entwickler.press

Scrum

Schnelleinstieg

Andreas Wintersteiger

Andreas Wintersteiger

Scrum

Schnelleinstieg

entwickler.press

Andreas Wintersteiger
Scrum – Schnelleinstieg
ISBN: 978-3-86802-266-7

Ein Imprint der Software & Support Media GmbH

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

Ihr Kontakt zum Verlag und Lektorat:
Software & Support Media GmbH
entwickler.press
Geleitsstr. 14
60599 Frankfurt am Main
Tel.: +49 (0)69 630089-0
Fax: +49 (0)69 930089-89
lektorat@entwickler-press.de
http://www.entwickler-press.de

Lektorat: Sebastian Burkart
Korrektorat: Frauke Pesch
Satz: Dominique Kalbassi
Belichtung, Druck & Bindung: M.P. Media-Print Informationstechnologie GmbH, Paderborn

Inhaltsverzeichnis

Vorwort

Agile Softwareentwicklung verspricht viele Vorteile. So z. B. die Transparenz über den Projektfortschritt, große Flexibilität im Umgang mit Veränderungen und kürzere Time-to-Market. Aber auch die Motivation der Teammitglieder steigt und damit auch Ihre Effizienz. Scrum ist der mit Abstand prominenteste Vertreter agiler Vorgehensweisen. Das liegt mit Sicherheit an seiner Einfachheit, den wenigen und klaren Rollen, der übersichtlichen Anzahl an Meetings und einer geringen Zahl vorgeschriebener Artefakte. Trotz dieser Einfachheit gibt es nach meiner Erfahrung zu viele, die auf Halbwissen basierend mit Scrum beginnen, die wenigen Regeln nicht befolgen und sich wundern, dass Scrum nicht alle ihre Probleme löst. Die wenigen Spielregeln von Scrum sollte man schon kennen und ernst nehmen, wenn man von den Vorteilen, die Scrum bietet, profitieren will.

Andreas Wintersteiger hat mit diesem Buch eine sehr kompakte und doch sehr runde und inhaltsreiche Darstellung von Scrum vorgelegt, die jedem hilft, der sich einen fundierten Überblick und Einblick verschaffen will. Halbwissen zu Scrum existiert schon ausreichend, dieses Buch aber vermittelt Vollwissen.

Mir gefällt es sehr gut, dass Andreas seine Darstellung von Scrum an vielen Stellen um seine Meinung und seine Erfahrung aus der Coaching-Praxis ergänzt hat. So kann der Leser unterscheiden zwischen dem, was Scrum ausmacht, und dem, was der erfahrene Coach empfiehlt. Ich glaube, dass wir in unseren Erfahrungen und Empfehlungen einen sehr ähnlichen Blick auf Scrum haben, zumindest haben wir bei den zahlreichen gemeinsam gegebenen Scrum-Trainings immer hervorragend harmoniert. Danke für diese Trainings, Andreas, Danke für dieses Buch.

Henning Wolf
CEO, it-agile GmbH

Danksagung

An erster Stelle bedanke ich mich bei meiner Frau Petra, die bereits die vergangenen Jahre hindurch mein massiv eingeschränktes Zeitbudget für Privates ertragen musste. Als ich ihr eröffnete, ein zweites Buch zu schreiben, unterstützte sie mich, ohne ein Wort darüber zu verlieren, welche Auswirkungen das auf unser Eheleben hat – danke für deine Unterstützung und dein Verständnis.

Danke an meine Mutter für das oftmalige Lesen und Korrigieren des Manuskripts und sorry für meine Beistrichsetzung nach Programmierlogik ;-)

Ein großes Dankeschön auch an meinen Verleger und Lektor Sebastian Burkart von entwickler.press, seine Flexibilität und Geduld mit mir war vorbildlich. An meine Mitarbeiter bei Objectbay möchte ich ein Dankeschön für die Unterstützung durch Lesen und Hinweisen auf Unverständlichkeiten richten.

Bei meinen Kollegen und Freunden in der Agile- und Scrum-Community möchte ich mich bedanken für die gute Zusammenarbeit und großartigen Diskussionen, die wir in den letzten Jahren führten. Viele Erkenntnisse und Erfahrungen beruhen darauf. Besonderer Dank gilt den Kollegen, mit welchen ich in letzter Zeit gemeinsame Trainings und Coachings halten durfte. Danke an Christoph Mathis, Henning Wolf, Stefan Roock, Peter Beck, Simon Roberts, Boris Gloger, Timo Foegen, Sven Blesin, Daniel Haslinger, Johannes Link, Josef Scherer und Andreas Havenstein.

Ganz besonderen Dank richte ich an unsere zahlreichen Kunden, die uns ihr Vertrauen geschenkt haben und sich mit uns als ihre Begleiter auf die Reise begeben haben, Scrum im Unternehmen einzuführen. Viele praktische Erfahrungen stammen aus unseren Engagements bei ihnen – herzlichen Dank dafür. Ich hoffe, mit diesem Buch ein wenig davon zurückgeben zu können.

1 Einleitung

Agile Softwareentwicklung hat es in den Mainstream geschafft. Elf Jahre nach dem agilen Manifest sind agile Vorgehensweisen keine exotischen Modelle mehr von Eigenbrötlern, die sich ihre eigene Welt schaffen, sondern eine anerkannte und bewiesenermaßen erfolgreiche Art und Weise, Software zu entwickeln[1]. Agile Entwicklung ist heute nicht nur erfolgreich, sondern deutlich *erfolgreicher* als klassische, wasserfallorientierte Vorgehensmodelle – dafür haben wir heute Evidenz.

Mussten wir in den ersten Jahren der letzten Dekade immer wieder den Beweis antreten, so sind wir heute einen bedeutenden Schritt weiter: Es ist nicht nur die Akzeptanz im breiten Feld vorhanden, ein großer Teil der Unternehmen strebt agile Entwicklungspraktiken und -methoden an. Darunter ganz besonders Scrum, das aus heutiger Sicht sicherlich das erfolgreichste Framework[2] unter den agilen Methoden ist. Scrum erfreut sich zunehmender Beliebtheit und Verbreitung. Immer mehr Unternehmen und Teams setzen Scrum und agile Entwicklungspraktiken ein – oder behaupten das zumindest.

Als Scrum Coach komme ich zu vielen Unternehmen, und als Trainer führe ich viele Gespräche mit Teammitgliedern oder Vertretern von Organisationen, die Scrum – oft schon seit Jahren – verwenden. Dabei stelle ich manchmal fest, dass es viele Missverständnisse gibt, was Agile und Scrum ausmacht und was nicht.

---

1 In Trainings habe ich schon öfter zu hören bekommen, dass Scrum und agile Methoden sich eine völlig eigene, künstliche Welt schaffen und die Teams, in dieser Blase lebend, so tun, als würde es die Außenwelt nicht geben.

2 Wir werden in Kapitel 4 genau betrachten, warum wir Scrum nicht als Methode, sondern als „Framework" bezeichnen.

1.1 Agile Softwareentwicklung

Im Gegensatz zu den klassischen Vorgehensweisen stützen sich agile Prozesse auf frühes und häufiges Feedback aufgrund von fertig gestellten Teillieferungen. Diese Teile müssen jedoch voll funktionsfähig sein, um den Anspruch an wertvolles Feedback zu erfüllen. Nur wirklich funktionierende Software erzeugt realistisches Feedback. Pläne, Diagramme oder Folienpräsentationen im Kontrast dazu verlangen danach, dass sich Menschen immer etwas vorstellen müssen, denn das Ergebnis ist nicht real. Je nach Komplexität und Vorstellungsvermögen der Beteiligten entsteht damit auch realistisches oder eben unrealistisches Feedback. Im klassischen Vorgehen entsteht reales Feedback oft erst nach Durchlauf vieler Phasen, was oft mehrere Monate, manchmal sogar bis zu einem Jahr dauert.

Iteratives Vorgehen

Im Kern einer jeden agilen Methode stehen *Iterationen*, die Zyklen fixer Dauer sind und zum Ziel haben, Feedback zu liefern. In iterativen Vorgehensweisen werden sämtliche Aktivitäten, die wir aus den klassischen Vorgehensmodellen kennen, wie Analyse, Design, Implementierung, testen, integrieren, Systemtest etc., durchgeführt, jedoch mehrmals. Wir durchlaufen diese Phasen in Zyklen.

Inkrementelles Vorgehen

Bei der inkrementellen Entwicklung wird das iterative Vorgehen verwendet, jedoch auf andere (neue) Teile des Systems angewendet (manchmal wird hierfür auch der Terminus „iterativ-inkrementell“ verwendet). Es handelt sich also um eine Erweiterung des Produkts, und die Iteration betrachtet damit eine neue Entwicklung[3]. Bei der inkrementellen Ent-

3 In der iterativen Entwicklung werden diese Aktivitäten erneut durchlaufen, jedoch das bestehende Produkt betrachtet, bereinigt, erneuert oder Teile davon neu aufgebaut. Iteratives Vorgehen hilft dabei, das Produkt zu verbessern [14].

wicklung wird der Prozess verbessert. Das Ergebnis einer Iteration ist ein Inkrement des Produkts, es wird damit ausgehend von einem kleinen funktionierenden Durchstich ständig erweitert und funktional gehalten.

Inkremente sind beispielsweise Features oder Teile der Produktfunktionalität. Es ist wesentlich festzustellen, dass bei der inkrementellen Entwicklung ein vertikal durch die Systemarchitektur liegender Querschnitt geliefert wird. In klassischen Vorgehensweisen geschieht oft das exakte Gegenteil, es werden dabei ganze Schichten geliefert. Ein funktionaler und damit feedbackfähiger Querschnitt ist erst nach Lieferung der letzten Schicht möglich. Aus diesem Grund erfolgt bei klassischer Entwicklung das Testen so spät.

In der inkrementellen Entwicklung konzentriert man sich auf kleine (vertikale) Teile, die auch sehr früh getestet werden können. Alle anderen Aktivitäten können damit auch beinahe zeitgleich in einer Iteration erledigt werden (Details dazu werden in Kapitel 6 vertieft).

Das Entwicklungsvorhaben wird nicht mehr für den gesamten Umfang und damit auch nicht für die gesamte Zeit im Detail geplant. Eine Planung mit hohem Detailgrad beschränkt sich auf eine Iteration. Planung und Entwicklung – und damit auch die Validierung der Planung – wechseln sich in raschen Rhythmen ab.

Mit bereits deutlich weniger Details beschäftigt sich der nächsthöhere Planungshorizont, der der unmittelbar folgenden Iterationen: Hier werden nur mehr gröbere Ziele auf Basis der Erkenntnisse aus den bisherigen Iterationen geplant.

Mehr Kundenwert in kürzerer Zeit

Agile Prozesse liefern schneller bessere Ergebnisse und sind fokussiert auf die frühe Bereitstellung von Kundenwert. Es wird bei der Priorisierung auf die Auslieferung von Features mit hohem Kundenwert geachtet. Mit jedem Inkrement steht so nach wenigen Wochen ein lieferbarer Teil des Produkts zur Verfügung, das im Idealfall vom Kunden bereits genutzt werden kann.

In einem typischen Scrum-Projekt wird z. B. alle zwei Wochen ein voll funktionsfähiger Teil der Software geliefert und steht für Feedback durch Kunden, Anwender oder Management zur Verfügung. Durch den Fokus auf die frühe Auslieferung von Features mit hohem Kundenwert entsteht deutlich früher ein bereits einsetzbares Produkt, das auf die Problemstellung fokussiert ist.

1.2 Agile Werte und Prinzipien

Die Bewegung in Form einer Ablehnung von klassischen, schwerfälligen Methoden und Vorgehensmodellen begann in den frühen 90er Jahren damit, dass einzelne Produktentwicklungsteams die bisher bekannten Prozesse der Softwareentwicklung wie iterativ und inkrementelles Vorgehen weitergetrieben haben. Darunter fallen auch Ken Schwaber und Jeff Sutherland, die Scrum entwickelt haben. Neben ihnen gab es eine Reihe weiterer, heute bekannter Persönlichkeiten, die ähnliche Praktiken einsetzen und damals noch „leichtgewichtig" genannte Methoden entwickelten. Viele von ihnen haben dazu auch auf wissenschaftlichen Konferenzen publiziert.

Das Agile Manifest

Im Februar 2001 trafen sich diese Personen und versuchten zusammenzutragen, was ihre Vorgehensweisen gemeinsam haben, und kamen – obwohl sie es zuvor nicht glaubten – zu einer Menge an Werten, die sie gleichermaßen hochhielten, und einem Dutzend Prinzipien sowie zu einem Konsens über deren Formulierung. Sie erschufen ein Manifest aus Werten, basierend auf Vertrauen und Respekt für einander, die die Organisationsmodelle fördern, die auf Individuen und Zusammenarbeit fußen. Übersetzt klingt das Manifest in etwa so:

Wir entdecken dadurch bessere Wege, Software zu entwickeln, indem wir es selber tun und andere dabei unterstützen, es zu tun. Durch diese Arbeit sind wir dazu gekommen,

- *Individuen und deren Interaktionen über Prozesse und Werkzeuge,*
- *funktionierende Software über umfassende Dokumentation,*
- *die Zusammenarbeit mit dem Kunden über Vertragsverhandlungen und*
- *das Eingehen auf Veränderung über das Befolgen eines Plans*

zu stellen. Während die Dinge auf der rechten Seite durchaus einen Wert darstellen, schätzen wir die Dinge auf der linken Seite mehr.

Sehr oft wurde dieses Manifest missverstanden und fehlerhaft interpretiert. Keineswegs war damit gemeint, dass es von nun an gar keine Dokumentation geben sollte, gemeint waren die mehreren hundert Seiten langen Dokumente, die niemand wartet und kaum jemand liest.

Diese Aussagen wurden durch zwölf Prinzipien gestützt, die den Kern der agilen Softwareentwicklung definieren. Diese Prinzipien greifen ineinander und wirken als Ganzes:

- Unsere höchste Priorität ist es, den Kunden durch frühe und kontinuierliche Lieferung werthaltiger Software zufriedenzustellen.
- Wir begrüßen veränderte Anforderungen auch in einer späten Entwicklungsphase. Agile Prozesse nutzen Veränderungen für den Wettbewerbsvorteil des Kunden.
- Liefere funktionierende Software häufig, von wenigen Wochen bis zu wenigen Monaten, mit einer Präferenz für kürzere Intervalle.
- Anwender und Entwickler müssen täglich zusammenarbeiten.
- Entwickle Projekte um motivierte Personen herum. Schaffe die Umgebung und Unterstützung, die sie benötigen, und vertraue ihnen, dass sie ihre Arbeit machen.
- Die effizienteste und effektivste Methode, Informationen in ein und innerhalb eines Entwicklungsteams zu übermitteln, ist von Angesicht zu Angesicht.
- Funktionierende Software ist das primäre Maß für Projektfortschritt.

- Agile Prozesse fördern nachhaltige Entwicklung. Die Sponsoren, Entwickler und Anwender sollen unendlich lang ein konstantes Entwicklungstempo beibehalten können.
- Kontinuierliche Aufmerksamkeit für technische Exzellenz und gutes Design unterstützt Agilität.
- Einfachheit – die Kunst, unnötige Arbeit zu vermeiden – ist essenziell.
- Die besten Architekturen, Anforderungen und Designs entstehen in selbstorganisierenden Teams.
- Das Team reflektiert in regelmäßigen Abständen über die Verbesserung seiner Arbeitsweise, verbessert sie und passt sie an.

1.3 Agile Methoden

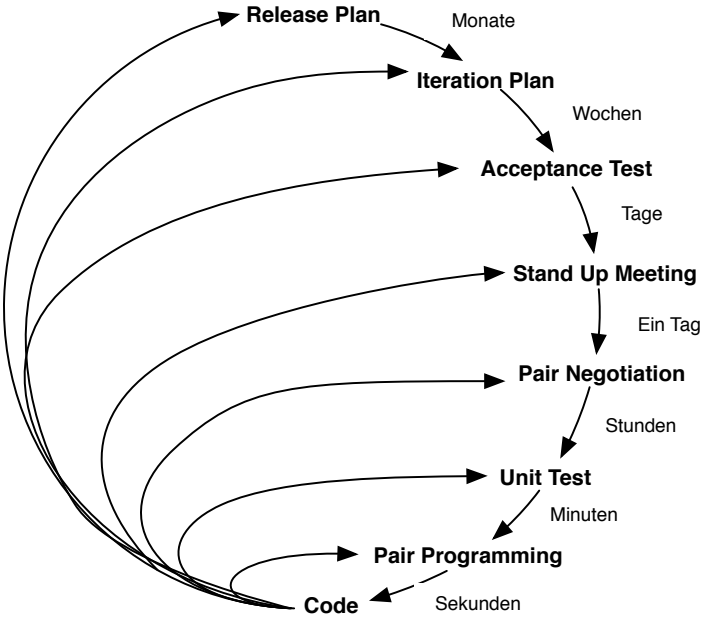
Zum Zeitpunkt der Publikation des Agilen Manifests wurde auch die Agile Alliance durch die Teilnehmer und originalen Unterzeichner gegründet. Diese waren gleichzeitig auch Vertreter der damals breiten agilen Methodenlandschaft. Von den ehemaligen rund 20 Vertretern dieser agilen Methoden sind heute nur noch einige wenige relevante übrig geblieben, die in der Praxis tatsächlich Anwendung finden. Ich möchte kurz auf drei davon eingehen, die mir heute wichtig erscheinen: XP, Scrum und Lean/Kanban.

1.3.1 XP

Extreme Programming (XP) war zumindest in Europa früher deutlich bekannter als Scrum. Kent Beck publizierte 1998 die Methode erstmals mit vier Werten und zwölf Praktiken. In einer zweiten Ausgabe wurde sie leicht erweitert und hat nun fünf Werte, 13 Praktiken, außerdem sind 14 Prinzipien beschrieben.

Extreme Programming hat, wie der Name bereits vermuten lässt, seinen Schwerpunkt in den Praktiken der Entwicklung (Programmierung), definiert aber auch einen Ablauf (Prozess), siehe Abbildung 1.1.

Zu den Praktiken gehören unter anderem testgetriebene Entwicklung, kontinuierliche Integration, inkrementeller Entwurf, räumliche Nähe, Pair Programming, kollektive Verantwortung für den Code etc. Diese Praktiken gehören heute zum guten Ton bei der agilen Entwicklung und haben damit für das Überleben von XP gesorgt. XP hatte immer einen stärkeren Fokus auf die Praktiken und weniger auf den Prozess.



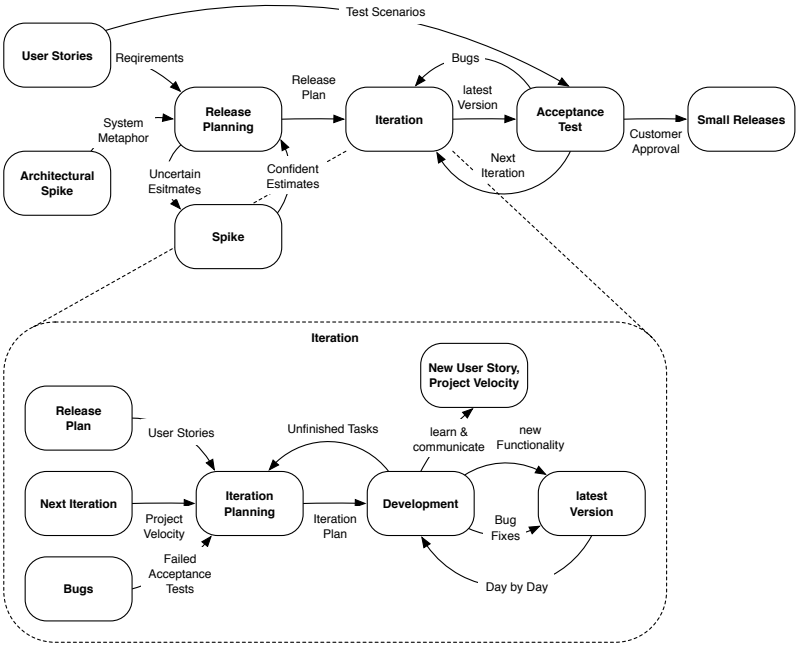


Abbildung 1.1 Feedbackzyklen und Prozesse in XP (vgl. [17] und [18])

XP definiert dennoch Rollen, Iterationen und auch Meetings, z. B. das tägliche Treffen (Daily Standup) oder die Retrospektive. Kent Beck hat viele Ideen aus Scrum übernommen und damit eine mächtigere Methode entwickelt. In XP liegt jedoch der Schwerpunkt auf den Feedbackzyklen, wovon es sieben definiert (vgl. Abb. 1.1). Diese greifen ineinander, sodass ein konzentrisches Zyklensystem entsteht, das vom Minutentakt bis zum Rhythmus von mehreren Monaten dauert.

Ein wesentliches Merkmal von XP ist auch die Unterteilung der Planung in unterschiedliche Horizonte und damit unterschiedliche Detailgrade.

1.3.2 Scrum

Bei Scrum hingegen findet man gar keinen Schwerpunkt auf technischen Praktiken. Scrum legt deutlich mehr Wert auf die geregelte Zusammenarbeit in selbstorganisierten Teams. Es definiert dazu drei Rollen, vier Meetings und drei Artefakte. Eines der Artefakte ist die Forderung nach einem auslieferbaren Produktinkrement am Ende einer Iteration (in Scrum „Sprint" genannt).

Betrachtet man die Mächtigkeit der Methoden, so ist XP deutlich mehr „verordnend" als Scrum, hat also mehr Regulative. Um jedoch Scrum erfolgreich in der Praxis umzusetzen, z. B. das auslieferbare Inkrement am Ende einer Iteration, benötigt man die aus XP bekannten technischen Praktiken.

Scrum hat seine Wurzeln in der empirischen Prozesssteuerung. Durch zeitnahe Feedbackschleifen wird in Scrum zunächst auf Prozessverbesserung fokussiert[4]. Der Name Scrum kommt aus dem Rugby-Sport, wo dieser Begriff übersetzt etwa „Gedränge" bedeutet. Das Scrum in Rugby ist ein komplexer Vorgang, wo sich zwei Teams gegeneinander drängen, um auf einer Seite des Gedränges den Ball zu „befreien". Das geschieht durch eine *Bewegung des gesamten Teams*. Diese Bewegung des ganzen Teams ist Pate für den Namen Scrum, denn auch hier in der Softwareentwicklung geht es darum, dass ein selbstorganisiertes Team die Bewegung durch eine Iteration von der Anforderung bis hin zum auslieferbaren Produktinkrement als eine homogene Einheit durchführt.

4 Wie wir im Rest des Buches noch sehen werden, hat Scrum zwei Feedbackschleifen, um sowohl den Prozess zu verbessern als auch das Produkt inkrementell zu erweitern.

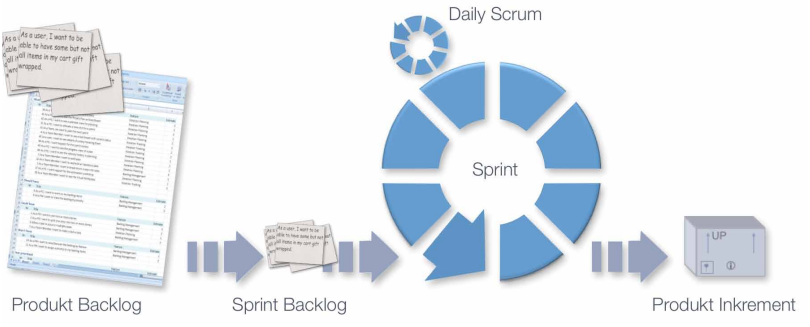


Abbildung 1.2 Der Scrum Flow

So ein Team arbeitet in einem Sprint an einer von ihm ausgewählten Menge an Aufgaben. Diese Aufgaben oder besser Anforderungen an die Software stehen in einer Liste, dem Produkt-Backlog. Sprints sind geschützte Zeiträume fixer Dauer, zum Beispiel von zwei Wochen, während derer das Team nicht gestört wird. Im Gegenzug liefert das Team am Ende fertiggestellte, funktionierende Software, die die zuvor gewählten Anforderungen umsetzt. Ein wesentliches Merkmal von Scrum sind selbstorganisierende Teams. Scrum definiert explizit keinen Prozess, wie und mithilfe welcher Arbeitstechniken das Team die Software entwickelt. In Scrum geht es um die ständige Reflexion und Verbesserung. Dazu definiert Scrum kurze Feedbackzyklen: Der Sprint erlaubt dem Team, im Rhythmus weniger Wochen über die Ergebnisse und den Prozess der Entwicklung zu reflektieren. Ein weiterer Zyklus auf Tagesebene – das „Daily Scrum“ – erlaubt es dem Team, täglich zu reflektieren und seine Arbeit zu organisieren.

1.3.3 Lean und Kanban

Neben der agilen Bewegung hatte sich ein anderer Trend in die Softwarebranche bewegt, der jedoch aus der Produktion stammt. Genauer gesagt, entspringt diese Bewegung der Automobilbranche: Die Ideen *Lean*, *Lean Production* und *Lean Product Development* von Toyota haben sowohl die Entwicklung als auch die Produktion von Automobilen weltweit revolutioniert und fanden ihren Weg über viele weitere Branchen im letzten Jahrzehnt nun auch in die Softwareentwicklung.

Lean basiert auf zwei Säulen:

- Kontinuierliche Verbesserung (Kaizen): Hier geht es um den Problemlösungsprozess an sich und damit darum, die Arbeit, die vom Team gemacht wird, permanent zu verbessern.
- Respektiere die Leute: Es geht darum, zunächst Leute (und Teams) zu entwickeln und damit gute Produkte zu erzeugen. Dazu gehört neben Teamentwicklung und persönlicher Entwicklung, Partner zu entwickeln, aber auch verschwendungsvolle Arbeit zu vermeiden, die Wünsche des Kunden zu erfüllen usw.

Das Grundprinzip ist gleichzeitig auch eine grundlegende Philosophie von Lean: Ein nachhaltiges und langfristiges Denken zu etablieren und sich nicht auf kurzfristige Gewinne zu fokussieren. Basierend auf diesem Grundprinzip, konzentrieren sich die weiteren Prinzipien auf das Vermeiden von Verschwendung, holistische Managementprinzipien und laufende Prozessverbesserung. Toyotas *Lean* umfasst sowohl den Produktionsprozess (Manufacturing) als auch den Entwicklungsprozess von Produkten, also Automobilen.

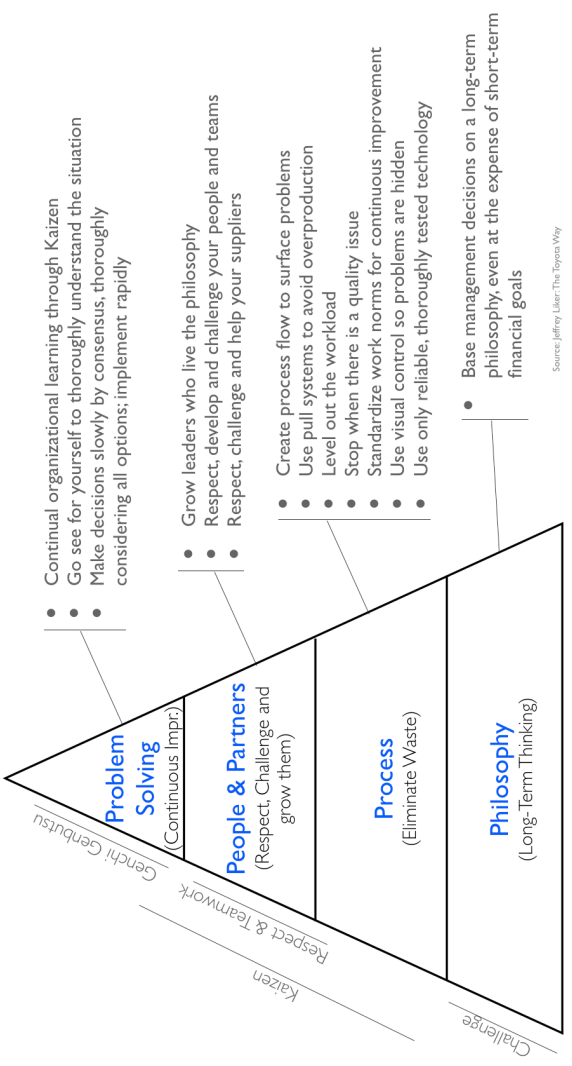


Abbildung 1.3 Die 14 Prinzipien von Lean (Quelle: [22]).