AGILES MANIFEST II: PROZESSE UND WERKZEUGE

Bei der Softwareentwicklung gehen wir ständig mit Prozessen und Werkzeugen um. Prozesse geben eine Idee davon, welche Personen in welchen Rollen zu welchen Zeitpunkten welche Tätigkeiten ausführen sollten. Werkzeuge unterstützen uns bei der Arbeit. In diesem Sinne sind Prozesse auch Werkzeuge. Im Umgang mit Werkzeugen müssen wir uns aber immer ein bisschen vorsehen: Wenn ich einen Hammer in der Hand habe, sieht jedes Problem wie ein Nagel aus. Schaue ich genauer hin, erkenne ich die Schraube vielleicht, aber muss ich jetzt für jedes Problem ein Werkzeug mit mir führen, das ganz genau passt?

der autor

Henning Wolf
(E-Mail: henning.wolf@it-agile.de)
ist Geschäftsführer der it-agile GmbH. Er verfügt über langjährige Erfahrung aus agilen Softwareprojekten (XP, Scrum, Kanban) als Entwickler, ScrumMaster und Coach. Er hilft Unternehmen, agile Methoden erfolgreich einzuführen.

Ohne Prozesse wären wir ziemlich aufgeschmissen. Wir könnten zwar irgendwie gemeinsam versuchen, täglich zu schauen, wie wir unserem Ziel näher kommen. Aber über das „irgendwie" ließe sich ständig neu streiten – oft sicher auch zu Recht.

Prozesse sind was Feines …

Prozesse geben uns Orientierung. Sie definieren einen Zuschnitt von Verantwortlichkeiten in Rollen, sodass wir für unsere Prozesse die richtigen Leute mit den richtigen Qualifikationen und Ermächtigungen auswählen können, aber auch, damit diese Rolleninhaber wissen, was von ihnen erwartet wird, und so arbeitsfähig werden.

Zusätzlich definieren Prozesse Abläufe, in welcher Reihenfolge welche Meetings oder Tätigkeiten stattzufinden haben. Sie legen fest, wer daran teilnimmt, wer verantwortlich ist und wer das Ergebnis wie prüft. Es entsehen dann sehr detaillierte Prozesse, deren Schicksal es in Organisationen meist ist, dass bestenfalls um sie herum gearbeitet wird, meist jedoch werden sie irgendwann einfach ignoriert. Wir sprechen dann vom Prozesshandbuch gerne als Schrankware. Häufige Gründe für diesen Effekt sind: Aus der Orientierung, die ein Prozess eigentlich geben soll, wird durch Hinzufügen von immer mehr meist gut gemeinten Anweisungen, Schritten und Rollen ein enges Korsett, das sich für die ausführenden Menschen anfühlt, als wären sie den Prozess exekutierende Roboter. Dieses Schicksal ereilt meist vor allem reichhaltige Prozesse oder Vorgehensmodelle, die auf der anderen Seite eine große Sicherheit versprechen wollen. Agile Prozesse setzen hier tendenziell eher auf Orientierung und eine leichte Unterspezifizierung, die der moderne Wissensarbeiter mit gesundem Menschenverstand füllen darf.

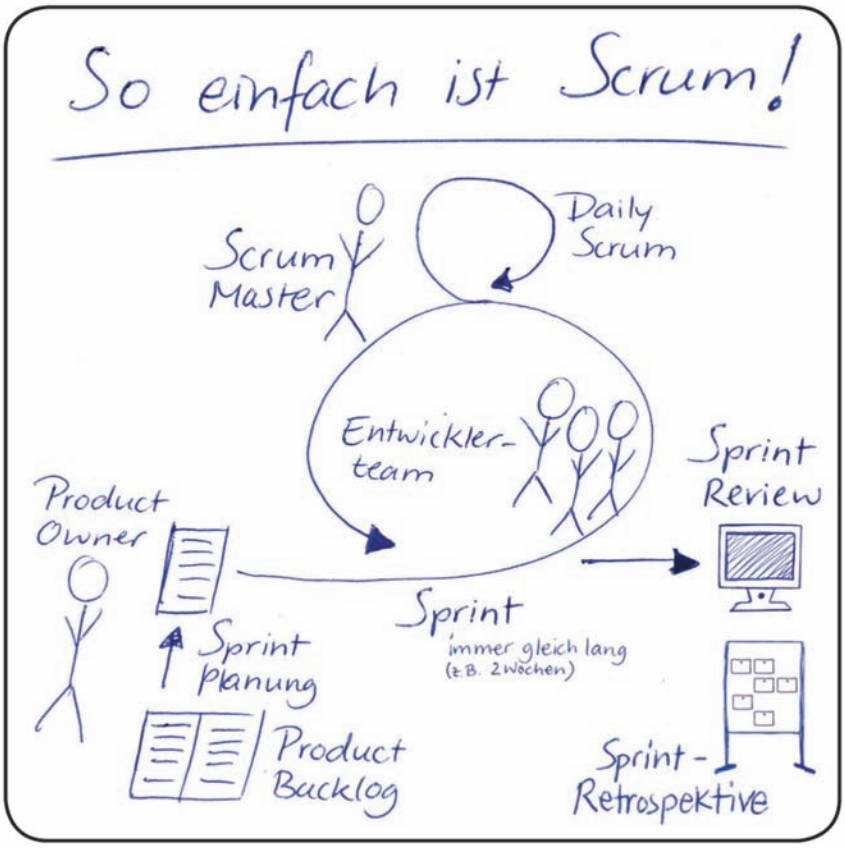

Abb. 1: So einfach kann ein Prozess sein. Dafür bleiben Detailfragen offen.

Der Begriff „Prozess" wird von den wenigsten agilen Methoden direkt verwendet – Scrum wird beispielsweise als „Management-Framework" bezeichnet (siehe Abbildung 1). Es enthält mit den drei Scrum-Rollen und dem Scrum-Flow allerdings ausreichend Hinweise für einen gelebten Scrum-Prozess: Von der Sprint-Planung über die Sprint-Durchführung mit Daily-Scrums bis hin zum Sprint-Review und der Sprint-Retrospektive sowie anschließendem Start des nächsten Sprints. In diesem Sinne habe ich agile Methoden schon oft als Grundlage für leichtgewichtige Prozesse gesehen, die den ausführenden Menschen Orientierung bieten, ohne sie zu bevormunden oder zu gängeln.

… wenn man sich dran hält

Letztlich nutzen aber die besten Prozesse nur dann etwas, wenn man sich an sie hält. Das gilt völlig unabhängig von der Frage, wie leichtgewichtig der Prozess ist. Zumindest gibt es Untersuchungen darüber, dass Teams mit einem klar definierten Prozess, den sie auch tatsächlich anwenden, eine deutlich höhere Chance haben, ihr Projekt erfolgreich durchzuführen. Das verwundert nicht – schließlich ist Softwareentwicklung Teamarbeit und Teammit-

glieder profitieren von der Orientierung, die Prozesse geben können. Neben der Frage nach dem richtigen Vorgehen gibt es schließlich ausreichend viele andere Herausforderungen.

Wir machen, was wir wollen!

Das bedeutet nicht, dass wir die Frage nach dem richtigen Vorgehen – zumindest im agilen Kontext – nicht doch noch stellen würden. Es bedeutet aber eben keinesfalls jeder würde machen, was er will. Stattdessen setzen die meisten agilen Methoden für bestimmte Bereiche, wie beispielsweise die Entwicklung, in kurzen Iterationen auf selbstorganisierte Teams, in denen zumeist jeder diszipliniert machen sollte, was man gemeinsam festgelegt hat. Natürlich gibt es Abweichungen in Ausnahmefällen, diese gehören aber als Thema in die nächste Prozessreflektion, die das Team am Ende einer Iteration in einer Retrospektive vornimmt. Solche Retrospektiven sind auch genau der richtige Ort, an dem das Team seinen Prozess anpassen kann. Für solche Prozessanpassungen gelten zwei Regeln:

- Erlaubt ist, was erfolgreich macht (dann interessiert es nicht, ob der Prozess „Srum" heißt).
- Bevor man eine Komponente (bzw. eine Rolle, ein Meeting oder ein Artefakt) ändert, anpasst oder neu einführt, sollte man sich bewusst sein, wie die entsprechende Funktion oder Verantwortlichkeit bisher aussah und wie diese zukünftig weiterhin sichergestellt wird oder ob man bewusst darauf verzichten kann.

Generell gilt, dass es zwar menschlich und verständlich ist, dass viele Teammitglieder Prozessen einfach nur folgen. Am erfolgreichsten ist man aber in Teams, deren Mitglieder auch verstanden haben, warum der Prozess so ist, wie er ist. So setzen die meisten agilen Methoden (durch kurze Iterationen) beispielsweise auf Vertrauen und Fehler und lassen es in gewissem Maße zu, aus diesen zu lernen. Die meisten reichhaltigen Methoden dagegen haben eher den Anspruch, bei der Fehlervermeidung zu helfen. Damit wird von den Grundwerten auch schnell klar, wo Organisationen und Prozesse inkompatibel werden können oder wo – neben der mechanischen Ausübung eines neuen Prozesses – auch auf anderen Ebenen ein Umdenken erforderlich wird.

Werkzeuge

Bei der Softwareentwicklung verwenden wir reichlich Werkzeuge: Editoren, Compiler, Build-Server, Konfigurationsmanagement-Systeme, GUI-Builder, Debugger usw. Diese Werkzeuge nehmen uns wesentliche Teile unserer Arbeit ab oder unterstützen und erleichtern uns die Arbeit. Sinnvollerweise muss dafür die Verwendung des Werkzeugs erst erlernt werden: Richtig effizient arbeitet man mit einer integrierten Entwicklungsumgebung erst dann, wenn die Tastenkürzel und Möglichkeiten, z. B. für automatisierte Refactorings, verinnerlicht wurden.

Neben diesen elektronischen gibt es aber auch physikalische Werkzeuge, beispielsweise ein Taskboard (siehe Abbildung 2), an dem Aufgaben auf Karten oder Haftnotizen stehen und das dem Team zur Koordination sowie zur schnellen und groben Übersicht über den Fortschritt der aktuellen Iteration dienen kann.

Es gibt aber auch viele Werkzeuge, die eher abstrakte Beschreibungen für Vorgehenselemente sind, die man zu seinem Prozess hinzufügen könnte, um bestimmte Probleme zu lösen: Wenn man etwa die Qualität des Codes erhöhen, neue Mitarbeiter einarbeiten oder generell Wissens-

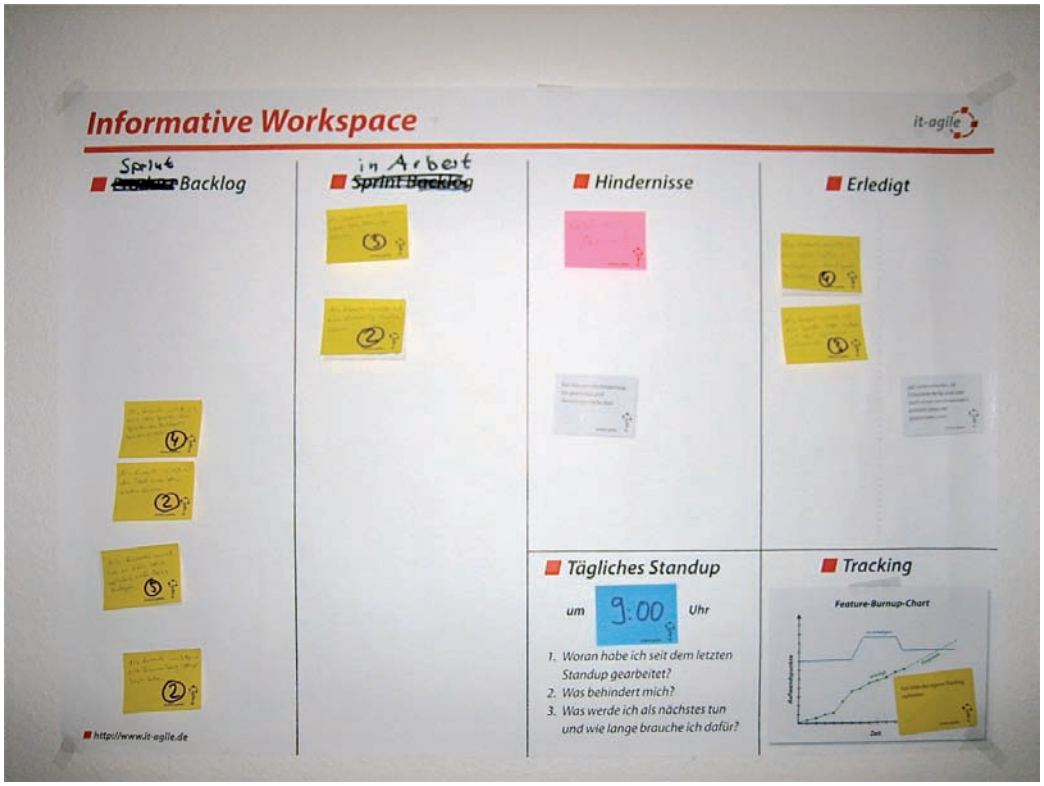

Abb. 2: Das typische Taskboard agiler Teams ist ein Low-Tech-Tool, das für viele Teams völlig ausreichend ist.

inseln vermeiden möchte, kann man ein Werkzeug wie das Programmieren in Paaren anwenden. Es ist nicht die einzige Möglichkeit, den beschriebenen Problemen zu begegnen, es gibt weitere, wie zum Beispiel Dokumentationsstandards oder regelmäßige Reviews.

Entscheidend ist nicht, dass jedes Werkzeug in jedem Projekt oder für jedes Team zum Einsatz kommen muss. Entscheidend ist, dass wir für die anstehenden Probleme Alternativen in unserem Werkzeugkoffer haben. Dabei stellen zuweilen die Werkzeuge selbst auch ein Problem dar, da sie oft in ihrer Anwendung noch nicht bekannt oder gut beherrscht sind, sodass sich die gewünschten Effekte nicht sofort einstellen, sondern Zeit brauchen. Diese Zeit muss man neuen Werkzeugen geben. Dabei gilt es zu beachten, dass man zuweilen dem Problem noch mehr auf den Grund gehen muss, um festzustellen, ob man gerade an der richtigen Stelle mit Optimierungen beschäftigt ist. Durch den reinen Austausch des Build-Werkzeugs von *Ant* auf *Maven* ohne Veränderungen an der Komponenten- oder Modulstruktur des Systems wurden vermutlich noch in keinem Projekt die Build-Probleme gelöst.

Prozesswerkzeuge: nur fein, wenn klein

Als Berater und Coach habe ich sehr oft mit Projekten zu tun, die sich die Einführung eines neuen Prozesses ohne ein diesen Prozess unterstützendes Werkzeug nicht vorstellen können. Das finde ich nachvollziehbar, sehe ich mich als Coach doch genau als dieses Werkzeug. Aber weit gefehlt, viele suchen nach einem elektronischen Tool, das ihnen „den Prozess macht" und dabei noch ganz nebenbei alle bisherigen Reports ausspuckt, damit man nach Außen nichts ändern muss. Nun machen wir ja meist deshalb etwas Neues, weil wir gerade etwas ändern wollen – insofern ist der Wunsch nach äußerer Stabilität nur bedingt verständlich. Ich kann auch nur halb erklären, warum ich mich mit dem Wunsch nach dem agilen Prozesswerkzeug so schwer tue. Ich glaube, meine Hauptbefürchtung ist, dass man versucht, statt einem anderen jetzt eben einem agilen Prozess zu folgen. Aber folgen ist mir nicht genug, denn ich bin fest davon überzeugt, dass man Agilität leben muss. Dazu muss man sie verstanden haben und nicht Dinge tun, weil das Tool diese verlangt. Deswegen wünsche ich mir zuerst das Verständnis und erst danach die Auswahl eines geeigneten Werkzeugs bzw. einer Werkzeugkombination.

Zudem gibt es einen weiteren Effekt, den ein Werkzeugeinsatz bewirken kann, wenn man Werkzeuge vor dem Verständnis der dahinter liegenden Aufgabe nutzt: Häufig beobachte ich bei neu agil arbeitenden Teams, dass die User-Storys und/oder Tasks in einem Issue-Tracker verwaltet werden, beispielsweise in „Mantis" oder „Jira". Weil man mit diesen Werkzeugen neben dem Schätzaufwand auch den tatsächlichen Aufwand je Task oder User-Story erfassen kann, glauben viele, dass es auch vernünftig und sinnvoll wäre, dies zu tun. Dabei gehört es zum Verständnis des selbstorganisierten agilen Teams, dass von außen betrachtet nur die Geschwindigkeit des Teams in der Summe der erledigten Aufgaben einer Iteration (*Velocity*) interessiert. Ich will doch im Mix der Aufgaben, Verschätzungen und Teammitglieder gerade zukünftig immer konstant gleich schlecht schätzen. Es kommt also gar nicht darauf an, dass ich in der Vorhersage des Aufwands der einzelnen Aufgabe besser werde. Gerade wenn Projektleiter aber die erfassten Soll- und Ist-Zeiten überwachen, wird diese Idee gänzlich ad absurdum geführt. Wenn das Team diese Messung machen möchte, weil es nicht anders weiß, woher eine gegebenenfalls stark schwankende Geschwindigkeit zwischen den Iterationen herkommt, finde ich es für eine Zeit lang in Ordnung, diese Messung vorzunehmen. Sie hat dann aber auch niemanden außerhalb des Teams zu interessieren.

Das wichtigste nicht-elektronische Prozesswerkzeug ist übrigens die Retrospektive: Sie erlaubt uns eine Reflektion und eine darauf basierende Anpassung unseres Prozesses und unserer Werkzeuge sowie unserer Werkzeugverwendung.

Fazit

Prozesse sind Werkzeuge, Werkzeuge sind wichtig. Wir müssen sie aber erst beherrschen lernen und ordentlich anwenden, damit sie uns nützen. Gerade bei Prozesswerkzeugen, die teilweise den Anspruch haben, den kompletten Entwicklungsprozess integriert zu erlauben, ist das kritisch zu betrachten. Denn es fällt oft schwer, zwischen den eigentlichen Bedürfnissen und den Anfordernissen oder Möglichkeiten der konkreten Werkzeuge zu unterscheiden. ■