst kann zum Beispiel schon ein Risiko sein. Auch das Setzen auf eine neue Technologie birgt ein großes Risiko. Als Folge von großen Risiken werden häufig auch die Kosten vergrößert, um z.B. die Mitarbeiter zusätzlich zu schulen und zu beraten. Trotzdem sind Risiken immer eine sekundäre Dimension.

Aus dieser Aufstellung folgt, dass die einzigen initialen Merkmale zur Skalierung eines Projekts der Umfang und die Anzahl der Personen sind. Ein Projekt mit großem Umfang kann man zwar mit einem kleinen Team durchführen, doch vor allem in großen Firmen wird ein Projekt mit großem Umfang fast immer von vielen Personen durchgeführt.

Arbeitet in einem Projekt eine große Anzahl von Personen, so haben auch in der Regel alle anderen Dimensionen eine entsprechende Größe. Man wird schwerlich ein Projekt mit einem großen Team finden, das einen kleinen Umfang hat, nur drei Monate dauert und für das dabei nur einige hunderttausend Euro ausgegeben werden. Das Projekt mag nicht unbedingt ad hoc ein großes Risiko in sich bergen, doch ist die Größe der anderen Dimensionen ein Risiko an sich. Ist zum Beispiel der Umfang groß und ändert sich dieser auch noch (in Form von sich ändernden Anforderungen), so ist das Risiko hoch, dass man die Funktionalität nicht in den Griff bekommt. Ist zum Beispiel der Zeitrahmen groß, besteht ein hohes Risiko, dass das Projekt niemals beendet werden wird, da das Projektende über lange Zeit nicht in Sicht ist.

Ich werde mich in diesem Buch vor allem auf den Aspekt großer Teams beziehen. Da große Teams in der Regel auch Umfang, Dauer, Kosten und Risiken vergrößern, werden damit alle hier genannten Dimensionen von Größe eine Rolle spielen.

## Warum spielt Größe eine Rolle?

»Groß« ist ein relativer Begriff und dies gilt auch bei der Teamgröße. Sprechen wir über 2, 10, 100, 1.000 oder sogar noch mehr Personen? Dies ist insofern wichtig, da jede neue Größenordnung einen anderen Einfluss auf den Prozess hat. Man betrachte z.B. den Einfluss der Teamgröße auf die Kommunikation:

- ❑ **Bei 2 oder mehr Personen** müssen die Personen anfangen miteinander zu kommunizieren. Solange ein Projekt von nur einer Person durchgeführt wird, kennt diese Person (hoffentlich) alle Aspekte, die zur Durchführung des Projekts notwendig sind. Sie kennt das Design und den Code des gesamten Systems. Sobald eine zweite Person hinzukommt, ist dies nicht mehr automatisch sichergestellt. Die Personen müssen kommunizieren. Nur so können sie ein gemeinsames Verständnis darüber entwickeln, was gerade vorgeht und was für den gemeinsamen Fortschritt notwendig ist. Es wäre zum Beispiel ziemlich ungünstig, wenn beide dasselbe Teilproblem unabhängig (also nicht durch paarweises Programmieren) lösen und dies erst dann feststellen, wenn sie versuchen, ihren Code zusammenzuführen.
- ❑ **Bei 10 oder mehr Personen** muss man anfangen, nicht nur den Fortschritt, sondern auch die dazugehörige Kommunikation explizit zu koordinieren. Man muss Kommunikationskanäle extra etablieren, um Themen mit der ganzen Gruppe diskutieren zu können.
- ❑ **Bei 100 und mehr Personen** ist es selbst mit einem Großraumbüro nicht mehr möglich, alle Personen in einem Raum unterzubringen. Dadurch muss die »natürliche« Kommunikation, die normalerweise innerhalb eines Raums auftritt, durch andere Maßnahmen sichergestellt werden.
- ❑ **Bei 1.000 und mehr Personen** ist die Wahrscheinlichkeit groß, dass sich Teams nicht nur über mehrere Räume, sondern sogar über mehrere Gebäude oder Standorte verteilen. Aus diesem Grund kennen sich die einzelnen Teammitglieder untereinander oft gar nicht mehr.

Dieses Beispiel zeigt nicht nur, dass groß relativ ist, sondern auch, dass jede »Vergrößerung« spezielle Auswirkungen haben kann.

## Worum geht es in diesem Buch?

Der Inhalt dieses Buchs basiert auf Erfahrungen, die ich in Projekten gesammelt habe, die eine Größenordnung von einer bis 200 Personen umfassten. Durch die Arbeit mit diesen unterschiedlich großen Teams habe ich viele Vorgehensweisen entdeckt, die die Skalierung von agilen Prozessen für große Teams ermöglichen.

Will man nun agil vorgehen, gibt es nach meiner Erfahrung bereits signifikante Auswirkungen auf ein Team, das aus mehr als 20 Personen besteht. Allerdings werden auch kleine Teams mit z.B. 10 Personen definitiv von diesem

Buch profitieren, speziell dann, wenn sie in ein großes Unternehmen eingebettet sind. Der Schwerpunkt dieses Buchs liegt aber bei noch größeren Projekten, die von Teams mit etwa 30 bis 200 Personen und eigentlich immer in großen Unternehmen durchgeführt werden.

Die speziellen Aspekte von noch größeren Teams, also Teams mit mehr als 1.000 Personen, werden mangels Erfahrung nicht explizit untersucht. Allerdings gibt es auch dort Aspekte, die in diesem Buch behandelt werden und deren Kenntnis bei der erfolgreichen Abwicklung derartiger Projekte signifikant weiterhelfen können. Genauso können auch kleine Teams von vielen Diskussionen in diesem Buch profitieren. Das Buch hilft das agile Wertesystem zu verstehen und zeigt dabei Wege auf, wie dieses Wertesystem auch in großen Projekten mittels des verwendeten Prozesses bewahrt werden kann. Dabei wird auch der Unterschied zwischen dem agilen Wertesystem an sich und dessen Umsetzung in einem bestimmten Prozess, wie z.B. *Extreme Programming*, deutlich.

Meine Erfahrungen basieren im Wesentlichen auf Teams, bei denen ein Großteil der Mitarbeiter beieinander saß und nur wenig der Entwicklungsarbeit auf einen anderen Standort verlagert war. Aus diesem Grund ist das verteilte Arbeiten über mehrere Standorte kein ausgesprochener Schwerpunkt dieses Buchs, auch wenn das Thema in Kapitel 3 (ab Seite 64) und Kapitel 6 (ab Seite 170) behandelt wird.

Die fachlichen Bereiche dieser Projekte unterliegen keinen besonderen Einschränkungen. Die meisten Projekte wurden in den Bereichen Finanzdienstleistung/Banking, Automobilbau und Telekommunikation durchgeführt. Es gab aber auch Projekte in den Bereichen Hardware- und Softwaretechnologie-Entwicklung.

Ich habe meine Erfahrungen natürlich mit vielen anderen Personen ausgetauscht, wobei deren Erfahrungen meistens in der gleichen Größenordnung lagen. Einige Personen arbeiteten sogar in Projekten mit mehr als 350 Personen und berichteten dabei über die gleichen Herausforderungen. Ich weiß allerdings nicht, was in Projekten mit mehr als 1.000 Personen passiert, nehme aber an, dass dort einige neue Aspekte und Herausforderungen auftreten. Es bleibt aber festzuhalten, dass alle Überlegungen und Vorschläge in diesem Buch wirklich auf konkreten Erfahrungen mit großen Teams und großen Projekten beruhen. Dabei handelt es sich im Wesentlichen um meine Erfahrungen, die durch andere bestätigt und ergänzt wurden, was ich im Einzelfall explizit erwähne.

## **Abgrenzung zu agilen Prozessen**

Dieses Buch ist weder ein Buch über agile Prozesse im Allgemeinen, noch werden einzelne agile Prozesse im Speziellen diskutiert. Es ist auch kein Buch zum Thema *Skalierung von Extreme Programming*. Trotzdem werden Sie vermutlich einige Praktiken von *Extreme Programming* in diesem Buch entdecken. Dabei

handelt es sich um die Praktiken, die auch für große Teams skaliert werden können.

Da das Buch nicht von agilen Prozessen im Allgemeinen handelt, gibt es auch keine ausführliche Einführung in agile Prozesse. Ich liefere zwar am Anfang von Kapitel 2 eine sehr kurze Einführung in die grundlegenden Aspekte agiler Prozesse, für eine detaillierte Einführung in agile Prozesse sei aber auf [Cockburn02] verwiesen.

## Gesamtprozess versus Prozesse für Teilteams

Wie wir später diskutieren werden (siehe Kapitel 3), wird ein großes Team normalerweise in kleine Teilteams aufgeteilt. Da bereits etliches zu agilen Prozessen in kleinen Teams gesagt wurde, gehe ich nicht auf die Prozesse ein, die innerhalb dieser Teilteams verwendet werden. Ich konzentriere mich dagegen auf den Prozess, der all diese Teilteams zusammenführt und dafür sorgt, dass man trotz des Einsatzes von vielen Personen agil zusammenarbeitet. Es geht also weniger darum, alle Aspekte agiler Prozesse zu behandeln, die in großen Teams eine Rolle spielen, sondern viel mehr um eine Diskussion der Aspekte, die sich in großen Projekten und großen Teams anders darstellen.

Dass es diese Unterschiede tatsächlich gibt, wurde bereits am Beispiel Kommunikation erläutert: Auch für agile Prozesse gilt, dass diese nicht linear skalieren. Durch neue Größenordnungen können neue Schwierigkeiten auftreten, da einige Aspekte eines Prozesses ab einer gewissen Größe eine natürliche Grenze haben. Damit wird es notwendig, mit diesen Aspekten anders umzugehen. Hinzu kommen die Probleme, die erst mit großen Teams auftreten.

Zusammenfassend handelt es sich hier nicht um ein Buch, das einfach nur agile Prozesse skaliert. Es geht vielmehr darum, die besten Praktiken vorzustellen, die dabei helfen, die agilen Prinzipien so zu skalieren, dass die agilen Werte gewahrt bleiben.

## Wer sollte dieses Buch lesen?

Dieses Buch wendet sich an den so genannten *Change-Agent*. Damit bezeichne ich in diesem Kontext eine Person, die einen agilen Prozess in großen Teams trotz aller Schwierigkeiten aufbauen oder etablieren will. In einigen Kapiteln dieses Buchs (zum Beispiel im Kapitel 6, *Umgang mit dem Unternehmen*) kann es sich auch um einen *Change-Agent* handeln, der z.B. die Arbeit kleiner Teams oder die Arbeit von Teams in einer nicht agilen Umgebung verbessern will. Ich gehe davon aus, dass der *Change-Agent* mit agilen Prozessen im Allgemeinen oder einem der agilen Prozesse im Speziellen (wie *Extreme Programming*) bereits vertraut ist.

Darüber hinaus ist das Buch auf jeden Fall für Personen von Interesse, die

- ❑ bereits versucht haben, agile Methoden in großen Projekten einzusetzen, und denen dies nicht gelungen ist,
- ❑ bereits versucht haben, agile Methoden in großen Projekten einzusetzen, und dabei erfolgreich waren,
- ❑ agile Methoden bisher nicht in großen Projekten eingesetzt haben, dies aber gern einmal tun würden,
- ❑ an das lineare (Wasserfall-)Modell glauben und meinen, agile Prozesse würden (in großen Projekten) ohnehin nicht funktionieren,
- ❑ an agile Prozesse glauben, aber davon ausgehen, dass diese nie im Großen funktionieren können.

Als Leser arbeiten Sie also typischerweise in einem großen Projekt als Manager, Prozesscoach, Berater oder Entwickler. Sie oder andere wollen dabei einen agilen Prozess einsetzen, sind aber unsicher, ob und wie das gehen kann.

## Wie ist das Buch aufgebaut?

Das Buch hat folgenden Aufbau:

- ❑ **Kapitel 2, *Agilität und Größe***, führt die Prinzipien und das Wertesystem agiler Prozesse ein und wirft dabei die Frage auf, was diese für große Teams bedeuten.
- ❑ **Kapitel 3, *Umgang mit großen Teams***, beschäftigt sich mit dem Einfluss, den ein Wechsel auf agile Prozesse für große Teams hat. Welche Auswirkungen hat dieser Wechsel auf die einzelnen Personen und wie kann man Teams in Teilteams aufteilen sowie Teamrollen so definieren, dass alles zusammenspielt? Darüber hinaus betrachten wir virtuelle Teams am Beispiel von verteilten Teams und der Open-Source-Gemeinde.
- ❑ **Kapitel 4, *Umgang mit dem Prozess***, konzentriert sich auf die Aspekte des Prozesses, die es ermöglichen, viele Teilteams zu koordinieren, ohne diese dabei zu sehr zu reglementieren. Das Ziel besteht darin, all diese Teams so zusammenzubringen, dass sie in all ihren Aktivitäten agil bleiben.
- ❑ **Kapitel 5, *Umgang mit der Technologie***, betrachtet die Auswirkungen der Projekt- und Teamgröße auf die Architektur. Es beschreibt die Rolle des Architekten und inwiefern die Architektur dem Team dient. Außerdem werden einige Techniken und Praktiken diskutiert, die ein agiles System unterstützen.
- ❑ **Kapitel 6, *Umgang mit dem Unternehmen***, beschäftigt sich mit den Problemen, die bei einem agilen Projekt allein dadurch verursacht werden, dass das Projekt in einem großen Unternehmen durchgeführt wird. Dies ist deshalb ein Thema, da große Projekte in der Regel in großen Firmen durch-

---

geführt werden und große Firmen an sich meist nicht mit der Flexibilität agiler Projekte Schritt halten können. Diese Probleme können in gleicher Form auch bei kleinen Projekten auftreten, die innerhalb einer großen Organisation durchgeführt werden.

- **Kapitel 7, *Ein Projektbericht***, liefert schließlich einen zusammenhängenden konkreten Erfahrungsbericht über ein großes agiles Projekt.