mitp Professional

Essential Scrum

Umfassendes Scrum-Wissen aus der Praxis

von
Kenneth S. Rubin

2014

mitp/bhv 2014

Verlag C.H. Beck im Internet:
www.beck.de
ISBN 978 3 8266 9047 1

Inhaltsverzeichnis: Essential Scrum – Rubin

mitp

Kenneth S. Rubin

Mit Vorworten von Mike Cohn und Ron Jeffries

Essential Scrum

Umfassendes Scrum-Wissen aus der Praxis

Kapitel 1

Einführung

Am 21. Juni 2000 war ich als stellvertretender Vorstandsvorsitzender bei Genomica tätig, einer Bioinformatik-Firma in Boulder, Colorado. Dieses Datum ist mir deshalb in besonderer Erinnerung, weil mein Sohn Asher an diesem Tag um 1 Uhr nachts geboren wurde. Und seine Geburt war zweifellos ein guter Auftakt für diesen Tag. Asher wurde tatsächlich am berechneten Geburtstermin geboren (das passiert in den USA nur bei etwa 5% der Geburten) – wir (oder besser gesagt meine Frau Jenine) hatten unser Neun-Monats-»Projekt« also absolut pünktlich abgeschlossen. Und zur Krönung des Ganzen war ihm auch noch ein sehr hoher Apgar-Wert attestiert worden, was eindeutig belegte, dass wir ein gesundes, hochwertiges »Erzeugnis« produziert hatten! Unser wichtigster »Anteilseigner« (Stakeholder), unser älterer Sohn Jonah, war begeistert, einen jüngeren Bruder zu haben. Ein pünktlich geliefertes, hochwertiges Produkt und noch dazu hocherfreute Anteilseigner – es war wirklich ein guter Tag!

Nach einem kurzen Nickerchen rief ich meine E-Mails ab und sah, dass der Vorstandschef von Genomica eine dringende Nachricht geschickt hatte, in der er mich bat, um 8 Uhr zu einem Treffen des Vorstands zu erscheinen. Widerstrebend verließ ich das Krankenhaus und ging zu der Sitzung.

Als ich ankam, erfuhr ich, dass der Leiter der technischen Entwicklungsabteilung am Abend zuvor gefeuert worden war und ich das 90 Leute starke Technikteam geerbt hatte. Das überraschte mich allerdings nicht: Bereits seit mehreren Monaten hatten die Führungsmannschaft und der Vorstand Genomicas Unfähigkeit diskutiert, wertige Produkte von guter Qualität pünktlich auszuliefern – und der Leiter der technischen Entwicklung hatte im Mittelpunkt dieser Diskussion gestanden.

Jetzt war ich verantwortlich dafür, die Ergebnisse unserer Produktentwicklung deutlich zu verbessern. Ich erinnere mich noch, wie paradox es mir vorkam, dass zwei so bedeutsame Ereignisse wie die glückliche Geburt meines Sohnes und die Übernahme meines neuen Verantwortungsbereichs auf ein und denselben Tag fielen.

Da ich mit dem Verkauf und dem Marketing ohnehin schon ziemlich ausgelastet war, gestand man mir zu, dass ich nach meinem Ermessen einen Technikchef einstellen konnte, der mir direkt unterstellt war. Also besetzte ich diesen Posten mit Mike Cohn (Cohn 2004; Cohn 2006; Cohn 2010) und wir beschlossen, es mit Scrum zu versuchen.

1.1 Was ist Scrum?

Scrum ist ein agiler Ansatz zur Entwicklung innovativer Produkte und Dienstleistungen. Abbildung 1.1 zeigt ein solches agiles Entwicklungskonzept in einer einfachen, generischen Variante.

Bei einem agilen Ansatz beginnen Sie zunächst mit einem **Product Backlog** – einer priorisierten Liste der Funktionen und Fähigkeiten, die erforderlich sind, um ein erfolgreiches Produkt zu entwickeln. Wenn Sie sich an das Product Backlog halten, arbeiten Sie immer zuallererst die wichtigsten Aufgaben ab, also diejenigen mit der höchsten Priorität. Sollten Ihnen dann irgendwann die Ressourcen ausgehen (wie z. B. die Zeit), haben somit alle noch verbleibenden Aufgaben eine niedrigere Priorität als die bereits abgeschlossenen Arbeiten.

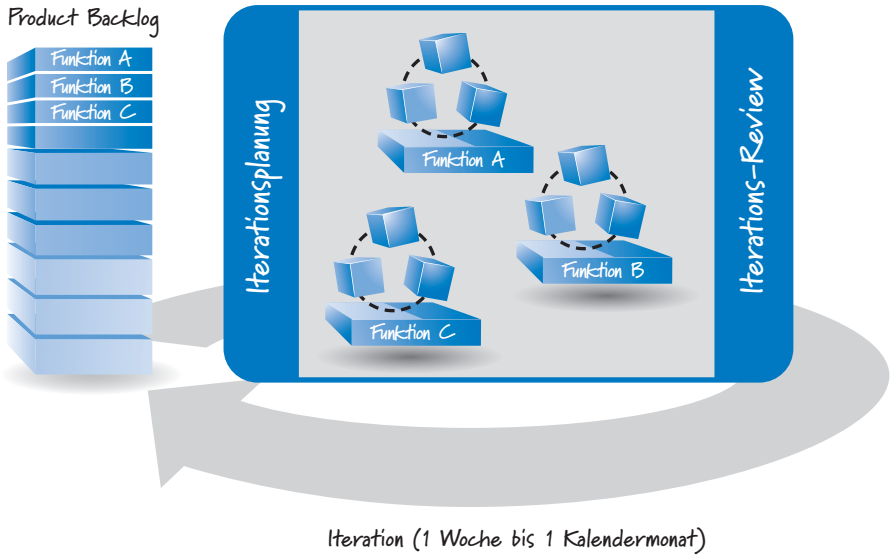

Abb. 1.1: Übersicht über die agile Entwicklung

Die Arbeiten selbst werden in kurzen, zeitlich fest begrenzten (time-boxed) **Iterationen** durchgeführt, wobei sich der hierfür vorgesehene Zeitrahmen normalerweise zwischen einer Woche und einem Kalendermonat bewegt. Während der einzelnen Iterationen erledigt ein Team die Arbeiten, die nötig sind, um abgeschlossene, funktionierende Elemente herzustellen, die dann in die Produktion überführt werden können. Das Team organisiert sich selbst und kann sich **mehrerer Funktionen** annehmen – wie etwa dem Entwerfen, Bauen und Testen.

Üblicherweise ist in der Planung eines Product Backlogs deutlich mehr Arbeit vorgesehen, als ein Team innerhalb einer kurzen Iteration bewältigen kann. Deshalb legt das Team zu Beginn jeder Iteration zunächst einmal fest, welche hoch priorisierte Teilmenge des Product Backlogs in der kommenden Iteration erledigt werden soll. In dem Beispiel aus Abbildung 1.1 ist es z. B. übereingekommen, sich den Funktionen A, B und C zu widmen.

Am Ende der Iteration prüft das Team die abgeschlossenen Funktionen noch einmal gemeinsam mit den Stakeholdern, um deren Feedback zu erhalten. Und je nachdem, welche Kritikpunkte dabei zutage treten, können der Product Owner und das Team ihre Pläne für die nächsten Arbeitsschritte ändern. Falls die Stakeholder bei genauerer Betrachtung einer bereits abgeschlossenen Funktion also etwa feststellen, dass noch eine andere Funk-

tion in das Produkt eingebracht werden muss, die zuvor unberücksichtigt geblieben war, kann der Product Owner hierfür einfach ein entsprechendes neues Element anlegen, das dann an der passenden Stelle im Product Backlog eingefügt und in einer künftigen Iteration bearbeitet wird.

Am Ende der einzelnen Iterationen sollte das Team ein potenziell auslieferungsfähiges Produkt haben (oder ein Inkrement des Produkts, also eine Produktfunktionalität), das prinzipiell freigegeben werden könnte. Sollten individuelle Freigaben nach jeder Iteration hingegen nicht sinnvoll sein, könnten alternativ auch Funktionssätze aus mehreren Iterationen zusammen freigegeben werden.

Nach Beendigung einer Iteration fängt der ganze Prozess mit der Planung der nächsten Iteration wieder von vorn an.

1.2 Die Ursprünge von Scrum

Die reiche Geschichte von Scrum lässt sich bis zu einem Artikel im *Harvard Business Review* aus dem Jahre 1986 zurückverfolgen: »The New New Product Development Game« (Takeuchi und Nonaka 1986). Dieser Artikel beschreibt, wie Unternehmen wie Honda, Canon und Fuji-Xerox mit einem skalierbaren, teambasierten Ansatz für die **ganzheitliche Produktentwicklung** erstklassige Ergebnisse erzielten. Er betont außerdem die Bedeutung selbstorganisierter Teams mit Entscheidungsvollmacht und umreißt die Rolle des Managements im Entwicklungsprozess.

Die besondere Relevanz dieses Artikels von 1986 bestand darin, dass er viele der Konzepte miteinander verband, die am Ende das ausmachten, was wir heute als Scrum bezeichnen. Scrum ist kein Akronym, sondern ein Begriff aus dem Rugby, wo er sich auf eine Methode bezieht, das Spiel nach einem Verstoß oder einem Aus des Balls wieder aufzunehmen. Im Prinzip bezeichnet es das Gedränge, wenn zwei zum Angriff ansetzende Spielerhorden mit gesenkten Köpfen und verschränkten Armen aufeinander losstürmen, um in den Besitz des Balls zu gelangen.

Takeuchi und Nonaka benutzten diese Rugby-Metapher zur Beschreibung der Produktentwicklung:

> *Der ... »Stafettenlauf«-Ansatz für die Produktentwicklung ... widerspricht vermutlich dem Streben nach maximaler Geschwindigkeit und Flexibilität. Allerdings ist ein holistisches oder »Rugby«-Vorgehen – bei dem das Team als Einheit agiert und sich auf dem Weg zum Tor den Ball zuspielt – für die heutigen Konkurrenzbedingungen möglicherweise besser geeignet.*

1993 schufen Jeff Sutherland und sein Team bei der Easel Corporation den Scrum-Prozess, um ihn bei der Software-Entwicklung einzusetzen. Dabei kombinierten sie Konzepte aus dem Artikel von 1986 mit Konzepten aus der objektorientierten Entwicklung, der empirischen Prozesssteuerung, der iterativen und inkrementellen Entwicklung, der Software-Prozess- und -Produktivitätsforschung und aus komplexen adaptiven Systemen. 1995 veröffentlichte Ken Schwaber bei der OOPSLA 1995 einen ersten Artikel zum Thema Scrum (Schwaber 1995). Seither haben Schwaber und Sutherland sowohl zusammen als auch einzeln mehrere Scrum-spezifische Veröffentlichungen herausgebracht, darunter *»Agile Software Development with Scrum«* (Schwaber und Beedle 2001), *»Agile*

Project Management with Scrum« (Schwaber 2004) und »*The Scrum Guide*« (Schwaber und Sutherland 2011).

Obwohl Scrum vorwiegend für die Entwicklung von Software-Produkten eingesetzt wird, lassen sich seine Grundwerte und Prinzipien auch für die Entwicklung anderer Produkte oder zum Organisieren der **Arbeitsabläufe (Workflow)** unterschiedlicher Betätigungsfelder nutzen. So habe ich beispielsweise mit Organisationen zusammengearbeitet, in denen Scrum ebenso erfolgreich für die Abwicklung und Verwaltung von Hardware-Entwicklungen, Marketingprogrammen und Verkaufsinitiativen eingesetzt wurde.

1.3 Wieso Scrum?

Aber aus welchem Grund war denn nun ein agiler Ansatz wie Scrum eine gute Wahl für Genomica? Zunächst einmal war unübersehbar, dass die frühere Vorgehensweise in Bezug auf die Entwicklungsarbeit bei Genomica einfach nicht funktionierte. Das war die schlechte Nachricht. Die gute Nachricht lautete, dass sich in diesem Punkt fast alle Beteiligten einig waren.

Genomica agierte in einem komplexen Betätigungsfeld, in dem es mehr unbekannte als bekannte Faktoren gab. Wir stellten Produkte her, die noch niemals zuvor gebaut worden waren. Unser Fokus lag auf technologisch absolut neuartigen, sich ständig weiterentwickelnden Informatikplattformen, die von Wissenschaftlern in der Forschung benutzt werden würden, um neue bahnbrechende Moleküle zu entdecken. Dementsprechend benötigten wir eine Entwicklungsmethode, die es uns erlaubte, neue Ideen und Ansätze schnellstmöglich auf den Prüfstand zu stellen und herauszufinden, welche Lösungen machbar waren und welche nicht. Wir hatten ein Partnerunternehmen, dem wir alle paar Wochen funktionierende Ergebnisse präsentieren mussten, um ein Feedback zu erhalten, weil unser Produkt in deren Linie von DNA-Sequenzern passen musste. Aber die daraus resultierende Notwendigkeit für schnelle Machbarkeitsstudien und Feedback passte nicht zu der bei uns üblichen ausführlichen Vorausplanung.

Darüber hinaus wollten wir es vermeiden, schon im Vorfeld einen riesigen Architekturentwurf ausarbeiten zu müssen. Ein früherer Versuch, eine neue Generation von Genomicas Kernprodukt zu schaffen, hatte dazu geführt, dass die Organisation fast ein ganzes Jahr ausschließlich mit der Arbeit an der Architektur zugebracht hatte, um eine umfassende, einheitliche Bioinformatik-Plattform herzustellen. Doch als die erste rein forschungsorientierte Anwendung auf diese Architektur aufgesetzt wurde und wir schließlich Designentscheidungen validieren konnten, die wir Monate zuvor getroffen hatten, dauerte es ganze 42 Sekunden, um auf dem Bildschirm von einem Feld zum nächsten zu springen. Und wenn Sie glauben, dass der typische Benutzer ungeduldig ist, dann stellen Sie sich nur mal einen promovierten Molekularbiologen vor, der 42 Sekunden warten muss! Es war katastrophal. Wir brauchten einen anderen, ausgeglicheneren Designansatz, der einen gewissen Anteil des Vorabdesigns mit einer gesunden Dosis an neuem, situationsbezogenen »Just-in-time«-Design kombinierte.

Außerdem sollten unsere Teams vielseitiger werden. Bislang hatte Genomica wie die meisten Organisationen gearbeitet: Die Entwicklungsabteilung übergab die Arbeiten erst dann an die Testteams, wenn sie vollständig abgeschlossen waren. Jetzt wollten wir hingegen dazu übergehen, dass sich die Teammitglieder wesentlich häufiger miteinander synchroni-

sierten – vorzugsweise täglich. Denn in der Vergangenheit hatte die mangelhafte Kommunikation hinsichtlich gravierender Probleme, die bereits während der Entwicklungsphase aufgetreten waren, zu einer Verschärfung der zugrunde liegenden Fehler geführt. Die Leute in den unterschiedlichen Bereichen redeten einfach nicht oft genug miteinander.

Aus diesen und anderen Gründen entschieden wir, dass Scrum sich gut für Genomica eignen würde.

1.4 Ergebnisse bei Genomica

Als wir beschlossen, es mit Scrum zu versuchen, war diese Methode noch relativ unbekannt – das erste Scrum-Buch erschien erst im darauffolgenden Jahr (Schwaber und Beedle 2001). Also sammelten wir so viele verfügbare Informationen wie möglich und bemühten uns sehr, das Erlernte umzusetzen – was in jedem Fall deutlich besser funktionierte als die Art und Weise, wie wir vorher vorgegangen waren (siehe Tabelle 1.1).

Der Aufwand für die Entwicklung eines vergleichbaren Volumens an Produktfunktionalität reduzierte sich mit der Scrum-Entwicklung im Vergleich zu unserem früheren planorientierten Vorgehen nach dem **Wasserfall-Modell** auf ein Zehntel (berechnet in Mann-Monaten). Mindestens genauso wichtig war, dass die Scrum-Entwicklung siebenmal schneller voranging als die Wasserfall-Entwicklung – und damit produzierte sie auch etwa siebenmal mehr wertvolle Funktionen als das Wasserfall-Entwicklungsverfahren. Noch überzeugender war allerdings die Tatsache, dass wir die Software in einem zeitlichen Rahmen an unseren Partner liefern konnten, der die Anforderungen für die Einführung seiner neuen Hardware-Plattform erfüllte. Und dadurch waren wir nun in der Lage, eine bereits lange bestehende Partnerschaft zu stärken, was sich wiederum positiv auf den wirtschaftlichen Wert von Genomica auswirkte.

Maß	Wasserfall	Scrum
Aufwand	10x	1x
Schnelligkeit	1x	7x
Kundenzufriedenheit	Schlecht	Ausgezeichnet

Tabelle 1.1: Scrum-Ergebnisse bei Genomica

1.5 Kann Scrum Ihnen helfen?

Die Erfahrungen, die Genomica vor der Einführung der Scrum-Technik gemacht hatten, nämlich dass Funktionen entwickelt wurden, die niemand wollte und diese zudem in schlechter Qualität und verspätet ausgeliefert wurden, sind nicht ungewöhnlich. Wie viele andere Organisationen hat auch Genomica überlebt, weil es nicht schlechter als die Konkurrenz war. Das gleiche Problem konnte ich bereits beobachten, als ich Mitte der 1980er Jahre begonnen hatte, in der kommerziellen Software-Entwicklung zu arbeiten. Und für viele hat sich die Situation auch mehr als 30 Jahre später noch nicht verbessert.

Welche Antwort würden Sie bekommen, wenn Sie Ihre Manager und Entwickler heute zusammenrufen und fragen würden: »Seid ihr mit den Ergebnissen eurer Software-Ent-

wicklungsbemühungen zufrieden?« oder »Glaubt ihr, dass wir unseren Kunden zu einem spürbaren zeitlichen, wirtschaftlichen und qualitativen Nutzen verhelfen?«

In sehr vielen Fällen haben die Leute, die ich im Rahmen meiner weltweiten Trainings- und Beratungstätigkeit dazu befragt habe, mit einem klaren »Nein« geantwortet. Vielmehr tönte es von allen Seiten: »Die Projektausfallrate ist inakzeptabel hoch«, »Die Auslieferung erfolgt immer verspätet«, »Die Rentabilität liegt unter den Erwartungen«, »Die Software-Qualität ist mies«, »Die Produktivität ist beschämend«, »Niemand übernimmt die Verantwortung für die Ergebnisse«, »Die Arbeitsmoral ist schlecht«, »Wir haben eine hohe Mitarbeiterfluktuation«. Und dann wird mit einem desillusionierten Lächeln und ironischem Unterton noch angefügt: »Es müsste doch auch besser gehen.«

Bei aller Verdrossenheit scheinen sich die meisten Leute damit abgefunden zu haben, dass Unzufriedenheit zur Software-Entwicklung einfach dazugehört. Dabei muss das nicht so sein.

Organisationen, die Scrum gewissenhaft einsetzen, haben da ganz andere Erfahrungen gemacht (siehe Abbildung 1.2).

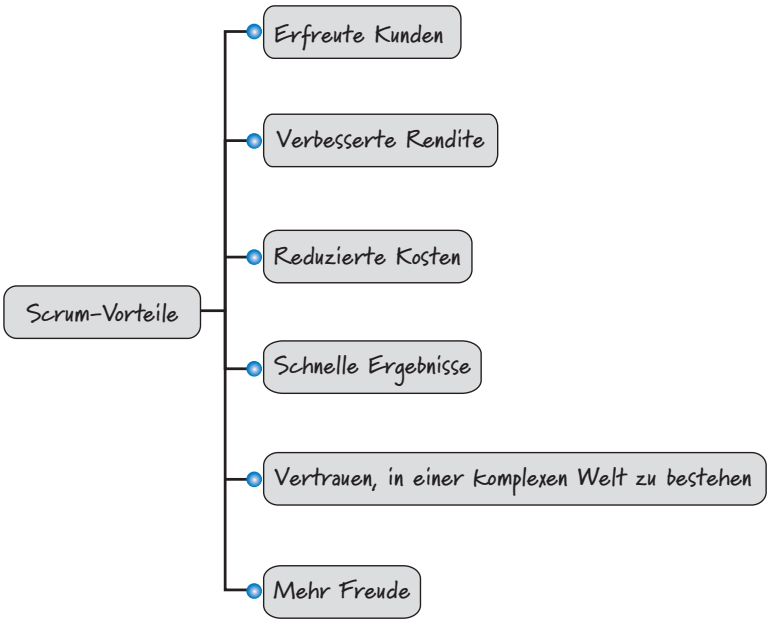

Abb. 1.2: Die Vorteile von Scrum

Diese Organisationen begeistern ihre Kunden in schöner Regelmäßigkeit, indem sie ihnen wirklich das liefern, was sie sich wünschen, und nicht bloß die Funktionen, die am ersten Tag festgelegt wurden, als noch niemand den wahren Bedarf kannte. Außerdem erwirtschaften sie eine bessere Rendite, weil sie häufiger kleinere **Releases** herausbringen. Und durch das schonungslose Aufdecken mangelhafter organisatorischer Abläufe und **Verschwendungen** sind diese Organisationen zudem in der Lage, die Kosten zu senken.

Da sich Scrum darauf konzentriert, funktionierende, integrierte, getestete und geschäftsfördernde Funktionen zu liefern, führt jede Iteration zu Ergebnissen, die zügig ausgeliefert werden können. Darüber hinaus erleichtert es dieses Entwicklungsverfahren den Organisationen, in einer komplexen Welt, in der man sich schnellstmöglich anpassen muss, weil inzwischen alles miteinander vernetzt ist – Konkurrenten, Kunden, Anwender, Aufsichtsbehörden und andere Akteure – erfolgreich zu sein. Und schließlich bereitet Scrum wirklich allen Beteiligten mehr Spaß an der Sache – nicht nur den Kunden, sondern auch den Leuten, die die eigentliche Arbeit verrichten –, denn sie arbeiten öfter und enger zusammen, was zu verbesserten zwischenmenschlichen Beziehungen und größerem gegenseitigen Vertrauen unter den Teammitgliedern führt.

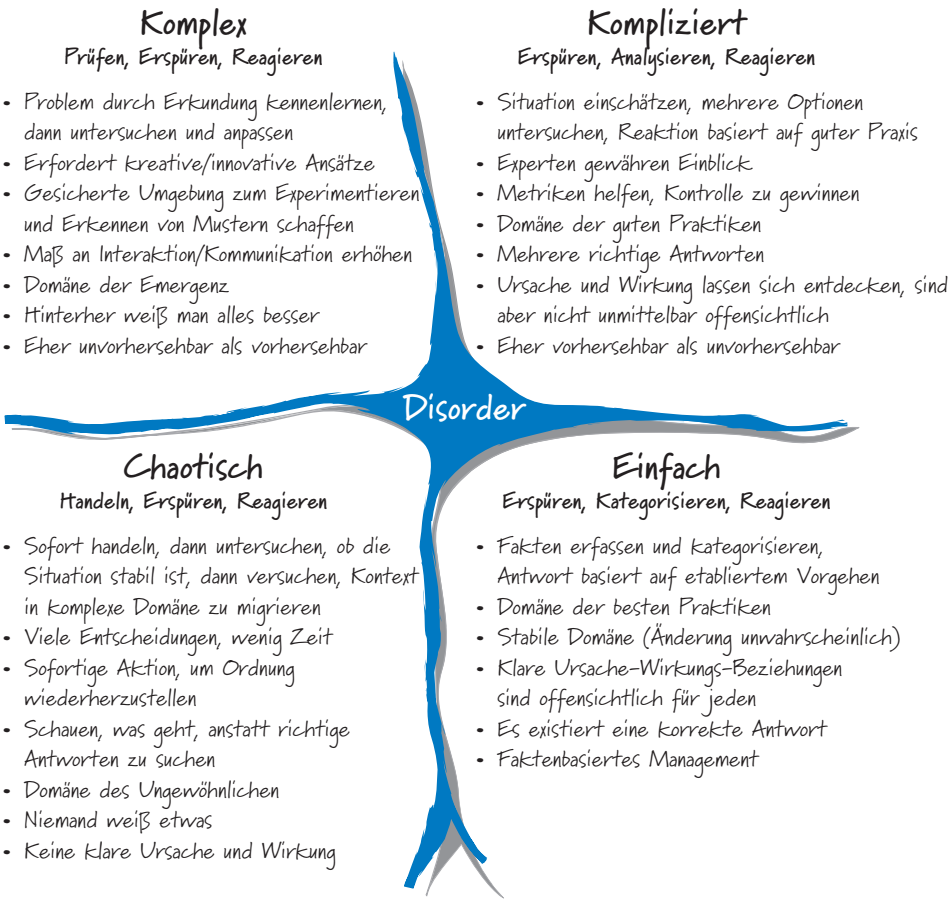

Abb. 1.3: Cynefin-Framework

Verstehen Sie mich nicht falsch: Scrum ist zwar in vielen Situationen eine ausgezeichnete Lösung, aber trotzdem nicht in allen Fällen das geeignete Mittel zum Zweck. Das **Cynefin**-Framework (Snowden und Boone 2007) ist ein sinnvolles Hilfsmittel, um die vorliegende

Situation beurteilen und feststellen zu können, ob Handlungsbedarf besteht und welcher Verfahrensansatz der richtige ist. Dieses Framework definiert und vergleicht die Eigenschaften fünf unterschiedlicher Domänen: **Simple** (einfach), **Complicated** (kompliziert), **Chaotic** (chaotisch), **Complex** (komplex) und **Disorder** (Nicht-Wissen, Regellosigkeit), wobei Letztere dann zutrifft, wenn sich Ihre Situation keiner der anderen Domänen eindeutig zuordnen lässt (siehe Abbildung 1.3). Im Folgenden werde ich anhand des Cynefin-Frameworks Situationen aufzeigen, in denen Scrum eine sinnvolle Lösungsmöglichkeit darstellt und in denen das nicht der Fall ist.

Zunächst einmal muss an dieser Stelle jedoch festgehalten werden, dass sich die vielen Facetten der Software-Entwicklung und des Software-Supports nicht fein säuberlich in nur eine Cynefin-Domäne pressen lassen. Vielmehr handelt es sich um ein weit gefasstes Betätigungsfeld, das einige sich überschneidende Aspekte, aber auch individuelle Aktivitäten umfasst, die in unterschiedliche Domänen fallen (Pelrine 2011). Die Software-Entwicklungsarbeit lässt sich also zwar meist den Domänen **Complicated** oder **Complex** zuordnen, es wäre aber naiv zu behaupten, dass die Software-Entwicklung eine komplexe Domäne wäre. Das gilt vor allem, wenn wir definieren, dass sie ein Arbeitsspektrum aufweist, das von der Entwicklung innovativer neuer Produkte über die Wartung und Weiterentwicklung vorhandener Software-Produkte bis zum Betrieb und Support von Software reicht.

1.5.1 Die Complex-Domäne

Beim Umgang mit komplexen Problemen hat man es mit mehr unvorhersehbaren als vorhersehbaren Faktoren zu tun. Wenn es eine richtige Lösung gibt, erkennt man diese oft erst im Nachhinein. Es ist die Domäne der **Emergenz**, des »In-Erscheinung-Tretens«. Wir müssen Sachverhalte erkunden, um das Problem zu ergründen, dann müssen wir sie **überprüfen** und anhand unserer Erkenntnisse eine **Anpassung** vornehmen (»Inspect and Adapt«). Das Arbeiten in komplexen Domänen erfordert ein kreatives und innovatives Vorgehen. Vorgestanzte Routinelösungen lassen sich einfach nicht anwenden. Wir müssen eine gesicherte Umgebung für Experimente schaffen, damit wir wichtige Informationen entdecken können. In dieser Umgebung ist ein hohes Maß an Interaktion und Kommunikation unerlässlich. Die Entwicklung innovativer neuer Produkte fällt ebenso in diese Kategorie wie das Verbessern vorhandener Produkte mit innovativen neuen Funktionen und Eigenschaften.

Scrum ist für das Arbeiten in einer komplexen Domäne besonders gut geeignet. In solchen Situationen ist unsere Fähigkeit zum Erkunden, Untersuchen und Reagieren entscheidend.

1.5.2 Die Complicated-Domäne

Komplizierte Probleme sind die Domäne der bewährten Praktiken und werden von Experten dominiert. Es mag mehrere richtige Lösungen geben, allerdings ist eine Expertendiagnose nötig, um diese zu ermitteln. Scrum kann sicher auch mit solchen Problemen zurechtkommen, jedoch ist es möglicherweise nicht das ideale Mittel zum Zweck. So wäre ein Versuch der Leistungsoptimierung durch Anpassung diverser Parameter, um die beste Gesamtsystemleistung zu finden, erfolgversprechender, wenn man Experten versammelte, die die Situation einschätzen, mehrere Optionen untersuchen und ihre Lösungsvorschläge

auf einer bewährten Praxis aufbauen. Ein Großteil der alltäglichen Software-Wartung (das Durchführen von Produkt-Support oder das Beheben von Defekten) fällt in diese Kategorie. Hier sind viele der taktischen, quantitativen Ansätze wie Six Sigma besonders gut geeignet, wenngleich diese auch bei einfachen Domänen eingesetzt werden können.

1.5.3 Die Simple-Domäne

Bei einfachen Problemen sind Ursache und Wirkung klar erkennbar. Oft ist die richtige Lösung offensichtlich und unbestritten. Dies ist die Domäne der anerkannt besten Praktiken. Es gibt also bekannte Lösungen. Sobald wir die Fakten unserer Situation erfasst haben, können wir die passende vordefinierte Lösung ermitteln. Scrum kann auch für einfache Probleme eingesetzt werden, ist aber für diese Art von Problemstellung möglicherweise nicht das effizienteste Werkzeug. Besser wäre es, einen Prozess mit einer Menge wohldefinierter wiederholbarer Schritte anzuwenden, von denen man weiß, dass sie das Problem beheben. Wollten wir z. B. immer wieder das gleiche Produkt herstellen, wäre ein wohldefinierter Fließbandprozess besser geeignet als Scrum. Auch das Ausliefern des gleichen Commercial-off-the-shelf-Produkts (COTS; quasi ein seriengefertigtes Produkt aus dem Elektronik- bzw. Software-Bereich) an die hundertste Kundenumgebung lässt sich vermutlich besser erledigen, indem man eine wohldefinierte und bewährte Anzahl von Schritten für die Installation und Konfiguration des Produkts durchführt.

1.5.4 Die Chaotic-Domäne

Chaotische Probleme erfordern eine schnelle Reaktion. Wir haben eine Krise und müssen sofort handeln, um weiteren Schaden zu vermeiden und wenigstens eine gewisse Ordnung wiederherzustellen. Nehmen Sie z. B. an, dass eine Universität einen Artikel veröffentlicht hätte, der besagt, dass unser Produkt einen mangelhaften Algorithmus enthält, der fehlerhafte Ergebnisse erzeugt. Unsere Kunden haben auf der Grundlage der Ergebnisse unseres Produkts beträchtliche Investitionen getätigt und verklagen uns nun auf Schadenersatz. Unser führender Algorithmendesigner wandert gerade auf Borneo durch den Dschungel und ist in den nächsten zwei Wochen nicht erreichbar. In diesem Fall ist Scrum nicht die beste Lösung. Uns interessiert hier nicht, welche Priorität ein Backlog hat und wie wir die nächste Iteration erreichen können – wir müssen stattdessen in der Lage sein, schnell zu handeln und den Schaden zu begrenzen. Bei chaotischen Problemen muss jemand das Heft in die Hand nehmen und Entscheidungen treffen.

1.5.5 Disorder (Nicht-Wissen, Regellosigkeit)

Mit der Disorder-Domäne haben Sie es dann zu tun, wenn sich Ihre Situation nicht eindeutig einer der anderen Domänen zuordnen lässt. Das ist insofern gefährlich, weil Sie nicht wirklich wissen, wie Sie Ihre Situation einschätzen sollen. In solchen Fällen neigen die Menschen dazu, die Lage entsprechend ihrer persönlichen Vorlieben zu interpretieren und auch danach zu handeln. In der Software-Entwicklung sind viele Menschen mit phasenbasierten, sequenziellen Ansätzen vertraut, die gut in einfachen Domänen funktionieren, und haben deshalb auch eine Vorliebe dafür. Leider sind diese für viele Vorgänge in der Software-Entwicklung nicht besonders gut geeignet. Ich werde das in Kapitel 3 näher erläutern. Sie können sich aus der Disorder-Domäne befreien, indem Sie die Situation in ein-

zelne Teile zerlegen und diese jeweils einer der vier anderen Domänen zuordnen. In der Disorder-Domäne probieren Sie gar nicht erst Scrum anzuwenden, sondern versuchen vielmehr, diese Domäne zu verlassen.

1.5.6 Unterbrechungsgesteuerte Arbeit

Für stark unterbrechungsgesteuerte Arbeitsabläufe ist Scrum nicht besonders gut geeignet. Nehmen Sie einmal an, Sie betreiben eine Organisation für Kunden-Support und wollen Scrum einsetzen, um Ihre Support-Aktivitäten zu organisieren und zu verwalten. Ihr Product Backlog wird kontinuierlich befüllt, wenn Sie telefonisch oder per E-Mail Support-Anfragen bekommen. Zu keinem Zeitpunkt haben Sie ein Product Backlog, das sehr weit in die Zukunft reicht. Inhalt und Reihenfolge Ihres Backlogs könnten sich häufig ändern (stündlich oder minütlich).

In dieser Situation werden Sie nicht in der Lage sein, zuverlässig Iterationen von einer Woche oder mehr zu planen, weil Sie nicht im Voraus wissen, welche Arbeiten in absehbarer Zukunft anfallen werden. Und selbst wenn Sie glauben, die auf Sie zukommenden Aufgaben zu kennen, besteht eine relativ hohe Wahrscheinlichkeit, dass eine dringende Support-Anfrage (also eine Aufgabe mit hoher Priorität) eintrifft, die allen weitreichenden Plänen zuvorkommt.

In unterbrechungsgesteuerten Umgebungen sollten Sie sich eher einen alternativen agilen Ansatz namens **Kanban** zunutze machen. Kanban ist keine eigenständige Prozesslösung, sondern eher eine Methode, die auf einen vorhandenen Prozess aufgesetzt wird. Im Speziellen verlangt Kanban, dass Sie

- den Workflow visualisieren (wie etwa die Schritte, die eine Support-Organisation unternimmt, um eine Support-Anfrage zu beantworten),
- die laufenden Arbeiten (Work in Process; WIP) in jedem Schritt begrenzen, damit Sie sicher sein können, dass Sie nicht mehr Arbeiten erledigen müssen, als Sie aufgrund Ihrer Kapazität bewältigen können,
- den Workflow durch das System messen und optimieren, um stetig Verbesserungen zu erreichen.

Die optimalen Einsatzgebiete für Kanban sind die Bereiche Software-Wartung und -Support. Manche Anwender halten Kanban aufgrund seiner Fokussierung auf die Eliminierung einer Überlast (indem die laufende Arbeit an die Kapazität angepasst wird) und die Verringerung der Variabilität im Workflow bei gleichzeitiger Unterstützung eines evolutionären Vorgehens außerdem auch für den Einsatz in komplexen Domänen geeignet.

Sowohl Scrum als auch Kanban sind agile Ansätze für die Entwicklung. Beide haben Stärken und Schwächen, die man beachten muss, sobald man sich der Domäne bewusst geworden ist, in der man agiert. In manchen Organisationen können beide Vorgehensweisen eingesetzt werden, um die unterschiedlichen Systemanforderungen zu meistern, die nebeneinander existieren. So könnte man z. B. Scrum für die Entwicklung neuer Produkte verwenden und Kanban für den unterbrechungsgesteuerten Support und die Wartung.

1.6 Abschließende Bemerkungen

Scrum ist keine Wunderwaffe und auch kein Allheilmittel. Es kann Sie jedoch in die Lage versetzen, die Änderungen anzunehmen, die mit allen komplexen **Produktentwicklungen** einhergehen. Für Genomica und viele andere Unternehmen, die sich entschieden haben, einen Ansatz für die Software-Entwicklung zu übernehmen, der besser für ihre speziellen Anforderungen und Gegebenheiten geeignet ist, hat es jedenfalls funktioniert.

Das Scrum-Framework ist zwar einfach, dennoch wäre es ein Fehler anzunehmen, dass es leicht und schmerzlos einzusetzen sei. Scrum gibt Ihnen keine Antworten auf Ihre Prozessfragen vor, sondern bringt Ihr Team dazu, selbst die entscheidenden Fragen zu stellen und auch zu beantworten. Scrum liefert keine fertigen Rezepte für alle organisatorischen Krankheiten, sondern deckt Fehlfunktionen und Verschwendungen auf, die Organisationen daran hindern, ihr wahres Potenzial auszuschöpfen.

Diese Erkenntnisse sind für viele Organisationen erst einmal schmerzhaft. Sofern sie jedoch das anfängliche Unbehagen überwinden und sich bemühen, die Probleme zu beheben, die Scrum zutage gefördert hat, können diese Organisationen sowohl bei ihrer Entwicklungsarbeit als auch bei der Angestellten- und Kundenzufriedenheit Großes erreichen.

Der Rest dieses Buches widmet sich der Diskussion der wesentlichen Aspekte von Scrum. Ich beginne mit einer Beschreibung des gesamten Scrum-Frameworks, einschließlich seiner Rollen, Aktivitäten, Artefakte und Regeln. Wer weiß, wenn Sie Scrum richtig und unter den passenden Bedingungen einsetzen, werden Sie möglicherweise schon bald ebenso erfolgreich hochwertige Ergebnisse liefern wie meine Frau an diesem schicksalhaften Tag im Jahr 2000.