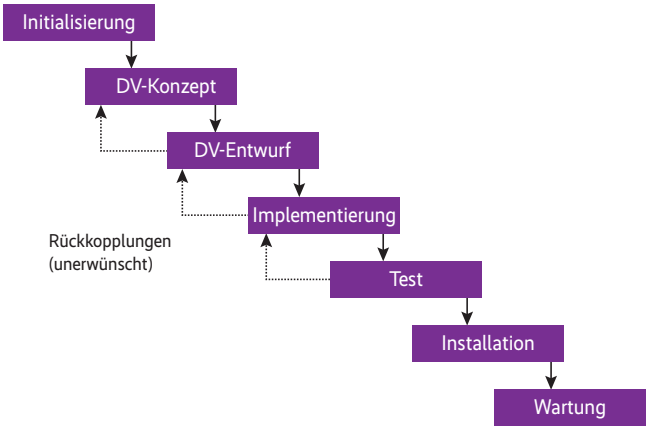
Wie agile Softwareentwicklung ein Berufsbild verändert

Ende des Einzelkämpfers

Andreas Boes, Matthias Grund, Charlotte Sanwald

Der Wandel, den agile Vorgehensweisen in der Softwareentwicklung eingeläutet haben, reicht weit über neue Arbeitstechniken hinaus. Programmieren ist in der agilen Welt nicht länger die exklusive Leistung einer einzelnen Person, sondern eines Teams.

Die Frage, ob Unternehmen nach agilen Methoden entwickeln oder das bürokratische Wasserfallmodell (Abbildung 1) bevorzugen, gilt inzwischen als anachronistisch. Zu heftig war und ist die Kritik an letzterer Vorgehensweise, die es kaum erlaubt, das laufende Projekt an neue Anforderungen anzupassen. Was den Wasserfall zu Fall brachte, ist sein immanenter Widerspruch: Ein Entwickler ist deswegen Entwickler, weil er Software guter Qualität erzeugen möchte. Der Kern der Softwarequalität liegt jedoch direkt im Code – egal wie viele Prozesse davor, dahinter oder darüber geschaltet werden. Und damit in der Programmierung, nicht in Vertragswerken und Dokumenten. Anders gesagt: Es ist nicht möglich, mangelhafte Code-Qualität weg zu designen oder weg zu analysieren. Schlechter Code ist schlechter Code und damit der Sprengsatz, der das ganze Projekt früher oder später sabotieren kann. Die tayloristischen Konzepte der klassischen Softwareentwicklung erkennen indessen den Wert des Programmierens nicht an. Stattdessen wurde immer stärker versucht, Verbesserungen zu erzielen durch noch mehr Segmentierung, noch mehr nachgelagerte Qualitätssicherung und mittels Reorganisationen. Kurz: Durch immer weiter reichendes Loslösen des Programmierens von den anderen Schritten des Prozesses. Diese einseitige Betonung erinnerte an Taylors Idee, Planung und Ausführung komplett zu trennen und den intelligenten Teil der Arbeit in die Arbeitsvorbereitung zu verlagern (Abbildung 3).



Das Wasserfallmodell gilt als veraltet, weil Rückkopplungen mit vorherigen Arbeitsschritten nicht möglich sind (Abb. 1).

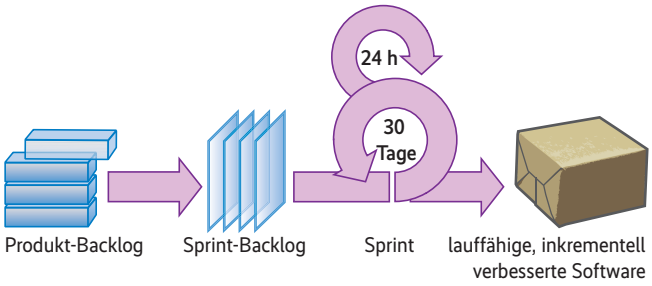
eXtreme Programming und Scrum (Abbildung 2) formierten bereits Mitte der 1990er-Jahre eine Gegenbewegung. Damit rückte das Programmieren wieder in den Mittelpunkt. Schnell schlugen die neuen Ansätze erste zarte Wurzeln in der Entwicklerwelt. Das Management hingegen zeigte sich skeptisch. Breite Akzeptanz auch in den Führungsetagen kam dann im Verbund mit den neuen Konzepten des Lean Management und der Lean Production. Das neue Produktionsmodell für IT-Arbeit betrachtet die Produktion als systemisch integrierte Wertschöpfungskette mit Fokus auf den Kundennutzen. Es ist dieser gemeinsame Sinnbezug auf den Kunden sowie das kollektive Wissen, über das die einzelnen Teilprozesse zusammenfinden.

Agiles Steigern der Produktivität

Als 17 renommierte Softwareexperten 2001 das „Manifesto for Agile Software Development“ (www.agilemanifesto.org) unterzeichneten, läuteten sie nicht quasi automatisch die schöne neue Entwicklerwelt ein. Denn auch die agilen Methoden ändern nichts daran, dass in jedem Projekt unterschiedliche Personen und Gruppen aufeinandertreffen, die zwar insgesamt gleichartige Interessen haben, jedoch mit unterschiedlicher Gewichtung. Das beginnt

beim Management, bei Scrum oft in Gestalt des Product Owner, für den Rentabilität eine dominante Rolle spielen muss. Für das Entwicklerteam hingegen hat die eigene Qualifizierung hohe Priorität. Vereinen könnten sich beide Anforderungen in einem übergreifenden Ziel: dem Steigern der Produktivität. Doch auch, wenn dieses Ziel klar und ein gemeinsames Anliegen ist – die favorisierten Wege dorthin können so unterschiedlich sein wie die Landschaftsbilder im Flachland und den Alpen. Und nicht jeder Weg entspricht auch im Detail dem, was sich beide Seiten wünschen.

Zunächst klingen die agilen Ideen uneingeschränkt gut: Mehr Kompetenz für das Team, mehr Entscheidungsgewalt und Verantwortung, und wenn Planung und Ausführung nicht länger getrennt werden, dann kommt endlich zusammen, was zusammengehört. Indes: Der schöne Satz, alles sei relativ, gilt auch hier. Allein die Dialektik der Transparenz ist es, die viele Entwickler skeptisch macht. Agile Methoden fordern Transparenz bis zur letzten Konsequenz. Das ist die Bedingung für die Kooperation, für die Möglichkeit, als Team kreativ zu sein und gemeinsam Lösungen zu suchen. So weit, so gut. Nur ist diese Transparenz nicht zum Schnäppchenpreis zu haben. Sie kostet, und der Preis, der auf ihr klebt, ist die „Wissensautonomie“ des Einzelnen. „Wissenssilos“ aufzubrechen, ist sicher im Interesse des Unternehmens, aber ist es auch im Sinne des Experten, der damit verliert, was ihn einzigartig macht? Spätestens bei der „Collective Code Ownership“ ist ein weitreichendes Bekenntnis der Einzelnen zur Gruppenarbeit gefordert. Gerade erfahrene Entwickler tendieren oft dazu, sich stark mit „ihrer“ Software zu identifizieren, sie praktisch als „ihr Baby“ zu betrachten. Da kann es kaum leichtfallen, zu tolerieren, dass andere in diese fast schon intime Sphäre eindringen und beispielsweise den eigenen Code verändern. Von der quasi privaten Sphäre bleibt bei der agilen Softwareentwicklung ohnehin nicht viel. Auch institutionalisierte Meetings wie der Daily Scrum, das Sprint Planning und der Review stellen das Programmieren ins helle Licht der Öffentlichkeit. Nicht zu vergessen, das Pair Programming, das gemeinsame Programmieren, als endgültiger Abschied vom „Alles meins“-Prinzip. Daraus resultieren neuartige Anforderungen an Entwickler: Sie müssen in der Lage und willens sein, ihr Wissen zu teilen. Das bedingt auch, sich Diskussionen zu stellen und andere zu überzeugen. Wo selbst der geniale Eremit aus seiner Einsiedlerhöhle gelockt und mitten in ein Team gestellt wurde, da zählt die Bereitschaft zur Offenheit massiv. Mit ihr wird die Fähigkeit elementar, das eigene Anliegen sprachlich darzustellen, abzugrenzen und gegebenenfalls zu verteidigen. Für viele Entwickler dürfte das in diesem Ausmaß eine neue Situation darstellen.



Scrum erlaubt und fordert Rückkopplungen: Durch ständige Verbesserungen will man bessere Software erreichen (Abb. 2).

Solche Offenheit bedingt grundlegendes Vertrauen. Vertrauen in die Tatsache, dass Offenheit nicht in erster Linie verwundbar macht. Anders ausgedrückt: Offenheit darf nicht die Basis von

Quelle: Wikipedia

Der US-amerikanische Ingenieur Frederick Winslow Taylor (1856 – 1915) wollte Arbeit in immer kleinere standardisierte Teilaufgaben zerlegen und so die Produktivität steigern (Abb. 3).

Kontrolle (der Taylor'schen Stoppuhr) und Gruppendruck sein, sondern von Kooperation und kollektiven Lernschleifen. Womit wir beim nächsten potenziellen Fallstrick wären: Kurze Lieferintervalle und eigene Aufwandsschätzungen des Teams sollen zu realistischen Zeitangaben und einem vernünftigen Arbeitstempo führen. Vor allem gute Teams laufen jedoch schnell Gefahr, sich selbst jedes Zeitpuffers zu berauben, einfach nur den Druck zu erhöhen und das Arbeitstempo in ungesunde Höhen zu schrauben. Nur ist das keinesfalls der Zweck des Ganzen. Gerade weil es vom Burndown-Chart zum Burnout ein kurzer Weg sein kann, sind die Unternehmen gefragt, gestaltend einzugreifen und sich beispielsweise dazu zu bekennen, dass unproduktive Zeiten keinesfalls verschwendet sein müssen, können sie doch wichtigen Zielen dienen wie der Regeneration und dem Teamzusammenhalt.

Ob und wie die Unternehmen gestaltend eingreifen, hängt davon ab, in welche Richtung sich die Softwareentwicklung bewegt. Gemäß der erwähnten Dialektik vieler agiler Prinzipien sind zwei grundsätzliche Szenarien denkbar.

In der Entwicklerhölle

Das erste Szenario steht im Zeichen einer Art von Industrialisierung, in der Software zu einem „Fließbandprodukt" wird und neue Formen der Kontrolle die Entwickler unter einen Druck setzen, der in austauschbarer, quasi seelenloser IT-Arbeit kumuliert. Diese Vorstellung aus der Entwicklerhölle hat nur nichts mit dem ursprünglichen agilen Gedanken zu tun. Denn deren Absichten zielen auf ein ganz anderes Szenario: Einen Produktivkraftschub sowie eine neue Qualität des Nutzens geistiger Produktivkraft.

Ungeachtet der Tatsache, dass das Team bei den agilen Methoden so sehr im Mittelpunkt steht, liegt den agilen Ideen eine zutiefst positive Grundhaltung gegenüber dem Individuum zugrunde: Der Einzelne, so die Überzeugung, ist gar nicht mehr zu bremsen im eigenen Drang nach der ständigen Verbesserung des eigenen Tuns. Deshalb verankert beispielsweise das Manifest den Grundsatz, dass „Individuen und Interaktionen mehr als Prozesse und Werkzeuge" zu schätzen seien. Denn jedes Team besteht aus qualifizierten Individuen, die durch den ständigen Austausch ihre eigenen Kenntnisse ebenso ständig erweitern. Das ist die Gegenleistung, die die einzelnen Teammitglieder für den Verzicht auf ihr Wissensmonopol bekommen.

Mehr noch: Ein Team, das sich über einige Sprints hinweg zu einem echten Team formiert hat, nimmt den Erfolgsdruck von einzelnen Schultern und kann bei weniger Stress mehr erreichen. Wie das? „Team-Empowerment" ist nicht nur der Leitgedanke im Hinblick auf den Teamgeist und die Art der Zusammenarbeit. Es bedeutet auch, die Entwicklerkapazitäten auf ein neues Niveau zu heben, in dem die entsprechenden handwerklichen Instrumente etabliert werden. Gemeint sind Arbeitstechniken wie die testgetriebene Entwicklung, Refactoring und die kontinuierliche Integration. Denn diese Arbeitstechniken sollen dem Team dazu verhelfen, das zu erreichen, was erfolgreiche Sprints ausmacht: Erstens die Fähigkeit, realistisch einzuschätzen, wie viel in welcher Qualität in einem Sprint produziert werden kann und zweitens das Vermögen, den eigenen Plan umzusetzen. Beides sind Fertigkeiten, die in einem Team selten „einfach so" vorhanden sein dürften. Ein Scrum-Sprint ohne entsprechende Engineering-Techniken und Erfahrung wäre jedoch ein mehr oder weniger aus der Luft gegriffener Rahmen.

Mit voller Konzentration

Auch das Scrum-Artefakt Product Backlog, in dem die Anforderungen an das Produkt genau definiert und vor allem festgeschrieben werden, nützt dem Entwicklerteam deutlich. Erstens, weil es vom Product Owner verlangt, ganz klar zu spezifizieren, was er oder sie wirklich will. Zweitens, weil Anforderungen, die nicht darin stehen, im fraglichen Sprint auch nicht erfüllt werden müssen. Analog verhält es sich mit dem Sprint-Backlog: Es macht präzise Vorgaben, auch zeitlich. Doch das bedeutet, dass sich die Entwickler auch voll und ganz auf diese Vorgaben konzentrieren können. Das unbeliebte „Ratet mal, was wir heute von euch wollen"-Spiel zwischen der anfordernden und der ausführenden Seite ist damit ausgespielt.

Nicht, dass es jemals eine Seite gegeben hätte, die es überhaupt spielen wollte. Aber wenn der Abgabe- oder Abschlussdruck zu groß wird, können sich durchaus Situationen ergeben, in denen die Lücke zwischen Wunsch und Wirklichkeit durch puren Druck verkleinert werden soll. Doch weil das so selten funktioniert, formulieren die agilen Methoden von vornherein andere Rahmenbedingungen.

In ihnen partizipieren die Entwickler an allen Teilen des Gesamtprozesses: An der Planung und der Schätzung, der Ausführung, dem Testen und der Auslieferung. Dieser Gedanke der grundlegenden Partizipation ist vermutlich der Schlüssel dafür, ob es gelingt, das gesamte ökonomische Potenzial der agilen Methoden auszuschöpfen. Sie erwarten von den Entwicklern das Ende des Einzelkämpfertums, exzellente kommunikative Fähigkeiten sowie die Bereitschaft zu konsequenter Teamarbeit. Dafür bietet sie außerordentliche Möglichkeiten, sich weiterzuentwickeln und – als Team – selbstbestimmt zu arbeiten. Insofern sollte sich das Szenario durchsetzen, das einen Produktivkraftschub fördert. Den Rahmen dafür geben die agilen Methoden vor.

Damit Entwickler das tun können, was sie wollen: Richtig gute Software entwickeln. (jd)

PD Dr. Andreas Boes
ist Privatdozent an der Technischen Universität Darmstadt und Vorstand des ISF München.

Matthias Grund
verantwortet im Vorstand der andrena objects ag die Bereiche Beratung und Entwicklung.

Dr. Charlotte Sanwald
ist bei der andrena objects ag zuständig für das Thema Kommunikation.