Bertil Muth
(Bertil.Muth@hood-group.com)
ist Senior Consultant bei der HOOD GmbH. Sein Schwerpunkt liegt in den Bereichen agiles Coaching und Consulting.

Uwe Valentini
(Uwe.Valentini@hood-group.com)
ist Berater, Trainer und Coach bei der HOOD GmbH. Vor dem Hintergrund seiner langjährigen Erfahrung in der Softwareentwicklung und im Requirements Engineering hat er sich auf agile Ansätze spezialisiert. Er coacht Entwicklungsteams, Scrum Master, Product Owner und Organisationen auf ihrem Weg in die Agilität.

Agiles Manifest für Manager

Agile Entwicklung ist im Kern keine Methode, keine Technik und auch kein Framework. Sie ist eine Denkweise! Diese Denkweise manifestiert sich im Agilen Manifest. Der Artikel untersucht die Bedeutung des Manifests für und die Auswirkungen auf das Management. Weiterhin wollen wir zeigen, dass die Werte und Prinzipien des Agilen Manifests weit mehr sind als nur gesunder Menschenverstand für Softwareentwickler.

Die Bedeutung des Agilen Manifests für Entwicklungsteams ist mittlerweile hinreichend klar, denn die Auswirkungen auf Teams werden immer besser verstanden und die dahinterstehende neue Denkweise wird weltweit in vielen Entwicklungsvorhaben sehr erfolgreich gelebt. Eine agile Transition ist jedoch nicht nur eine weitere Prozessänderung, die allenfalls die IT betrifft und die womöglich auch noch delegiert werden kann. Das wahre Potenzial des agilen Vorgehens entfaltet sich erst dann richtig, wenn die agile Denkweise die ganze Organisation erfasst, insbesondere das Management.

Worin besteht die „agile Denkweise“?

Diese Frage wurde 2001 von 17 Koryphäen der Softwareentwicklung im Agilen Manifest beantwortet [Agile]. Die Unterzeichner merken in den einleitenden Sätzen an, dass auch sie nicht auf alles eine Antwort besitzen („uncovering“), dass aber die kontinuierliche Verbesserung auf Basis der Werte und Prinzipien des agilen Manifests zum Erfolg führt („by doing it“).

Der Kern des agilen Manifests sind vier Wertepaare und zwölf Prinzipien. Um sie geht es in diesem Artikel.

Die vier Wertepaare

Value #1: “Individuals and interactions *over* processes and tools”

Für die meisten Organisationen sind bis heute Prozesse und Tools wichtiger als Individuen und Interaktionen: „Wenn wir den richtigen Prozess definieren und alle Mitarbeiter diesen genau befolgen, dann sind wir erfolgreich“. Dieses Vorgehen würde funktionieren, wenn jeder Mensch identisch

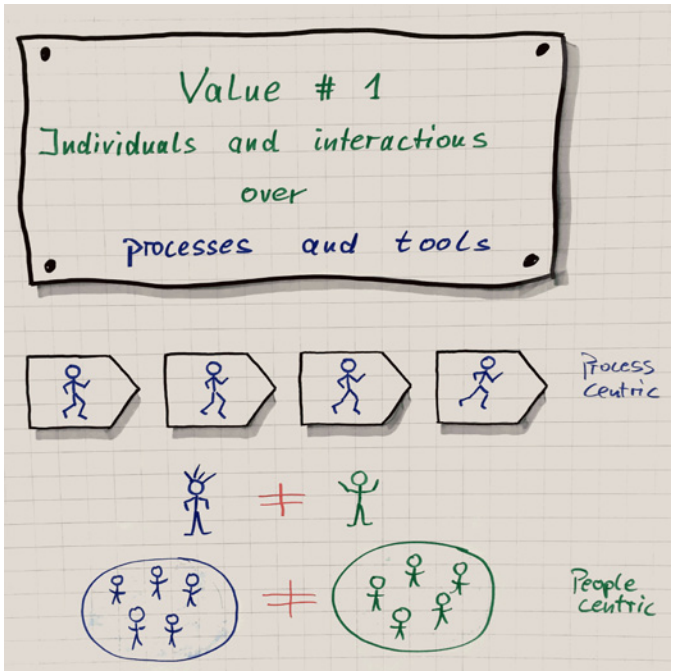

Abb. 1: Value #1.

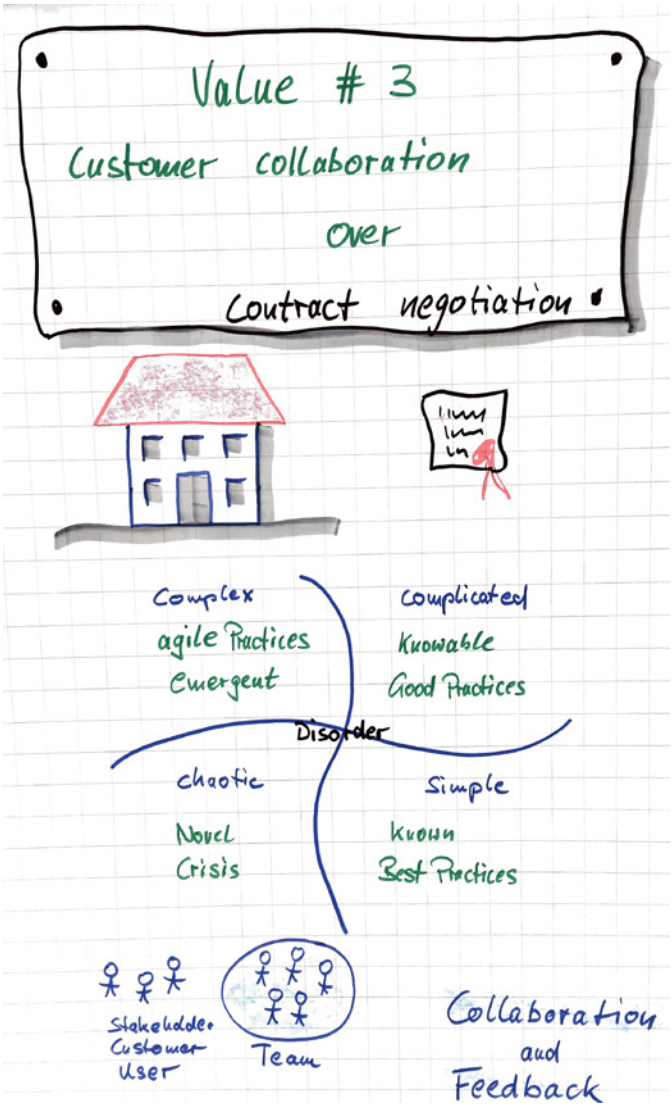

Abb. 2: Value #3.

ist. Identisch denkt und identisch handelt.

Aber Softwareentwicklung fordert die Kommunikation und Kreativität von Individuen. Sie wird von Wissensarbeitern durchgeführt, die komplexe Probleme aus verschiedenen Blickwinkeln betrachten müssen.

Da Teams aus Individuen bestehen, sind sie damit auch verschieden. Nicht alle Teams können gleich behandelt werden. Zwar müssen alle Teams die Philosophie von agiler Entwicklung teilen, aber schon die Basismethoden (Scrum, Kanban usw.) können unterschiedlich sein.

Dieses Wertepaar fordert das Management also dazu auf, die Menschen ins Zentrum zu stellen und nicht die Prozesse.

Value #2: "Working software *over* comprehensive documentation"

Das zweite Wertepaar ist das am häufigsten missverstandene. Es bedeutet keineswegs, dass Dokumentation im agilen Umfeld unwichtig ist. Die Dokumentation sollte nur weniger wichtig sein als das Ergebnis, die Software.

Ganz ohne Spezifikation, wie sie sich verhalten soll, kann Software wahrscheinlich nicht so entwickelt werden, dass sie den Erwartungen der Stakeholder gerecht wird. Die Sachverhalte in der Softwareentwicklung sind teilweise zu kompliziert, um sie nur verbal zu kommunizieren und zu erinnern. Aber was nützt eine umfangreiche Spezifikation, wenn die Software nicht funktioniert? Was nützt eine früh im Projekt erstellte Spezifikation noch, wenn sie nicht mehr aktuell ist, wenn die Entwicklung beginnt?

Die agile Denkweise fordert eine leichtgewichtige Just-in-time-Spezifikation, beispielsweise User Stories mit Akzeptanzkriterien. Die Balance zwischen dem erzeugten Wert (der Software) und der dafür notwendigen Dokumentation muss vom Management verstanden, gefordert und gefördert werden.

Value #3: „Customer collaboration *over* contract negotiation"

Kunden bezahlen den Kauf von Produkten. Deshalb möchten sie berechtigterweise auch eine Liefervereinbarung wie einen Vertrag. Dabei gibt es aber einige problematische Aspekte, je nachdem, um welche Art von Produkt es sich handelt.

Zum Beispiel ist ein Haus zu bauen vergleichsweise einfach, ein Hochhaus wahrscheinlich kompliziert. Aber in beiden Fällen ist die Lösung bekannt oder sie kann durch Analyse und Berechnungen festgestellt werden. Daher kann verlässlich geplant werden.

Softwareentwicklung ist jedoch ein komplexes Vorhaben. Solche Vorhaben sind nicht-linear, denn in ihnen gibt es keine einfachen Ursache-Wirkung-Beziehungen. Es ist daher unmöglich, die Wirkung einer Änderung exakt vorauszusagen.

Bei komplexen Vorhaben ist der Versuch, die Anforderungen detailliert im Voraus vertraglich zu fixieren, nicht nur Verschwendung, sondern wahrscheinlich sogar kontraproduktiv. Stattdessen sollte das Management zusammen mit dem Kunden und der Entwicklung eine gemeinsame Vision erarbeiten und als Partner zusammenarbeiten, um über Feedback auftauchende Probleme zu adressieren.

Value #4: "Responding to change *over* following a plan"

Das vierte Wertepaar adressiert Änderungen. Eine stabile, vorhersagende Planung hängt von stabilen Anforderungen ab. Aber die Anforderungen in komplexen Vorhaben ändern sich stetig. Das Management muss dafür sorgen, dass der Entwicklungsprozess auf häufige Änderungen ausgelegt ist. Für die Tätigkeit des Managements bedeutet das, adaptiv und kontinuierlich zu planen und die Planung auf Basis von regelmäßigem Feedback fortwährend anzupassen, statt blind an einem initialen Projektplan festzuhalten.

Die zwölf Prinzipien

Principle #1: "Our highest priority is to satisfy the customer through early and continuous delivery of valuable software"

Das erste Prinzip definiert das Ziel für Organisationen auf ihrem Weg in die Agilität. Das Ziel ist es, dem Kunden etwas für ihn Wertvolles zu liefern – das ist eine klare Anforderung.

Um zu wissen, was für den Kunden wirklich wertvoll ist, ist dessen Feedback nötig. Frühes und kontinuierliches Liefern

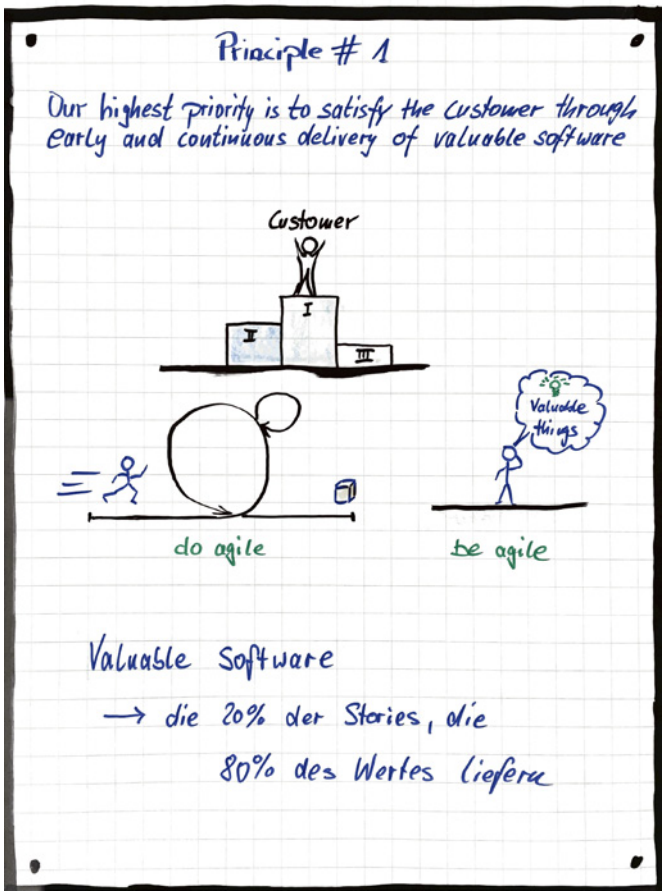

Abb. 3: Principle #1.

führt zu frühem und regelmäßigem Feedback. Die wertvolle Software entsteht aus den 20 Prozent der Anforderungen, die 80 Prozent des Wertes liefern. Nämlich aus den Anforderungen, die den Kunden am meisten begeistern.

Es liegt in der Verantwortung des Managements, die Ziele der eigenen Organisation mit den Wünschen der Kunden und dem Wert, der an die Kunden geliefert werden soll, in Einklang zu bringen.

Principle #2: "Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage."
Die Welt dreht sich. Die Softwareentwicklungswelt dreht sich noch viel schneller. Langfristige Planung mag einen in Sicherheit wiegen, bringt aber nicht die notwendige Flexibilität, um wettbewerbsfähig zu bleiben. Wenn auf Änderungen schneller als die Konkurrenz reagiert werden kann, dann wurde der Wandel zu einem Wettbewerbsvorteil. Es ist effektiver, Änderungen zu erleichtern, als zu versuchen, sie zu verhindern.

Aber agil ist keine Entschuldigung für unorganisiertes Vorgehen! Wenn zum Beispiel das Management ohne Rücksprache entscheidet, neue und unbekannte Anforderungen sofort umsetzen zu lassen, führt das ins Chaos. Das Entwicklungsteam muss das zurückweisen dürfen, und darauf bestehen, dass erst Gespräche über diese neuen Anforderungen stattfinden.

Das Management muss die Balance zwischen Chaos und Bürokratie, zwischen dem Abgeben von Verantwortung an Teams und dem Zusammenhalten des Ganzen halten.

Principle #3: "Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale."
Das dritte Prinzip adressiert das schnelle Liefern und damit die Möglichkeit, schneller am Markt zu sein. Funktionierende Software schnell zu liefern, ist technisch und organisatorisch schwierig, aber kritisch für den Erfolg eines Unternehmens in dynamischen Märkten. Auch wenn ein Team darin nicht gut ist, bringt es sehr viel, zu wissen, warum dies so ist. Es geht auch nicht nur um agile Entwicklungsteams, sondern um eine komplette agile Organisation – es geht um die Lieferung, nicht nur um die Entwicklung.

Das Management ist dafür verantwortlich, die notwendigen organisatorischen Änderungen durchzuführen, um eine schnelle Lieferung zu ermöglichen.

Principle #4: „Business people and developers must work together daily throughout the project."
Das vierte Prinzip ist glasklar formuliert: Die tägliche Kommunikation zwischen Fach-

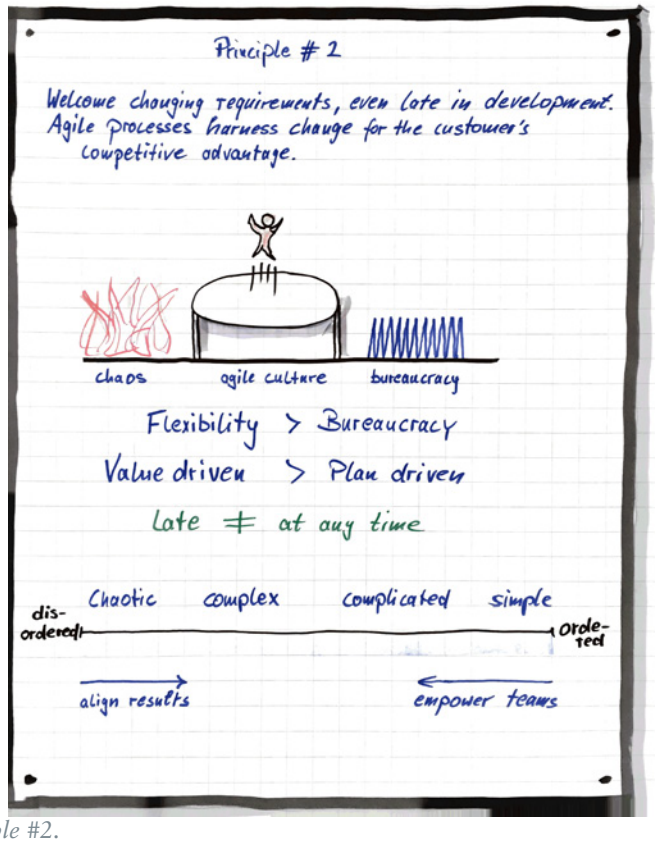

Abb. 4: Principle #2.

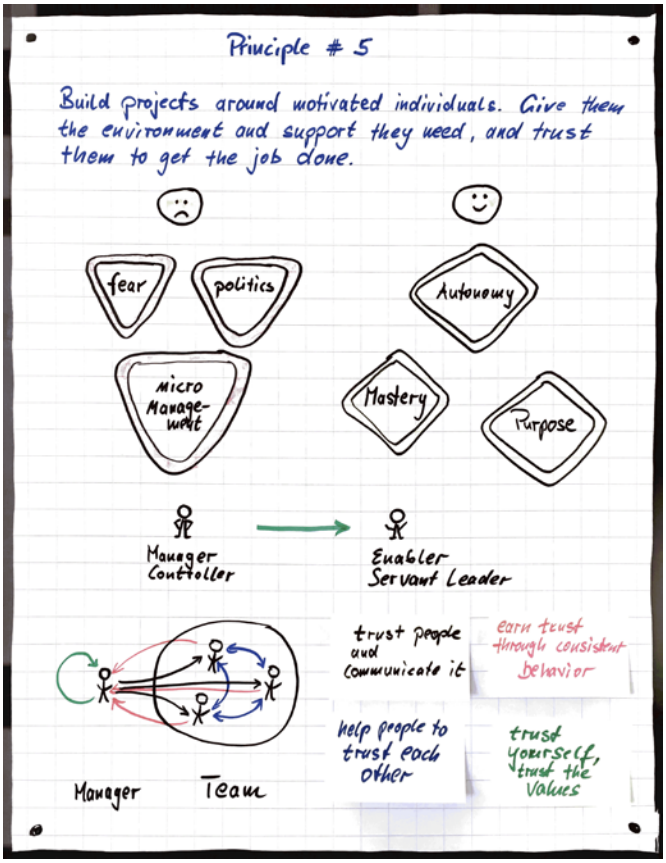

Abb. 5: Principle #5.

experten und IT ist essenziell. Probleme bei dieser Kommunikation sind der Hauptgrund für nicht erfolgreiche Projekte und ebenso für Probleme bei agilem Vorgehen. Es ist Aufgabe des Managements, diese kontinuierliche Kommunikation zu fordern und zu unterstützen.

Principle #5: "Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done."
Was motiviert Wissensarbeiter? Dan Pink nennt drei notwendige Dinge, damit Menschen intrinsisch motiviert sind [Pin11]:

- *Autonomy:* Mitarbeiter dürfen weitgehend selbst bestimmen, wie sie ihre Arbeit erledigen.
- *Mastery:* Mitarbeiter müssen die Möglichkeit haben zu lernen, um ihre Arbeit zu meistern und an ihr zu wachsen.
- *Purpose:* Mitarbeiter müssen Sinn und Zweck ihrer Arbeit verstehen und unterstützen.

Den Unterschied zwischen Erfolg und Misserfolg machen die Mitarbeiter.

Das Management muss dafür sorgen, dass die Mitarbeiter Zeit haben zu denken, zu planen, zu lernen, Beziehungen zu knüpfen und zu pflegen und die Kunden besser kennenzulernen. Das mittlere Management muss als „Servant Leaders" dafür sorgen, dass die Teams die Umgebung und die Unterstützung haben, die sie für ihre Arbeit brauchen. Das Topmanagement muss Enabler sein, es muss dem mittleren Management ermöglichen, unterstützend zu führen.

Principle #6: „The most efficient and effective method of conveying information to and within a development team is face-to-face conversation."
Das sechste Prinzip definiert den Goldstandard der Kommunikation. Die informationsreichste Art der Kommunikation ist das Gespräch von Angesicht zu Angesicht. Die Prinzipien #4 (Häufigkeit der Kommunikation) und #6 (Qualität der Kommunikation) können zu einem tiefen Verstehen dessen, was gemeinsam entwickeln werden soll, und damit zum Erfolg führen.

Principle #7: „Working Software is the primary measure of progress."
Es liegt in der Natur des Menschen, zu versuchen, alles zu messen. Dummerweise wird das, was gemessen wird, meistens auch gemacht. Die wirklich wichtigen Dinge können aber nicht gemessen werden, da sie zu komplex sind. Alle an einem Ent-

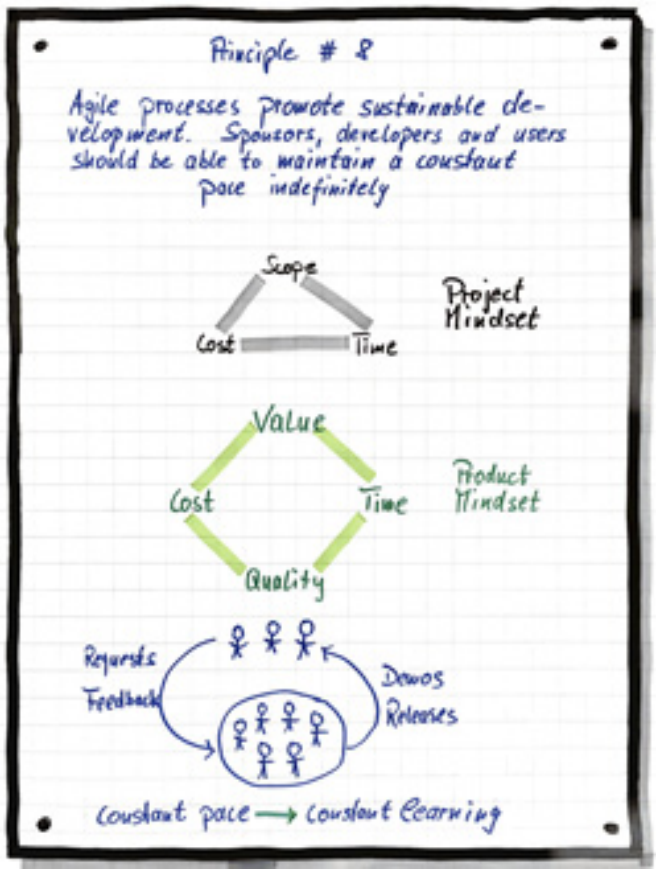

Abb. 6: Principle #8.

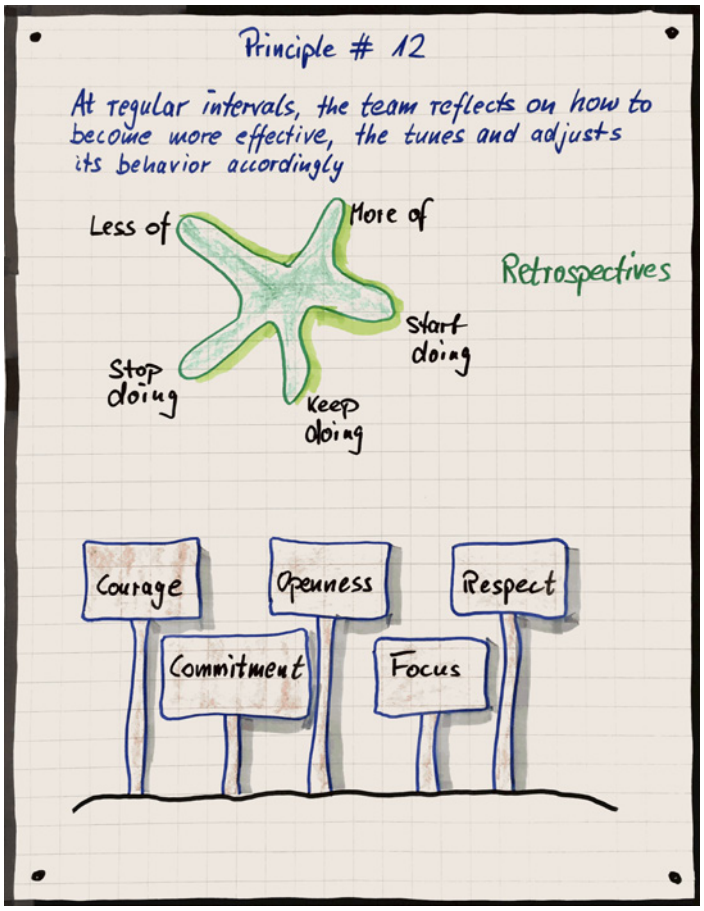

Abb. 7: Principle #12.

wicklungsvorhaben Beteiligten brauchen eine gemeinsame Sprache, um über den Fortschritt ihrer Bemühungen zu sprechen.

Das siebte Prinzip definiert lauffähige Software als diese gemeinsame Sprache und eben nicht Zahlen („80 Prozent der Spezifikation sind fertig", „das Design ist zu 90 Prozent fertig" usw.).

Principle #8: "Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely."
Für ein erfolgreiches agiles Vorgehen ist Nachhaltigkeit wichtig. Es ist nicht agil, nur die drei Seiten des eisernen Dreiecks („Scope, Cost und Time") zu berücksichtigen und die Qualität zu ignorieren. Wenn Geschwindigkeit über Qualität gestellt wird, erhöht sich die technische Schuld. Agil erweitert das Dreieck und verankert die Qualität als nicht verhandelbar im goldenen Viereck.

Um „indefinitely constant pace" erreichen zu können, müssen Praktiken wie automatisiertes Testen und Continuous Integration gemeistert werden. Es müssen die Fähigkeiten für nachhaltige Entwicklung aufgebaut und gepflegt werden. Die Organisation und das Management müssen hierfür den Weg ebnen.

Principle #9: „Continuous attention to technical excellence and good design enhances agility."
Das neunte Prinzip ist eine Erweiterung des achten. Es wird nicht an der Qualität gespart, um schneller zu werden. Gerade wenn die Qualität ernst genommen wird und die technischen Schulden verhindert werden, ist es möglich, schneller zu werden.

Komplexe Vorhaben sind nicht-linear und kleine Änderungen können unvorhersehbare und weitreichende Auswirkungen haben. Durch „Just-in-time"-Anforderungen statt umfangreicher Spezifikationen und emergentes Design statt Big Upfront Design kann Nicht-Linearität beherrscht werden. Das Unternehmen muss lernen, wie testbare Anforderungen erzeugt (Requirements Engineering, Behaviour Driven Development) und wie vor dem Codieren getestet werden kann (Test-Driven Development, TDD). Dazu wird schnelles Feedback darüber benötigt, ob die Software als Ganzes noch funktioniert wie gewünscht (Continuous Integration, CI, Continuous Delivery, CD). Technische Exzellenz muss aufgebaut und gepflegt werden.

Principle #10: Simplicity – the art of maximizing the amount of work not done – is essential.
Maximiere die Menge an Arbeit, die nicht gemacht werden muss! Was für einen Sinn ergibt es, schneller und besser in der Lieferung zu werden, wenn Softwarefunktionen geliefert werden, die nicht die Bedürfnisse der Kunden erfüllen?

Die in diesem Prinzip geforderte Einfachheit sollte aber nicht nur für die Auswahl der Softwarefunktionen, sondern auch für die Implementierung derselben gelten. Durch Test-Driven Development wird nur der Code geschrieben, der notwendig ist, und Behavior Driven Development sorgt für Einfachheit auf der Codeebene.

Einfachheit gilt auch auf der Prozessebene! Es ist viel effizienter, zu einem einfachen Prozess etwas hinzuzufügen, als von einem zu komplizierten Prozess etwas wegzunehmen.

Principle #11: „The best architectures, requirements and designs emerge from self-organizing teams."
Das elfte Prinzip basiert auf der Erfahrung der Verfasser des Manifests: Die bessere Software wird von selbstorganisierten Teams geschrieben.

Selbstorganisierte Teams sind fähig, sich als Antwort auf Störungen neu zu organisieren und die Störung zu beseitigen. Trotzdem wird in den meisten Organisationen die Arbeit auf den Managementebenen identifiziert, analysiert und in spezialisierte Aufgaben heruntergebrochen, die dann an spezialisierte Rollen zur Ausführung übergeben werden. Die Erfahrung zeigt aber, dass die Teams, die die Arbeit machen, am besten geeignet sind, die Entscheidungen bezüglich Architektur, Anforderungen und Design zu treffen.

Principle #12: At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.
Kontinuierliche Verbesserungen sind nur erreichbar, wenn die Teams die Möglichkeit haben, zu experimentieren. Aber Experimente können nicht von oben für alle Teams vorgeschrieben werden, denn jedes Team ist anders. Was für ein Team funktioniert, muss nicht auch für ein anderes Team funktionieren.

Beispielsweise kann Pair Programming in einem Team sehr gut funktionieren,

ein anderes ist dafür aber vielleicht noch nicht reif genug. Daher muss das Management sicherstellen, dass die Teams experimentieren können. Unterstützen und fordern Sie regelmäßige Retrospektiven. Leben Sie Werte wie Mut, Offenheit und Respekt vor.

Zusammenfassung

Das Agile Manifest ist eine Aufgabenliste für Manager, die ihre Organisation in die agile Welt führen möchten. Schwierigkeiten auf diesem Weg können immer auf Verstöße gegen die Werte oder Prinzipien des Manifests zurückgeführt werden. Ein reichhaltiger Fundus an agilen Praktiken kann helfen, diese Schwierigkeiten zu meistern. ■

Literatur & Links

[Agile] Manifesto for Agile Software Development (Agiles Manifest), siehe: agilemanifesto.org

[Pin11] D. H. Pink, The Surprising Truth About What Motivates Us, Riverhead Books, 2011, siehe: http://www.danpink.com/books/drive/