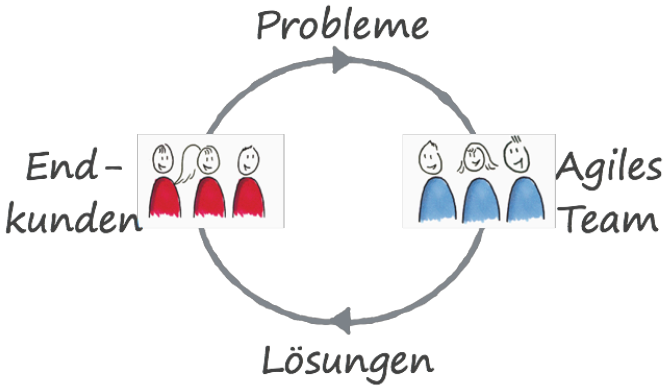
Agilität – eine Einführung

Begriffe, Konzepte, Grundverständnis



Februar 2019, Version 5

Über diesen Artikel

Dieser Artikel führt in Agilität ein. Er erläutert die grundlegenden Begriffe und Konzepte sowie ihr Zusammenwirken. Die typischen agilen Mechaniken sind nur die eine Seite der Medaille. Die dahinterstehenden Werte und Prinzipien sind mindestens genauso wichtig.

Dieser Artikel erläutert zunächst, dass und warum agiles Denken und Arbeiten immer wichtiger wird. Anschließend stellt er die agilen Kernideen und die agilen Werte (das agile Mindset) vor. Den Abschluss bilden die verbreiteten methodischen Ausprägungen Scrum und Kanban.

Der Artikel hilft, sich einen ersten Überblick über Agilität zu verschaffen und die wichtigsten Begriffe zu verstehen. Dieses Grundverständnis dient als Orientierungshilfe für die weitere Vertiefung.

Stefan Roock

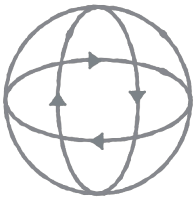
Coach und Trainer bei it-agile

stefan.roock@it-agile.de, Tel. 0172/429 76 17

Inhalt

Warum agil?

Die Welt wird komplexer: Globalisierung, Digitalisierung und nicht zuletzt die gewachsenen Anforderungen von Menschen an ihre Arbeit erhöhen die Komplexität der Welt für Unternehmen.



Die **Globalisierung** hat ehemals voneinander isolierte Märkte miteinander vernetzt. Unternehmen adressieren nicht nur einzelne regionale Märkte, sondern sind weltweit aktiv. Dadurch müssen die Unternehmen mit den vielen verschiedenen Gegebenheiten der einzelnen Märkte umgehen, die trotz ihrer Vernetzung nach wie vor existieren.

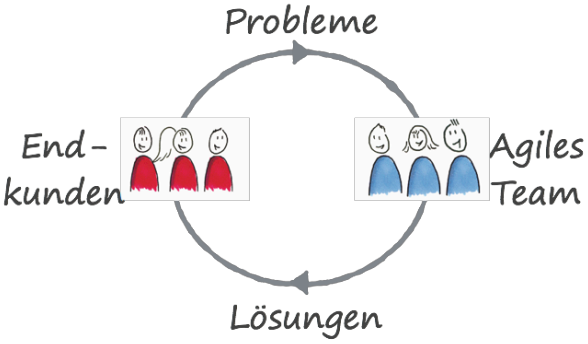
Die **Digitalisierung** bedroht die althergebrachten Geschäftsmodelle. In vielen Branchen ist die Bedrohung noch nicht konkret greifbar: Man weiß noch nicht, wie genau das digitale Geschäftsmodell aussieht, dass die existierenden Geschäftsmodelle ablösen wird. Früher oder später werden diese digitalen Geschäftsmodelle sichtbar werden. Spätestens dann müssen die Unternehmen sehr schnell reagieren, wenn sie nicht unter die Räder geraten wollen.

Die **Anforderungen der Menschen an ihre Arbeit** sind gestiegen (mind. in den Industrienationen). Montags bis freitags von 9-17 Uhr zur Arbeit zu gehen und als Gegenleistung ein Gehalt zu erhalten, ist immer mehr Menschen zu wenig. Sie wünschen sich Sinn und die Möglichkeit zur Entfaltung in der Arbeit. Unternehmen müssen den geänderten Bedürfnissen der Menschen gerecht werden, wenn sie auch in Zukunft qualifizierte und motivierte Mitarbeiter an sich binden wollen.

Agilität löst keine dieser Herausforderungen direkt. Sie kann Unternehmen aber zu der Flexibilität und Dynamik verhelfen, die sie benötigen, um der gewachsenen Komplexität gerecht zu werden.

Agile Kernideen

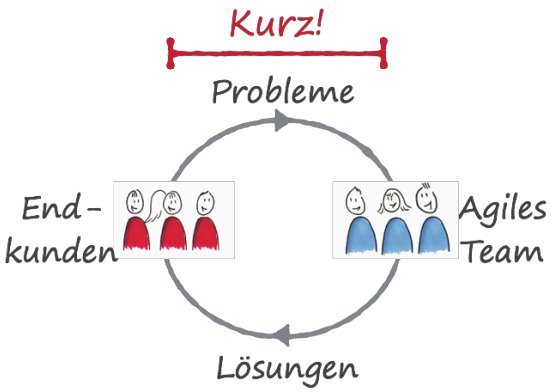
Die wichtigsten agilen Ideen sind sehr einfach. Letztlich geht es darum, dass agile Teams kundenrelevante Probleme lösen.



Dazu nutzen die agilen Teams den direkten Kundenkontakt. Sie verstehen so die Probleme und Bedürfnisse ihrer Kunden. Für diese Probleme finden agile Teams angemessene Lösungen und liefern diese an die Kunden.

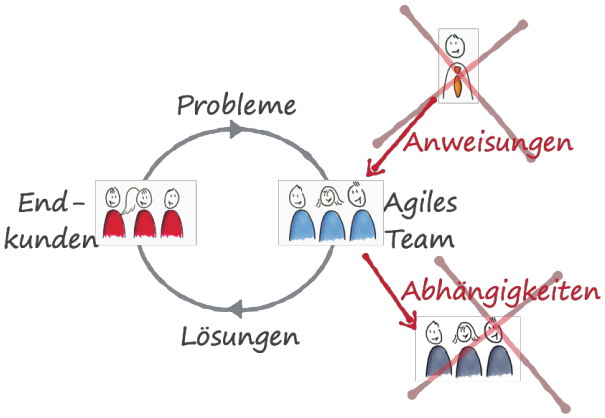
Ein Gradmesser für Agilität ist damit die **Direktheit der Kundeninteraktion**. Idealerweise spricht das Team direkt mit den Endkunden (und nicht vermittelt über Front-Office-Mitarbeiter, Anforderungsmanager, Product Owner o.Ä.). Außerdem liefert das Team direkt die Lösung an Kunden und muss dabei nicht über Dritte gehen, wie z.B. Account Manager oder Qualitätssicherung.

Hohe Marktdynamik erfordert **schnelle Reaktion**. Daher sollte der beschriebene Problem-Lösungszyklus möglichst schnell und häufig durchlaufen werden. Der Kunde sollte nicht Monate auf seine Lösung warten, sondern diese so schnell wie möglich erhalten. Wenn z.B. eine größere Software entwickelt werden muss, die nicht nach wenigen Wochen für den operativen Einsatz an den Kunden ausgeliefert werden kann, sollten dem Kunden mind. Zwischenstände gezeigt und Feedback eingeholt werden.



Durch die kurzen Zyklen wird das Lernen über Kundenprobleme, mögliche Lösungen sowie das beste Vorgehen beschleunigt.

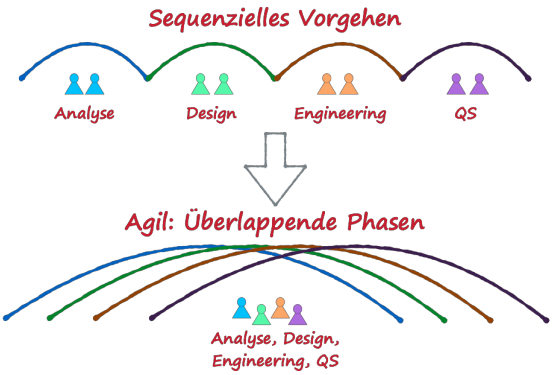
Das agile Team agiert dabei **autonom und selbstorganisiert**. Dies betrifft eingehende und ausgehende Beziehungen zu Dritten. Es erhält keine Anweisungen von außen, wie es sich zu organisieren hat. Außerdem ist es nicht von anderen Teams abhängig. Es kann die Kundenprobleme lösen, ohne auf andere Teams oder Abteilungen warten zu müssen.



Damit das Team kundenrelevante Probleme schnell und autonom umsetzen kann, muss es eine große Bandbreite an Fähigkeiten besitzen: das agile Team ist **interdisziplinär** besetzt. Welche Spezialisierungen im Team konkret notwendig sind, hängt von den Kunden und ihren Problemen ab. Ein Software-Entwicklungsteam könnte z.B. aus einem User-

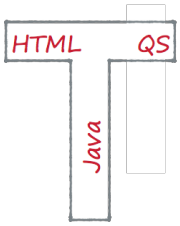
Experience-Designer, einem programmierenden Architekten, Frontend-Entwicklern, Backend-Entwicklern und Testern bestehen. In einem Übersetzerteam könnten beispielsweise Experten für die Zielsprachen arbeiten, z.B. Englisch, Französisch, Spanisch. Und in beiden Fällen könnte man auch auf die Idee kommen, Key-Account-Manager oder Vertriebler ins Team zu integrieren.

Das agile Team weicht von der klassischen sequenziellen Phasen-Organisation ab. Stattdessen organisiert es seine Arbeit so, dass sich die klassischen **Phasen überlappen**: Analyse, Design, Engineering und Qualitätssicherung werden nicht strikt nacheinander durchgeführt, sondern gleichzeitig.

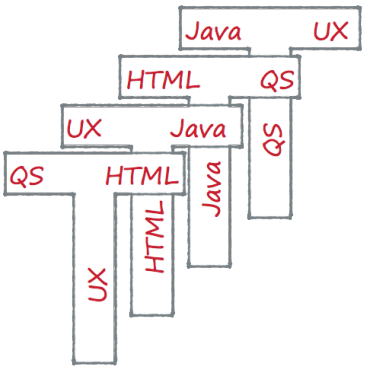


Wie stark sich die Phasen überlappen, hängt vom Kontext ab. Generell gilt: je dynamischer das Umfeld, desto stärker sollten sich die Phasen überlappen: im Extremfall führt das Team alle Phasen gleichzeitig aus. In Kontexten, die nicht ganz so dynamisch sind, reicht mitunter eine eingeschränkte Überlappung aus: eine Phase beginnt, bevor die vorhergehende Phase komplett abgeschlossen ist.

Damit ein interdisziplinäres Team gut als Team zusammenarbeiten kann, sind sogenannte **T-Shaped-Skill-Sets** notwendig. Die Spezialisierungen (z.B. Java-Programmierung) der Teammitglieder soll und muss erhalten bleiben. Gleichzeitig ist es meist aber sinnvoll, dass die Teammitglieder zusätzliche Fähigkeiten erwerben, die im Team benötigt werden (z.B. HTML-Programmierung und Qualitäts–sicherung).



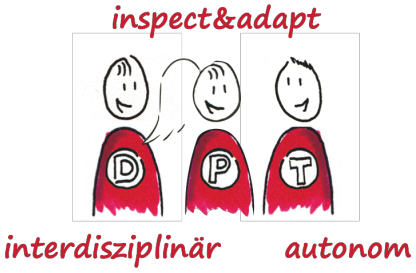
In diesen zusätzlichen Disziplinen werden die Teammitglieder nicht so effizient werden wie die jeweiligen Spezialisten. Sie können aber durch die überlappenden Skill-Sets dazu beitragen, dass die **Arbeit im Fluss** bleibt und nicht ins Stocken gerät, weil eine akute Engpass-Situation (z.B. beim Testen) existiert.



Die T-Shaped-Skill-Sets müssen nicht vorab existieren oder zentral geplant werden. Sie bilden sich von selbst in dem benötigten Maß heraus, wenn das Team auf eine Endpass-Situation mit Ausbildung (engl. Cross-Skilling) reagiert. Eine sehr effektive Technik für den Fähigkeitserwerb ist das Arbeiten in Paaren (engl. Pairing): ein Experte und ein „Neuling" bearbeiten eine Aufgabe gemeinsam.

Für das agile Team können wir zusammengefasst festhalten:

- Es ist autonom und arbeitet selbst–organisiert.
- Es ist interdisziplinär besetzt.
- Es verbessert Ergebnis und Prozess über Inspect & Adapt.

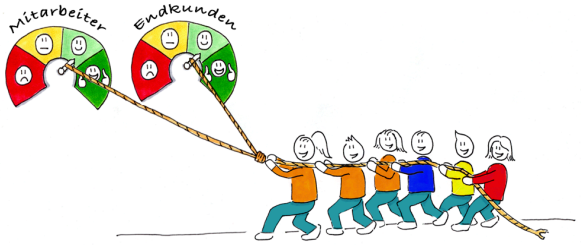


Agilität lässt sich samit in einem Satz beschreiben:

Autonome Teams mit Business-Fokus, die ihren Prozess in Besitz und Verantwortung nehmen.

Vorteile von Agilität

Agile Entwicklung bringt eine Reihe von Vorteilen mit sich:



- Durch die überlappenden Phasen verkürzt sich die **Time-To-Market**.
- Durch das inkrementell-iterative Vorgehen kann das Team schnell auf veränderte Rahmenbedingungen **reagieren**.
- Die größere Nähe des Teams zu den Endkunden sowie das iterative Vorgehen führen zu **angemessenere Lösungen** für die Kunden.
- Die große Diversität des cross-funktionalen Teams erhöht die Wahrscheinlichkeit von **Innovation** und führt damit zu erfolgreicheren Produkten.
- Die Mitglieder des Teams erleben durch den direkten Kundenkontakt und das Kundenfeedback mehr Sinn in ihrer Arbeit. Das wirkt **motivierend**.
- Für Sponsoren wirkt die Priorisierung nach Geschäftswert **risikominimierend**. Das Projekt kann fast jederzeit beendet werden und liefert trotzdem einen **Mehrwert**.
- Sponsoren bekommen eine größere **Planungssicherheit**, weil Restaufwände auf Basis bereits entwickelter Lösungs–eigenschaften prognostiziert werden können.

Das agile Manifest

Das *Agile Manifest*[1] definiert das agile Wertesystem. Es stammt aus der Software-Entwicklung, lässt sich jedoch universell anwenden auch auf die Entwicklung physischer Produkte oder das Erbringen von Dienstleistungen.

Das agile Manifest beginnt mit einer Präambel, gefolgt von vier Wertaussagen:

„Wir erschließen bessere Wege, Software zu entwickeln, indem wir es selbst tun und anderen dabei helfen.

Durch diese Tätigkeit haben wir diese Werte zu schätzen gelernt:

- Individuen und Interaktionen sind wichtiger als Prozesse und Tools
- Laufende Software ist wichtiger als ausführliche Dokumentation
- Zusammenarbeit mit dem Kunden ist wichtiger als Vertragsverhandlungen
- Reagieren auf Veränderungen ist wichtiger als Planbefolgung

Das heißt, obwohl wir die Werte auf der rechten Seite wichtig finden, schätzen wir die Werte auf der linken Seite höher ein."

Die wenigen expliziten Referenzen auf Software-Entwicklung lassen sich leicht auf jegliche Entwicklung und Leistungserbringung verallgemeinern (Verallgemeinerungen sind *kursiv* gesetzt):

„Wir erschließen bessere Wege, *Wert für Kunden zu schaffen*, indem wir es selbst tun und anderen dabei helfen.

Durch diese Tätigkeit haben wir diese Werte zu schätzen gelernt:

- Individuen und Interaktionen sind wichtiger als Prozesse und Tools
- *Wertschöpfung für Kunden* ist wichtiger als ausführliche Dokumentation
- Zusammenarbeit mit dem Kunden ist wichtiger als Vertragsverhandlungen
- Reagieren auf Veränderungen ist wichtiger als Planbefolgung

Das heißt, obwohl wir die Werte auf der rechten Seite wichtig finden, schätzen wir die Werte auf der linken Seite höher ein."

In klassischen Kontexten generieren die Dinge auf der rechten Seite *subjektiv wahrgenommene Sicherheit*. Wer sich an die Prozesse hält und die vorgeschriebenen Tools einsetzt, wer jede seiner Tätigkeiten detailliert dokumentiert, wer alle Eventualitäten in Verträgen berücksichtigt und wer sich an den Plan hält, kann bei Problemen nachweisen, dass er nicht schuld ist. Leider generieren wir auf diese Weise in

Original-Wertaussagen aus dem Agilen Manifest

Individuen und Interaktionen	wichtiger als	Prozesse und Tools
Laufende Software	wichtiger als	Ausführliche Dokumentation
Zusammenarbeit mit dem Kunden	wichtiger als	Vertrags-verhandlungen
Reagieren auf Veränderungen	wichtiger als	Planbefolgung

Verallgemeinerte Wertaussagen

Individuen und Interaktionen	wichtiger als	Prozesse und Tools
Wertschöpfung für Kunden	wichtiger als	Ausführliche Dokumentation
Zusammenarbeit mit dem Kunden	wichtiger als	Vertrags-verhandlungen
Reagieren auf Veränderungen	wichtiger als	Planbefolgung

[1] http://agilemanifesto.org

komplexen Märkten keinen Geschäftswert. In dynamischen Märkten brauchen wir die Flexibilität, die uns die Dinge auf der linken Seite geben.

Dieser Gegensatz erklärt, warum die Einführung agiler Arbeits- und Denkweisen in der Praxis häufig so schwierig ist. Alle Beteiligten müssen „Sicherheit durch Statik“ loslassen, um auf den Kunden und den Geschäftswert fokussieren zu können.

Die Prinzipien des agilen Manifests

Das agile Manifest ergänzt die vier Wertaussagen durch zwölf Prinzipien, die konkretisieren, wie die Werte sich auf die tägliche Arbeit auswirken. Die Prinzipien enthalten ebenfalls einige wenige explizite Referenzen auf Software-Entwicklung. Diese lassen sich ähnlich leicht verallgemeinern wie schon bei den Wertaussagen.

Die verallgemeinerten Prinzipien lauten wie folgt (Verallgemeinerungen sind wieder *kursiv* gesetzt).

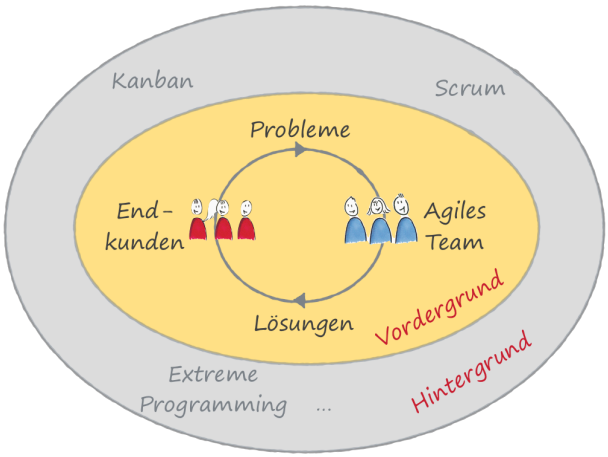
1. Unsere höchste Priorität ist es, den Kunden durch frühe und kontinuierliche *Wertschöpfung* zufrieden zu stellen.
2. Heiße Anforderungsänderungen selbst spät *im Prozess* willkommen. Agile Prozesse nutzen Veränderungen zum Wettbewerbsvorteil des Kunden.
3. Liefere regelmäßig und *in möglichst kurzen Abständen Wert an den Kunden.*
4. *Kunden und Team* müssen täglich zusammenarbeiten.
5. Errichte *Strukturen (Projekte, Teams etc.)* rund um motivierte Individuen. Gib ihnen das Umfeld und die Unterstützung, die sie benötigen und vertraue darauf, dass sie die Aufgabe erledigen.
6. Die effizienteste und effektivste Methode, Informationen an und innerhalb eines *Teams* zu übermitteln, ist im Gespräch von Angesicht zu Angesicht.
7. *Geschaffener Kundenwert* ist das wichtigste Fortschrittsmaß.
8. Agile Prozesse fördern nachhaltiges *Arbeiten*. *Alle Beteiligten* sollten ein gleichmäßiges Tempo auf unbegrenzte Zeit halten können.
9. Ständiges Augenmerk auf *operative* Exzellenz fördert Agilität.
10. Einfachheit - die Kunst, die Menge nicht getaner Arbeit zu maximieren - ist essenziell.
11. Die besten *Ergebnisse bei anspruchsvoller Arbeit* entstehen durch selbstorganisierte Teams.
12. In regelmäßigen Abständen reflektiert das Team, wie es effektiver werden kann und passt sein Verhalten entsprechend an.

Methodische Ausgestaltungen agilen Arbeitens

Es gibt eine ganze Reihe methodischer Ansätze für agiles Arbeiten. Am weitesten verbreitet sind Scrum und Kanban, die wir in der Folge kurz skizzieren.

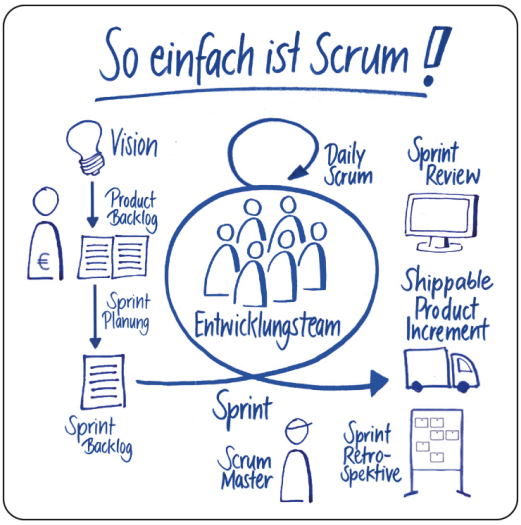
Aber auch aus anderen methodischen Ansätzen wie Extreme Programming, Feature Driven Development oder Naked Planning lassen sich Inspirationen für das eigene agile Arbeiten ziehen.

Welcher konkrete methodische Ansatz gewählt werden sollte, hängt vom eigenen Kontext ab. Wichtig ist, dass die oben beschriebenen Kernideen stets im Vordergrund bleiben und nicht durch die dogmatische Verwendung einer konkreten Herangehensweise behindert werden.



Agile Entwicklung mit Scrum

Scrum ist *ein* methodischer Ansatz für agiles Arbeiten. Scrum fokussiert dabei auf die Entwicklung innovativer Produkte oder Services[2]. Bei geringen Innovationsanforderungen in der Entwicklung (z.B. reines Bugfixing) oder bei der Erbringung von Services (z.B. Texte übersetzen) ist Scrum häufig nicht die beste Wahl.



Wir haben oben agiles Arbeiten auf einen prägnanten Satz gebracht:

Autonome Teams mit Business-Fokus, die ihren Prozess in Verantwortung nehmen und kontinuierlich verbessern.

Dieser Satz gilt genauso auch für Scrum. In dem Satz werden drei Perspektiven sichtbar, aus denen man Scrum betrachten kann:

1. Die *Produktperspektive* (Business-Fokus) beleuchtet, wie Produkte definiert und verbessert werden.
2. Die *Entwicklungsperspektive* (autonome Entwicklungsteams) beleuchtet, wie Teams Produkte entwickeln.
3. Die Verbesserungsperspektive (Prozess in Verantwortung nehmen) beleuchtet, wie Zusammenarbeit und Prozesse verbessert werden.

Diese drei Perspektiven werden im Scrum-Framework integriert, das so einfach ist, dass es auf einen Bierdeckel passt (siehe Abbildung rechts)[3].

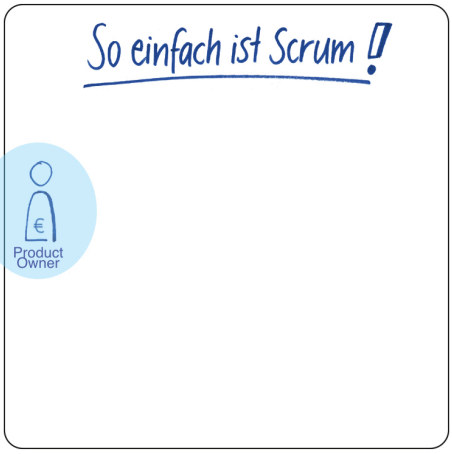
Wir beschreiben die drei dargestellten Perspektiven in den folgenden Abschnitten ausführlicher.

[2] Der Scrum Guide ist die offizielle Scrum-Definition: http://scrum-guides.org/

[3] Den dargestellten Scrum-Bierdeckel gibt es im it-agile-Shop: http://www.itagileshop.de

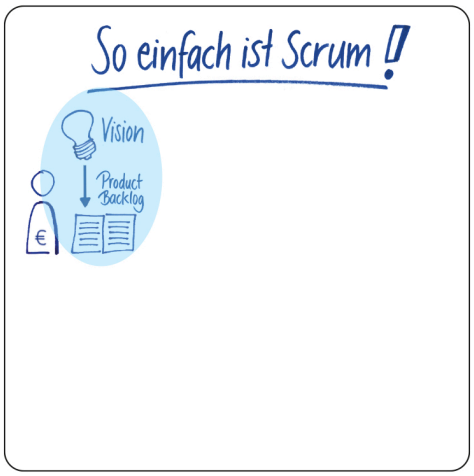
Scrum: Produktperspektive

Die Produktperspektive beginnt mit der *Product-Owner-Rolle*. Der Product Owner verantwortet den Produkterfolg, indem er den Produktnutzen durch die Priorisierung der Produkt-Features optimiert. Der Product Owner darf Entscheidungen über das Produkt treffen. Man kann sich den Product Owner auch als Unternehmer im Unternehmen vorstellen.



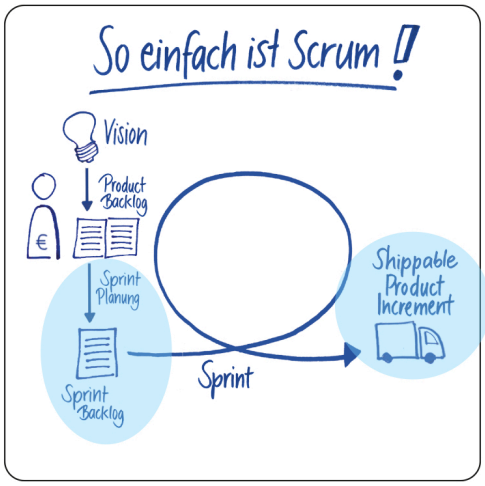
Für den Product Owner gilt das Highlander-Prinzip: „Es kann nur einen geben." Die Rolle kann in Scrum nicht von mehreren Personen geteilt wahrgenommen werden und schon gar nicht durch ein Komitee. Man möchte in Scrum, dass der Product Owner mit *einer* Stimme gegenüber dem Team und den Stakeholdern spricht und Entscheidungen schnell fällen kann.

Der Product Owner verfolgt eine Produktvision. Passend zur Produktvision pflegt der Product Owner ein *Product Backlog*, in dem die Produkteigenschaften beschrieben sind, die für den Produkterfolg notwendig erscheinen. Das Product Backlog wird durch den Product Owner *priorisiert* und durch das Entwicklungsteam *geschätzt*.

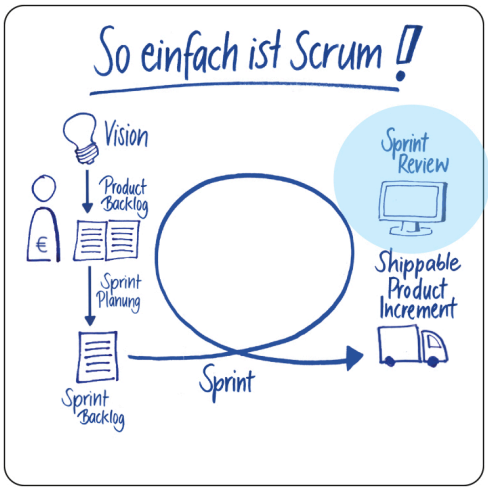


Scrum legt nicht fest, wie genau die Einträge des Product Backlogs gestaltet sind. Viele Teams machen gute Erfahrungen mit User Stories: Exemplarische Benutzungsszenarien aus Sicht eines Benutzers. User Stories haben eine andere Qualität als klassische Anforderungen. Bei User Stories liegt der Fokus darauf, ein gemeinsames Verständnis bei allen Beteiligten zu erzeugen und nicht darauf, dass die Beschreibung vollständig, widerspruchsfrei und korrekt ist.

Die Entwicklung erfolgt in Iterationen, die in Scrum *Sprints* heißen. Sprints haben eine immer gleiche Länge von max. 4 Wochen. Was im Sprint entwickelt wird, wird im *Sprint Planning* festgelegt. Hier werden hochpriorisierte Einträge aus dem Product Backlog ausgewählt, von denen das Entwicklungsteam meint, dass sie sie im Sprint umsetzen können. Das Ergebnis ist ein auslieferbares *Produktinkrement*. Ob das Produktinkrement tatsächlich an Kunden ausgeliefert wird, entscheidet der Product Owner. Die im Produktinkrement implementierten Features müssen aber auf jeden Fall produktionsreif sein (mind. entwickelt und qualitätsgesichert).



Das Entwicklungsteam demonstriert am Ende jedes Sprints das Produktinkrement im *Sprint Review* den Stakeholdern[4], damit diese Feedback zum Produkt geben können. Das Feedback wird vom Product Owner entgegengenommen und nach seinem Ermessen in das Product Backlog integriert.



Gute Fragen, um nützliches Feedback zu erhalten, sind:

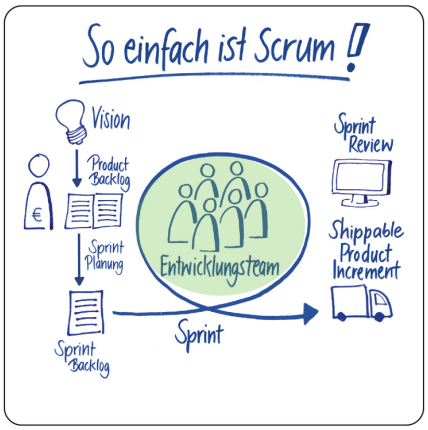
- „Was hindert uns daran, das vorliegende Produktinkrement produktiv zu benutzen?“
- „Wie kann das vorliegende Produktinkrement noch wertvoller gestaltet werden?“

[4] Stakeholder ist in Scrum jeder, der Interesse am Produkt oder Einfluss auf die Entwicklung hat: Kunden, Anwender, Sponsoren, Manager, Betriebsrat etc.

Scrum: Entwicklungsperspektive

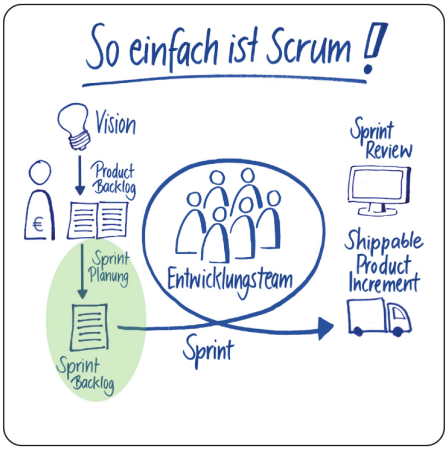
Das Entwicklungsteam entwickelt ausgehend vom Sprint Backlog ein auslieferbares Produktinkrement. Es besteht aus 3-9 Teammitgliedern, die alle Fähigkeiten vereinen, die notwendig sind, um das Sprint Backlog in das Produktinkrement zu überführen. Entwickler sind je nach Kontext Programmierer, UX-Experten, Designer, Handbuch-Autoren, Tester oder andere Experten.

Das Entwicklungsteam organisiert sich selbst. Es gibt weder eine formelle Hierarchie noch herausgehobene Rollen oder Positionen.



Das Entwicklungsteam bestimmt im *Sprint Planning*, wie viel Arbeit es in den Sprint aufnimmt. Es wendet das Pull-Prinzip an (es „zieht" Arbeit in den Sprint). Für die ausgewählten Einträge aus dem Product Backlog erstellt das Entwicklungsteam einen Plan für die Umsetzung im Sprint. Die ausgewählten Einträge aus dem Product Backlog zusammen mit dem Umsetzungsplan bilden das *Sprint Backlog*.

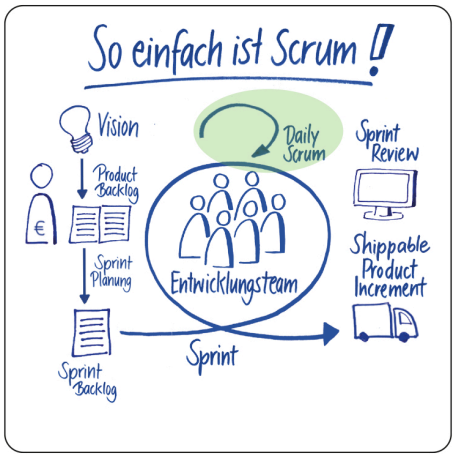
Das Entwicklungsteam macht so eine Vorhersage (Forecast) darüber, was es im Sprint schaffen kann. Diese Vorhersage soll die Qualität einer Wettervorhersage haben. In der Regel sollte das Entwicklungsteam das liefern, was es eingeplant hat. Es sollte aber niemand übermäßig überrascht sein, wenn das hin und wieder nicht klappt.



Während des Sprints treffen sich die Teammitglieder werktäglich zum Daily Scrum, um sich über den Arbeitsfortschritt und die nächsten Aufgaben im Sprint abzustimmen. Dazu kommen die Teammitglieder jeden Werktag zur gleichen Uhrzeit am gleichen Ort für maximal 15 Minuten zusammen und beantworten drei Fragen:

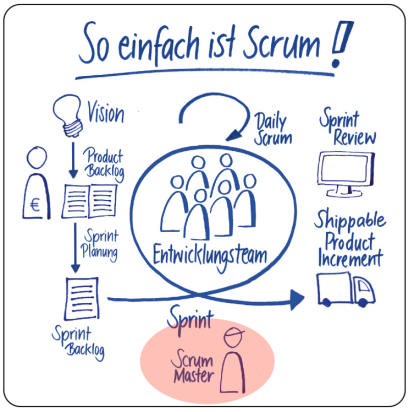
1. Was wurde seit dem letzten Daily Scrum erledigt, was uns dem Sprintziel nähergebracht hat?
2. Welche Hindernisse sehen wir auf dem Weg zum Sprintziel?
3. Was planen wir bis zum nächsten Daily Scrum zu erledigen, was uns dem Sprintziel näherbringt?

Der Product Owner ist ein optionaler Teilnehmer am Daily Scrum.

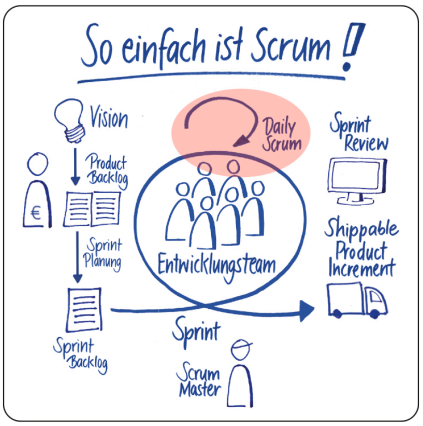


Scrum: Verbesserungsperspektive

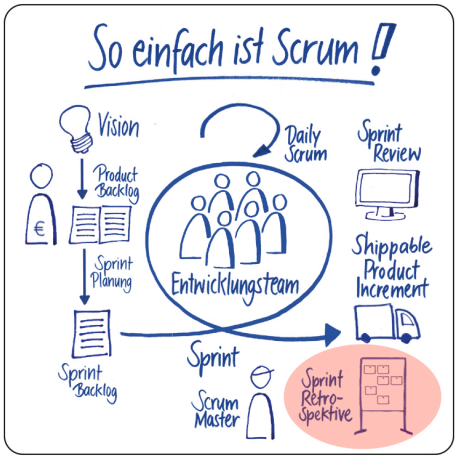
Der *Scrum Master* ist ein Coach für alle Beteiligten. Er sorgt dafür, dass Product Owner, das Entwicklungsteam und Stakeholder verstehen, wie Scrum funktioniert. Er hilft Ihnen, Scrum effektiv anzuwenden. Gegenüber dem Entwicklungsteam schafft er einen Rahmen, in dem sich das Team selbst organisieren kann und hält dem Team immer wieder den Spiegel vor. Der Scrum Master kümmert sich außerdem darum, dass Impediments (Hindernisse) identifiziert und beseitigt werden. Er moderiert in der Regel die Scrum-Meetings.



Die kontinuierliche Verbesserung des Entwicklungsprozesses ist bei Scrum in zwei Meetings verankert. Zum einen nehmen Verbesserungen ihren Ausgang im *Daily Scrum*, wenn Hindernisse identifiziert werden. Hindernisse sind in Scrum alles, was die Arbeit an aktuellen Aufgaben blockiert oder verlangsamt. Der Scrum Master kümmert sich um die Beseitigung der Hindernisse. Das bedeutet nur selten, dass er das Hindernis alleine aus der Welt schafft. Er wird dazu in der Regel mit weiteren Parteien im Unternehmen interagieren müssen (z.B. um finanzielle Mittel für schnellere Rechner zu beschaffen).



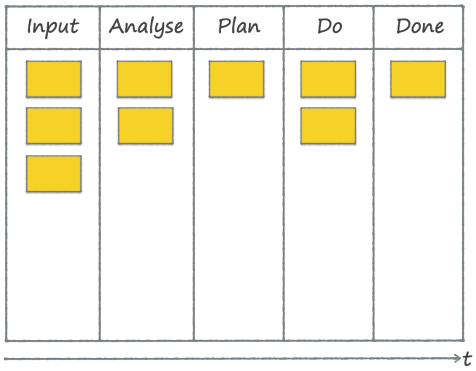
Zum anderen findet am Ende des Sprints die *Sprint Retrospektive* statt. Hier reflektiert das Entwicklungsteam zusammen mit dem Product Owner darüber, was im letzten Sprint gut und was weniger gut gelaufen ist. Auf dieser Basis definieren sie Verbesserungsmaßnahmen, die sie im nächsten Sprint umsetzen. Mindestens eine Maßnahme wird Teil des Sprint Backlogs des nächsten Sprints. Die Sprint Retrospektive wird durch den Scrum Master moderiert.



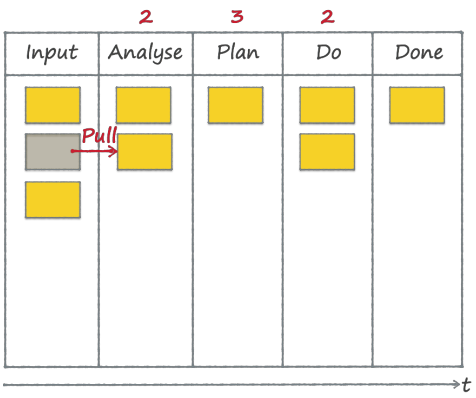
Kontinuierliche Verbesserung mit Kanban

Kanban[5] verfolgt einen anderen Ansatz als Scrum und startet mit der existierenden Situation. Kanban installiert einen kontinuierlichen Verbesserungsprozess, der ausgehend von den aktuellen Rollen und Arbeitsweisen real existierende Probleme identifiziert und beseitigt. Folgerichtig schreibt Kanban im Gegensatz zu Scrum keine Rollen, Meetings oder Artefakte vor. Damit eignet sich Kanban immer dann, wenn bereits eine funktionierende Arbeitsweise existiert, die schrittweise verbessert werden soll. Es ist damit breiter anwendbar als Scrum. Man kann auch Wartung, Support oder generell Service-Erbringung mit Kanban schrittweise in Richtung höherer Agilität bewegen.

Kanban beginnt stets damit, die **existierende Arbeit in dem existierenden Arbeitsfluss zu visualisieren**. Das geschieht über ein Kanban-Board. Die einzelnen Schritte des Workflows werden über die Spalten abgebildet. Die einzelnen Aufgaben „wandern" durch die Spalten.
In der Abbildung rechts findet sich ein einfaches Beispiel. In der Spalte „Input" stehen die anstehenden Aufgaben, die noch nicht begonnen wurden. Die Reihenfolge gibt die Priorisierung an. Danach durchlaufen die Aufgaben die Schritte „Analyse", „Planung" „(Plan") und „Umsetzung" („Do"). Vollständig erledigte Aufgaben wandern nach „erledigt" („Done"). Häufig werden physische Boards mit Haftnotizen verwendet, um Kanban-Boards abzubilden und mit ihnen zu arbeiten. Meist kommen die Beteiligten zu einer täglichen Kurz-Abstimmung zusammen, in der das Board aktualisiert wird.



Damit ist zunächst nur der Status Quo visualisiert. Damit Kanban als Verbesserungsprozess funktionieren kann, sind zwei Änderungen der bisherigen Arbeitsweise notwendig. Zum einen muss auf ein **Pull-System** umgestellt werden. Eine Aufgabe auf dem Board wird erst dann in die jeweils nächste Spalte *gezogen*, wenn dort Arbeitskapazität vorhanden ist. Es wird also nicht Arbeit zugewiesen oder in die jeweils nächste Spalte geschoben (Push).

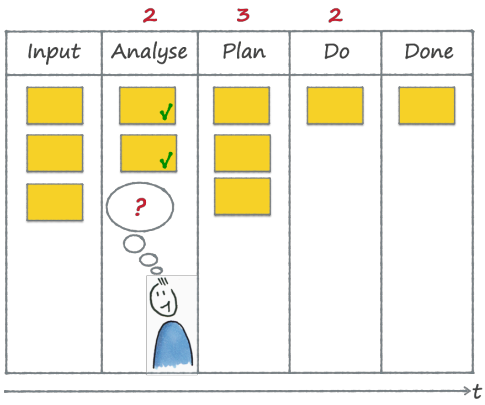


Außerdem fordert Kanban, dass die Menge der Arbeit in den einzelnen Schritten limitiert ist (**Work-in-Progress-Limit**; kurz: *WiP-Limit*). Diese WiP-Limits werden bei den entsprechenden Spalten annotiert. In dem Beispiel rechts dürfen also maximal zwei Aufgaben gleichzeitig analysiert, maximal drei geplant und maximal zwei umgesetzt werden.

Durch die WiP-Limits werden **Engpass-Situationen** sichtbar (engl. Bottlenecks): Wenn in einem Schritt das WiP-Limit erreicht ist, staut sich Arbeit immer weiter nach vorne. Der Stau entsteht immer direkt vor dem Engpass (während hinter dem Engpass häufig Unterauslastung anzutreffen ist).

[5] Die offizielle Kanban-Beschreibung findet sich in „Kanban Essentials Condensed": http://leankanban.com/guide/

In der Abbildung rechts sind drei Aufgaben im Schritt „Plan“ in Arbeit. Es können keine weiteren Aufgaben parallel geplant werden. Die beiden Aufgaben in „Analyse“ sind erledigt, können sich wegen des ausgeschöpften WiP-Limits bei „Plan“ aber nicht weiterbewegen. Gleichzeitig dürfen keine weiteren Aufgaben in „Analyse“ gezogen werden, weil auch dort das WiP-Limit erreicht ist. In einem Kontext, indem es spezialisierte Analysten gibt, die die Analysen durchführen, sind diese damit arbeitslos. Sie können keine weiteren Aufgaben analysieren. So erzeugen die WiP-Limits **Slack-Time**: Zeit, die nicht für die operative Arbeit verwendet werden kann. Diese Zeit soll dafür verwendet werden, das Gesamtsystem zu verbessern. Die Analysten sollten sich jetzt mit der Frage beschäftigen, wie der Prozess verbessert werden kann. Das kann z.B. dadurch passieren, dass weitere Planer integriert werden, die Planer von nicht-Planungsaufgaben befreit werden oder die Analysten das Planen lernen.



Wenn durch solche Maßnahmen der Engpass beseitigt wurde, wird an anderer Stelle der nächste Engpass sichtbar. Wird kein Engpass sichtbar, sollten die WiP-Limits reduziert werden, um weitere Engpässe deutlicher sichtbar zu machen.

Durch diese kontinuierliche Arbeit an den Engpässen wird der **Arbeitsfluss** (engl. ***Flow***) verbessert. Eine relevante Metrik hierfür ist die **Durchlaufzeit** (engl. ***Lead Time***) der Aufgaben. Nach Little’s Law sinkt die durchschnittliche Durchlaufzeit, wenn das Work-in-Progress gesenkt wird.

Kanban-Prinzipien und -Praktiken

Kanban definiert drei Change-Management- und drei Service-Delivery-Prinzipien[6].

Die drei Change-Management-Prinzipien sind:

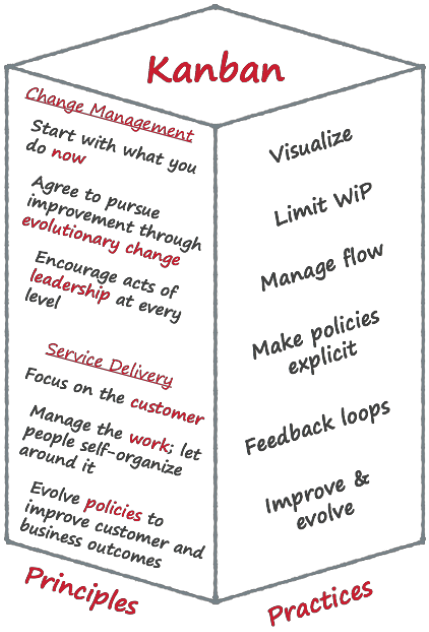
- **Start with what you do now.** (Nehme die existierenden Rollen, Artefakte und Arbeitsweisen als Startpunkt.)
- **Agree to pursue improvement through evolutionary change.** (Vereinbare, Verbesserungen durch evolutionäre Veränderungen zu anzustreben.)
- **Encourage acts of leadership at every level.** (Ermutige zu Leadership auf allen Ebenen vom einzelnen Mitarbeiter bis hin zum Top-Management.)

Die drei Service-Delivery-Prinzipien sind:

- **Focus on the customer.** (Den Kunden und seine Bedürfnisse und Erwartungen verstehen und darauf fokussieren.)
- **Manage the work; let people self-organize around it.** (Die Arbeit managen und nicht die Mitarbeiter. Rahmen für Selbstorganisation herstellen.)
- **Evolve policies to improve customer and business outcomes.** (Richtlinien und Vereinbarungen schrittweise verbessern, um Ergebnisse für Kunden und das eigene Unternehmen zu verbessern.)

Für die Umsetzung der Prinzipien schlägt Kanban sechs Praktiken vor:

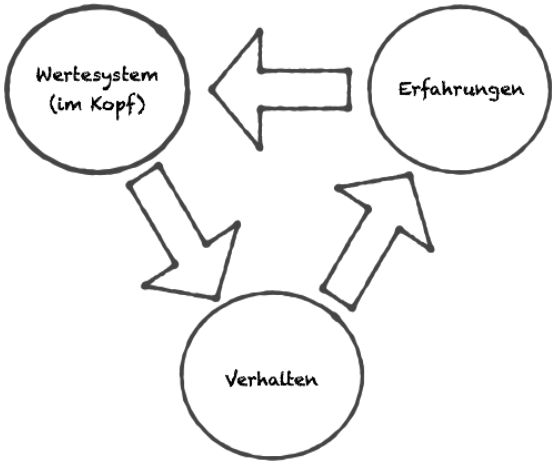
- Visualize
- Limit WiP
- Manage flow
- Make policies explicit
- Feedback loops
- Improve & evolve



Agil(er) werden

Agile Entwicklung erfordert veränderte Verhaltens- und Denkweisen bei allen Beteiligten. Nur die Mechanik eines Ansatzes wie Scrum zu installieren, reicht nicht aus.

Die notwendigen Verhaltensänderungen lassen sich nicht über Anweisungen herbeiführen, wie nebenstehende Abbildung zeigt. Jeder hat ein Wertesystem im Kopf. Ein Glaubenssatz könnte z.B. sein: „Vertrauen ist gut, Kontrolle ist besser." Dieses Wertesystem prägt das konkrete Verhalten, das wir an den Tag legen, z.B. „Hr.

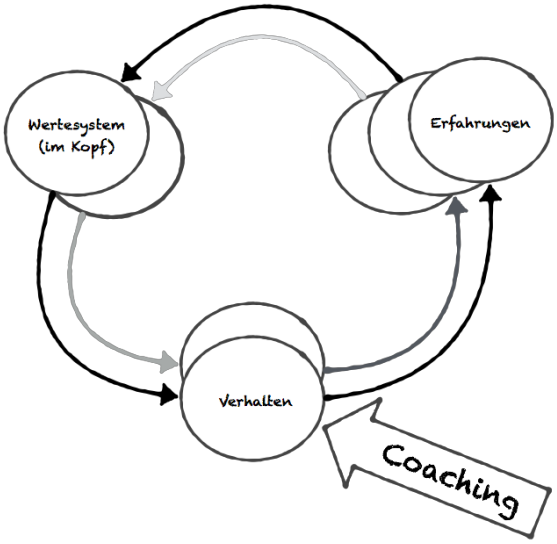


[6] Die Abbildung ist angelehnt an „Essential Kanban Condensed": http://leankanban.com/guide/

Müller, ich vertraue Ihnen diese Aufgabe an und möchte, dass sie mir morgen früh Bericht über den Fortschritt erstatten.“ Dieses Verhalten erzeugt im gegebenen Kontext bestimmte Reaktionen und Ergebnisse und prägt damit die Erfahrungen, die wir machen. So erfahren wir vielleicht am nächsten Tag, dass Hr. Müller mit der ihm anvertrauten Aufgabe noch nicht mal angefangen hat. Diese Erfahrungen wirken zurück auf unser Wertesystem („Gut, dass ich kontrolliert habe.“). In der Regel haben sich Zyklen entwickeln, in denen sich Werte und Erfahrungen gegenseitig verstärken („Nächstes Mal kontrolliere ich am besten halbtäglich.“).

Die neuen Verhaltensweisen müssen schrittweise erlernt werden. Ein verändertes Verhalten erzeugte andere Erfahrungen und schließlich ändert sich unser Wertesystem im Kopf.

Wer schon mal versucht hat, abzunehmen oder mit dem Rauchen aufzuhören, weiß, wie schwer es ist, angelernte Verhaltensweisen abzulegen. So geraten wir immer wieder in Situationen, in denen wir uns wider besseres Wissen unpassend verhalten. Und selbst wenn wir die gewünschte Verhaltensweise in einer Schulung eingeübt haben, fallen wir in Stress-Situationen häufig wieder zurück in alte Verhaltensmuster.



Coaching (durch einen Scrum Master oder einen externen Scrum-Coach) hilft dabei, Verhalten nachhaltig zu ändern. Dazu muss der Coach verstehen, was agile Entwicklung wirklich bedeutet und dazu muss er es bereits praktiziert haben – ansonsten hat der den Prozess der Verhaltensänderung selbst noch nicht durchlaufen.

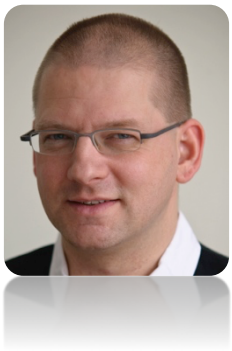
Um nachhaltig und erfolgreich agil(er) zu werden, muss also das Wissen vermittelt *und* Verhaltensweisen geändert werden. Eine Kombination aus Schulungen und Coaching ist unabdingbar.

Abschluss

Agilität ist in erster Linie eine Frage des Mindsets (Werte und Prinzipien). Konkrete methodische Ansätze wie Scrum und Kanban können dabei helfen, dieses Mindset zu erlernen und zu verankern. Sie entfalten ihre Wirksamkeit aber nur, wenn der Mindset-Wandel auch stattfindet.

Agile Entwicklung, Scrum und Kanban haben ihren Ursprung alle in der Software-Entwicklung. Agile Entwicklung wurde aber auch in vielen anderen Bereichen erfolgreich praktiziert: Entwicklung von physischen Produkten wie Autos, Entwicklung und Durchführung von Services (z.B. Übersetzungen) und Organisationsveränderungen (z.B. Unternehmensagilisierung). Wenn man stets die agilen Werte und Prinzipien vor Augen hat, ist der Transfer agiler Vorgehensweisen auf neue Anwendungsbereiche möglich.

Es ist sicher sinnvoll, sich über dieses PDF hinaus mit Agilität auf theoretischer Ebene zu beschäftigen. Aber irgendwann ist es notwendig, mit agilen Verfahren zu arbeiten, um wirklich zu verstehen, was diese im gegebenen Kontext bedeuten. Seien Sie nicht zu zögerlich, aber bleiben Sie offen für all die positiven und negativen Überraschungen, die sich bei der Anwendung zeigen werden.



Nehmen Sie gerne Kontakt mit uns auf:

Stefan Roock

Tel. 0172/429 76 17

stefan.roock@it-agile.de

http://www.it-agile.de