Agile Verfahren in der industriellen Software-Entwicklung

Sebastian Graf, PROMATIS software GmbH

Immer mehr Unternehmen entscheiden sich für die strategische Einführung agiler Entwicklungsverfahren und glauben, dadurch den Heiligen Gral unter den Vorgehensmodellen gefunden zu haben. Bewährte Verfahren, mit denen man über Jahrzehnte sehr gute Erfahrungen gemacht hat, werden dabei bedenkenlos über Bord geworfen. Oft wird dabei allerdings vergessen, dass agile Verfahren eine Ergänzung traditioneller Vorgehensmodelle darstellen und keinesfalls als deren Ablösung verstanden werden sollten.

Viele Unternehmen geben aktuell an, ihre zentralen Entwicklungsprozesse alle auf ein agiles Verfahren umzustellen oder eine solche Umstellung zu planen. Handelt es sich in diesen Fällen tatsächlich um eine radikale Umstellung auf ein agiles Verfahren, dann ist höchste Gefahr im Verzug, da in diesen Fällen Agilität in der Regel unreflektiert eingeführt wird, was in der Folge zu größten Problemen bis hin zu gescheiterten Projekten führen kann.

Selbstverständlich findet man zum Thema „Agilität" in der Presse und im Internet nur Positives. Fast schein es, als wäre Agilität ein Garant für erfolgreiche Projekte, zufriedene Kunden und hoch motivierte Entwicklerteams. Unglücklicherweise sieht die Realität in den meisten Fällen anders aus. Nach den ersten Erfahrungen mit der agilen Software-Entwicklung stellt sich nicht selten Ernüchterung ein und als kritischer Geist fragt man sich mitunter:

- Warum werden in agilen Projekten mehr oder weniger große Teile des Codes drei bis vier Mal entwickelt, bis die Anwender einmal damit zufrieden sind?
- Warum ändert sich das Datenmodell in einem agilen Projekt jeden Tag getreu dem Motto: „Sind uns Änderungen nicht gelungen, dann ändern wir auch Änderungen …"?
- Warum deckt eine mit agilen Methoden entwickelte Anwendung oftmals die fachlichen Anforderungen genauso gut oder schlecht ab wie eine Anwendung, die mit herkömmlichen Methoden entwickelt wurde?
- Warum haben agile Verfahren mit genau den gleichen Kommunikationsproblemen zu kämpfen wie herkömmliche Verfahren?

Aktuelle Studien

Mittlerweile sind agile Verfahren weitverbreitet, und viele Unternehmen investieren größere Summen in die Umstellung ihrer Entwicklungsprozesse auf agile Verfahren. So ist es auch kaum verwunderlich, dass alle Studien zu dem Schluss kommen, agile Methoden seien den klassischen Verfahren in puncto Qualität der Lösung und Effizienz des Entwicklungsprozesses haushoch überlegen. Betrachtet man die Vielzahl dieser Untersuchungen, so stellt man schnell fest, dass es sich dabei in vielen Fällen bestenfalls um pseudowissenschaftliche Abhandlungen handelt, bei denen das Ergebnis der Studie offenbar im Vorfeld bereits ausgemacht war und die einer ernsthaften wissenschaftlichen Überprüfung nicht standhalten.

Die wenigen Studien, die wirklich einem wissenschaftlichen Anspruch genügen, sprechen eine deutlich andere Sprache: So kommen beispielsweise Tore Dyba und Torgeir Dingsoyr in ihrer Abhandlung „Empirical studies of agile software development: A systematic review, Inform. Softw. Technol. (2008), doi:10.1016/j.infsof.2008.01.006", die diverse Studien zum Thema bewertet und vergleicht, zu der klaren Erkenntnis, es gebe keinerlei belegbare Hinweise dafür, dass sich mit agilen Verfahren schneller und effizienter entwickeln lässt und die dabei implementierten Lösungen qualitativ auch noch besser sind.

Eine der ausgewerteten Studien kam im direkten Vergleich zwischen V-Modell und Scrum sogar zu dem Ergebnis, dass das Scrum-Projekt im gleichen Zeitraum dreieinhalb Mal so viele Lines of Code hervorgebracht hatte wie das Team, das nach V-Modell vorgegangen war. In der Studie wurde jedoch leider vergessen zu erwähnen, dass beide Teams nach Abschluss des Projekts die absolut identische Funktionalität bereitgestellt hatten. Ein weiterer wesentlicher Kritikpunkt, der in vielen Studien einfach verschwiegen wird, besteht darin, dass agile Projekte offenbar dazu tendieren, den Blick für Design- und Architekturfragen zu verlieren. Ein Grund mehr, mit dem Einsatz agiler Verfahren in Schlüsselprojekten sehr vorsichtig umzugehen.

Erfahrungen aus der Praxis

Eines der prominentesten Argumente der Verfechter agiler Verfahren ist der Hinweis, dass es sich dabei nicht um einen komplexen wissenschaftlichen Ansatz handelt, der in der Realität nicht beherrschbar ist, sondern dass es ein praxiserprobtes Konzept sei, das von Praktikern für Praktiker ersonnen wurde.

Im Nachgang eines agilen Projekts, bei dem die Scrum-Methode zum Einsatz kam, wurde von einer Task-Force im Rahmen einer Nachbetrachtung, neuerdings auch „Lessons Learned" genannt, bei den Datenbank-Admi-

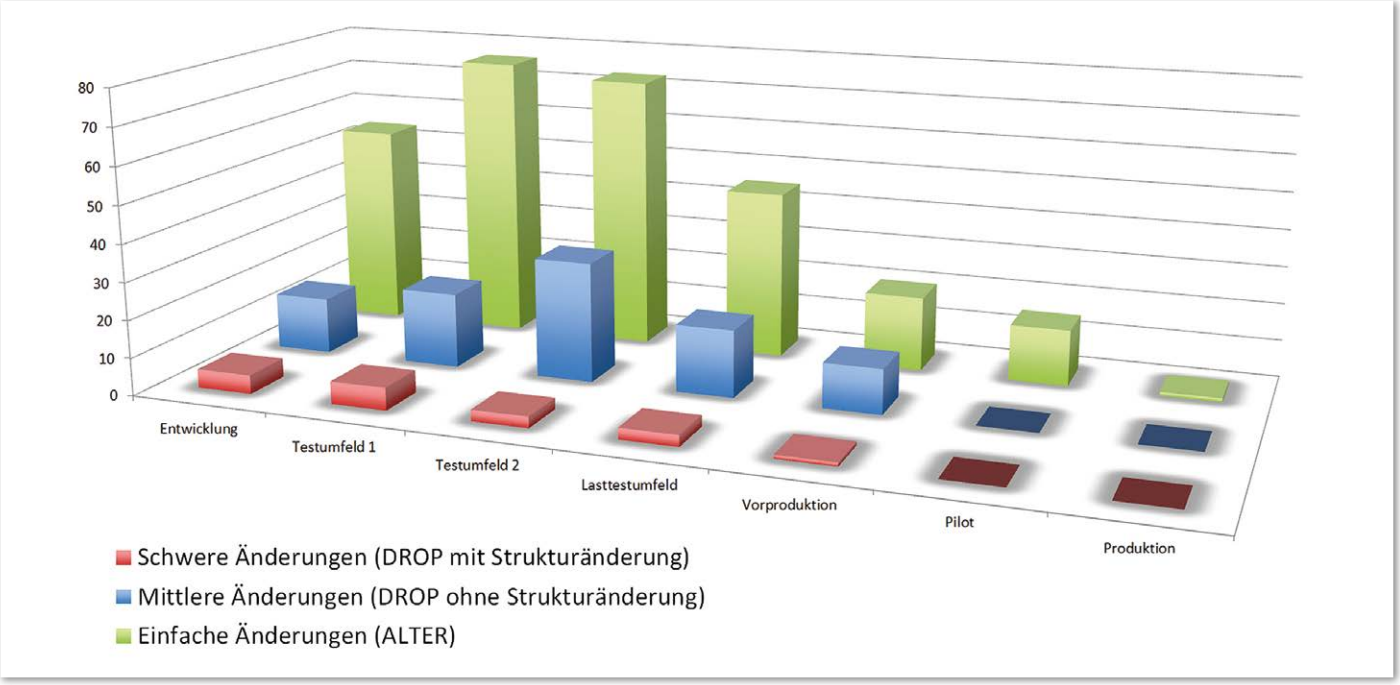

Abbildung 1: Wöchentliche Änderungsrate nach abgeschlossener System-Analyse

nistratoren eine Auswertung bezüglich der Änderungshäufigkeit mit Bezug auf das Datenmodell in Auftrag gegeben. Diese Untersuchung hat sehr interessante Erkenntnisse zutage gefördert *(siehe Abbildung 1)*.

Bei der Bewertung der Zahlen ist zu berücksichtigen, dass in fraglichem Projekt in einer vorgelagerten Analyse-Phase das komplette der Anwendung zugrunde liegende Datenmodell erstellt werden sollte. Lediglich die Anwendungs-Komponenten sollten nach agilen Methoden entstehen. Gegen den Protest der Datenbank-Experten wurden in der Analyse-Phase aber Detail-Fragen häufig auf die Entwicklungsphase verschoben, mit dem Hinweis, dass man aktuell die Details gar nicht kenne und sich das während der einzelnen Sprints ergeben müsse. Ein Irrtum, den man teuer bezahlen sollte.

Wie die Abbildung zeigt, wurden Änderungen am Datenmodell sogar noch bis in die Pilotierungsphase mit entsprechender Kundenwirkung durchgeführt. Teilweise haben sich sogar einzelne strukturelle Änderungen am Datenmodell bis in die Produktionsphase gezogen – für einen Datenbank-Experten eine schiere Horrorvorstellung.

Angesichts dieser chaotischen Vorgehensweise liegt natürlich die Versuchung nahe, dafür die agile Methode, die im Projekt gewählt wurde, verantwortlich zu machen. Dieses wäre allerdings etwas vorschnell und, um es vorwegzunehmen, agilen Verfahren gegenüber nicht wirklich gerecht. Um agile Verfahren und deren Stärken sowie Schwächen besser einschätzen zu können, sollen hier nochmals einige Prinzipien agiler Software-Entwicklung dargelegt werden.

Grundprinzipien der agilen Software-Entwicklung

Das Thema „Agilität in der Software-Entwicklung" ist gar nicht mehr so jung. Das Agile Manifest, auf das alle agilen Methoden zurückgehen, wurde bereits im Jahre 2001 veröffentlicht. Quasi auf dem Höhepunkt der Software-Krise haben sich einige namhafte Software-Entwickler mit dem Agilen Manifest auf zwölf Prinzipien der Software-Entwicklung verständigt. Diese sind einleuchtend und enthalten einfache und klare Botschaften:

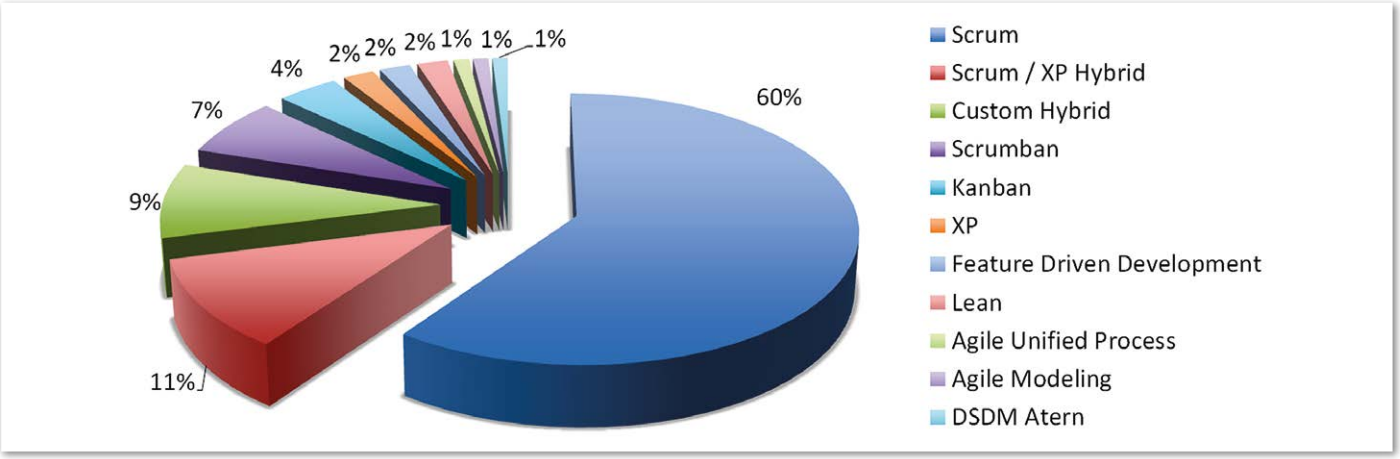

Abbildung 2: Die Verbreitung agiler Methoden

Abbildung 3: Schematische Darstellung der Scrum-Methode

1. Unsere höchste Priorität ist es, den Kunden durch frühe und kontinuierliche Auslieferung wertvoller Software zufriedenzustellen.
2. Heiße Anforderungsänderungen sind selbst spät in der Entwicklung willkommen. Agile Prozesse nutzen Veränderungen zum Wettbewerbsvorteil des Kunden.
3. Liefere funktionierende Software regelmäßig innerhalb weniger Wochen oder Monate und bevorzuge dabei die kürzere Zeitspanne.
4. Fachexperten und Entwickler müssen während des Projekts täglich zusammenarbeiten.
5. Errichte Projekte 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 Entwicklungsteams zu übermitteln, ist das Gespräch von Angesicht zu Angesicht.
7. Funktionierende Software ist das wichtigste Fortschrittsmaß.
8. Agile Prozesse fördern nachhaltige Entwicklung. Die Auftraggeber, Entwickler und Benutzer sollten ein gleichmäßiges Tempo auf unbegrenzte Zeit halten können.
9. Ständiges Augenmerk auf technische Exzellenz und gutes Design fördert Agilität.
10. Einfachheit – die Kunst, die Menge nicht getaner Arbeit zu maximieren – ist essenziell.
11. Die besten Architekturen, Anforderungen und Entwürfe entstehen durch selbstorganisierte Teams.
12. In regelmäßigen Abständen reflektiert das Team, wie es effektiver werden kann, und passt sein Verhalten entsprechend an.

Wer würde angesichts dieser in zwölf Thesen gegossenen humanistischen Grundhaltung schon widersprechen. Betrachtet man allerdings, wie heute in vielen Unternehmen auf das Manifest referenziert wird, so stellt man schnell fest, dass das Manifest für fragwürdige Argumentationen regelrecht missbraucht wird: Mal wird aus dem Manifest abgeleitet, dass eine Dokumentation völlig überflüssig ist, da nur noch lauffähige Software wichtig ist, mal wird postuliert, dass eine umfassende analytische Beschäftigung mit der Problemstellung kompletter Humbug ist, weil man ja nachher in kleinen Entwicklungseinheiten allen Details auf den Grund gehe. Dass es zu den zwölf Thesen noch eine detaillierte Erläuterung gibt, wie diese zu interpretieren sind und in der klar gesagt wird, dass saubere Dokumentation sehr wohl wichtig ist, wird von vielen Fans der agilen Software-Entwicklung gerne und bereitwillig ignoriert.

Wie leicht zu erkennen ist, handelt es sich beim Agilen Manifest nicht um eine Methode, ein Verfahren oder gar ein Vorgehensmodell, das beschreibt, wie ein Software-Projekt abzuwickeln ist, sondern eben lediglich um zwölf gut gemeinte Ratschläge, die nun wirklich nicht im Bereich der Nobelpreisverdächtigkeit liegen. Hier stellt sich die Frage, wie das Manifest in einem Projekt umgesetzt werden kann. Dabei kommen die verschiedenen agilen Verfahren ins Spiel, die auf dem Manifest aufsetzen und sich im Laufe der Zeit herausgebildet haben:

- Scrum
- Scrum / XP Hybrid
- Custom Hybrid
- Scrumban
- Kanban
- XP
- Feature Driven Development
- Lean
- Agile Unified Process
- Agile Modeling
- DSDM Atern

Scrum ist das Verfahren mit der größten Verbreitung. Laut dem aktuellen „7th Annual State of Agile Development Survey" *(siehe „http://www.versionone.com/pdf/7th-Annual-State-of-Agile-Development-Survey.pdf")* verwenden heute mehr als 54 Prozent der agilen Software-Projekte Scrum als Methode *(siehe Abbildung 2)*.

Vereinfacht gesagt, wird bei der Scrum-Methode ein Projekt in viele kleine Teilprojekte, sogenannte „Sprints", zerlegt. Ein Sprint erstreckt sich typischerweise über einen Zeitraum von ein bis vier Wochen und hat zum Ziel, eine Teilfunktion eines zu implementierenden Systems zu liefern. Scrum definiert auf organisatorischer Seite das Projekt-Team, den Scrum-Master, den Product Owner sowie den Kunden und umschreibt deren Aufgabe relativ detailliert.

Die Tasks eines Projekts werden im Product- und im Sprint-Backlog gesammelt *(siehe Abbildung 3)*. Für weitergehende Informationen sei auf die einschlägige Literatur verwiesen. Das Haupt-Augenmerk bei Scrum liegt auf den kurzen Entwicklungszyklen und der Forderung, zu jedem Zeitpunkt über einen lauffähigen Softwarestand zu verfügen.

Agile Stolpersteine

Einer der größten Stolpersteine bei der Einführung agiler Methoden besteht in der Art und Weise, wie diese im Unternehmen implementiert werden. Vielfach wird agiles Vorgehen als tollkühnes Piratenstück gesehen, das eben mal schnell von den Projekt-Mitgliedern eingesetzt wird.

Diese Revolution von unten fällt dem Management selbstverständlich irgendwann

auf. Bohrende Fragen nach den Neuerungen sind die Folge. Wenn diese Fragen ähnlich ungeschickt beantwortet werden, etwa durch einen kommentarlosen Hinweis auf das Agile Manifest, dann hat man sich das Management zum Gegner gemacht und sich aller Chancen beraubt, die Vorteile agiler Software-Entwicklung auszuspielen.

Da die Umstellung auf ein agiles Verfahren keine Kleinigkeit ist und die Änderungen recht schnell unternehmensweite Sichtbarkeit erreichen werden, ist man gut beraten, ein absolutes und bedingungsloses Management Commitment einzuholen und das Management in die agilen Prozesse einzubeziehen. Diese Einbeziehung ist insbesondere deshalb so wichtig, weil sich Agilität nicht einfach über Nacht im Unternehmen einführen lässt.

Die am agilen Prozess beteiligten Personen müssen ihre Rollen, Aufgaben, Pflichten und auch ihre Rechte im Rahmen der gewählten agilen Methode genau kennen. Nicht selten sind hierfür umfangreiche Budgets für Trainings erforderlich, bevor überhaupt an den Beginn eines agilen Entwicklungsprojekts zu denken ist.

Ein weiterer Stolperstein besteht in der Annahme, dass agile Methoden bedingungslos alle bisherigen Verfahren ersetzen und dass prinzipiell jedes Software-Projekt mit Agilität besser fährt. Dabei handelt es sich um einen fatalen Irrtum. Agile Methoden müssen als Alternative zu traditionellen, stark planungsorientierten Ansätzen verstanden werden und nicht als deren Ersatz. Zu dieser Erkenntnis kommen auch viele Verfechter agiler Methoden. Stellvertretend sei hier der Wikipedia-Eintrag zum Thema „Agile Softwareentwicklung“ zitiert: „Durch den Hype um agile Methoden werden diese manchmal fälschlicherweise als Allheilmittel bei Projektproblemen angesehen. Dies ist natürlich nicht so: Die Haupthinderungsgründe gelten für agile Verfahren genauso wie für traditionelle Verfahren.“

Insbesondere wird der Einsatz agiler Verfahren dann problematisch, wenn ein Projekt klar (vorher) definierte Anforderungen erfüllen muss und engen Zeit- oder Budget-Vorgaben unterliegt. Hier bieten die klassischen, ingenieursmäßigen Vorgehensmodelle mit klar definierten Phasen große Vorteile. Agile Verfahren eignen sich hingegen gut bei weichen und wenig ausformulierten Anforderungen beziehungsweise einem hohen Maß an externen Störfaktoren/Marktveränderungen *(siehe „http://de.wikipedia.org/wiki/Agile_Softwareentwicklung“)*. Leider wird in der Praxis in den meisten Fällen die Frage, ob im konkreten Fall ein agiles Verfahren angebracht ist oder nicht, in der Regel überhaupt nicht gestellt.

Ein agiles Entwicklungsprojekt benötigt, bevor der erste Sprint beginnen kann, ein Höchstmaß an Vorbereitung. Auch dieser Umstand wird sehr häufig vergessen. So ist unter anderem in der Vorbereitung zu klären, welche Sprints es geben soll, was deren Inhalte sind und welche Querbeziehungen es zwischen den Inhalten der einzelnen Sprints gibt. Werden Sprints ungeschickt geschnitten, sodass viele Sprints inhaltliche Abhängigkeiten zueinander haben, dann kann das im Verlauf der Bearbeitung der einzelnen Sprints zu endlosen Rework-Szenarien führen und somit das Projekt in den Abgrund reißen.

Vor Beginn des ersten Sprints sind nach Auffassung des Autors ein oder mehrere „agile Masterminds“ gefordert, die genau definieren, welche Sprints es gibt und was deren Inhalte sind. Dabei ist zu beachten, dass die Komplexität dieser Aufgabe einer komplexen Analyse in einem herkömmlichen Projekt in nichts nachsteht. Demnach muss der leider weitverbreiteten Ansicht, agile Projekte funktionierten getreu dem Motto „Einschalten und Loslegen“, eine klare Absage erteilt werden.

Betrachtet man die Verfahren agiler Software-Entwicklung, so wird schnell klar, dass sich diese mitunter recht gut auf das Entwickeln des Anwendungscodes übertragen lassen. Wie aber sieht es mit den anderen Anwendungsartefakten aus? Eignen sich diese ebenfalls für die Anwendung einer agilen Methodik? Stellt man diese Frage einem Projektleiter oder einem Projektteam, erntet man in der Regel ungläubige Blicke. Welche anderen Artefakte sollen denn da gemeint sein? Dass es neben ausführbarem Java-Code aber auch noch Artefakte wie ein Datenmodell oder ein Prozessmodell gibt, wird von den Protagonisten agiler Software-Entwicklung in der Regel völlig verdrängt. Beide Artefakte eignen sich aus Sicht des Autors jedoch in keiner Weise für eine Bearbeitung mit agilen Verfahren.

Bei einem Datenmodell handelt es sich um das Fundament einer Anwendung. Dieses sollte fertiggestellt sein und einer gewissen Stabilität genügen, bevor man mit anderen Artefakten darauf aufbaut. Schließlich würde auch kein Bauherr auf die Idee kommen, schon mal mit dem Bau des Erdgeschosses zu beginnen, solange der Keller noch nicht fertig ist.

Ferner gelten für das Datenmodell einer Anwendung leider nicht die gleichen Modularisierungskonzepte wie für den Anwendungscode. Man stelle sich etwa vor, in einem späten Sprint würde bekannt, dass in einer Tabelle des Datenmodells der Primärschlüssel leicht angepasst werden muss und dass sich dieser leider als Fremdschlüssel in vielen Tabellen wiederfindet. Diese Situation wird dazu führen, dass viele Ergebnisse aus früheren Sprints überarbeitet werden müssen, weil die „unscheinbare kleine lokale Änderung" weitreichende Konsequenzen für die komplette Anwendung hat.

Betrachtet man die aktuelle Literatur zum Thema „agile Software-Entwicklung", dann stellt man fest, dass derlei Fragestellungen dort leider komplett ausgeblendet sind. Für Datenbank-basierte Anwendungen ist aber genau dieses Problem eine klaffende Wunde im ansonsten so wunderschönen Körper der agilen Methodik, die es zu behandeln gilt.

Das zweite Artefakt, das sich nach Meinung des Autors nicht für die Anwendung von agilen Verfahren eignet, ist der Geschäftsprozess. Natürlich lassen sich Prozesse je nach gewähltem Modellierungsverfahren sehr schön über Sub-Prozesse modularisieren und eignen sich somit augenscheinlich sehr gut für eine Implementierung im Rahmen diverser Sprints. Was sicher richtig ist, ist die Feststellung, dass sich eine Prozess-Analyse aufgrund der Modularisierbarkeit optimal für ein agiles Vorgehen eignet; dabei jedoch gleich die Implementierung miteinzubeziehen und einen Sprint über die Analyse, das Design und die Implementierung eines Geschäftsprozesses laufen zu lassen, muss als absolut töricht bezeichnet werden. Hier gilt nach wie vor die alte Weisheit, dass man einen Prozess in seiner Gänze verstanden haben sollte, bevor man ihn umsetzt.

Die Einführung agiler Verfahren stellt weniger einen technologischen Wechsel als vielmehr einen sozialen/kulturellen Wandel dar. Obwohl sich eine Mehrheit der in diversen Studien befragten Entwickler kompromisslos für agile Verfahren entscheidet, scheint vielen der Beteiligten nicht klar zu sein, welche Veränderungen damit verbunden sind. Betrachtet man das Agile Manifest und dessen zwölf Thesen, dann kann man schnell erkennen, was sich mit Einführung agiler Verfahren alles ändern wird:

- Fachexperten und Entwickler müssen während des Projekts täglich zusammenarbeiten. Diese bedingungslose Nähe zwischen Entwicklung und Fachbereich ist nicht unbedingt jedermanns Sache. Viele Entwickler müssen hier eine über die Jahre hinweg kultivierte Einigelung in den eigenen vier Wänden der Entwicklung aufgeben.
- Die effizienteste und effektivste Methode, Informationen an und innerhalb eines Entwicklungsteams zu übermitteln, ist das Gespräch von Angesicht zu Angesicht. Gute und effiziente Kommunikation ist jedoch seit jeher ein Problem in den meisten Software-Projekten, egal ob diese agil abgewickelt werden oder nicht. Fraglich ist jedoch, ob es ausreichend ist, dass sich alle Entwickler einmal pro Tag in einem Kreis aufstellen und miteinander sprechen. Klar ist, dass agile Projekte keine „Knowledge Hider" und Kommunikationsmuffel dulden.
- Die besten Architekturen, Anforderungen und Entwürfe entstehen durch selbstorganisierte Teams. Das mag wohl sein, aber auch dabei handelt es sich um ein Problem aus dem Bereich der sozialen Kompetenz, mit dem jedes Projekt zu kämpfen hat, und das unabhängig von der gewählten Methode.
- Liefere funktionierende Software regelmäßig innerhalb weniger Wochen oder Monate und bevorzuge dabei die kürzere Zeitspanne. Dabei handelt es sich wohl um die Kernforderung des Agilen Manifests. Viele Entwickler sind sich aber nicht im Klaren darüber, dass genau diese Forderung ein Maximum an Transparenz und Nachprüfbarkeit der Leistung eines jeden Entwicklers verlangt – was sicher nicht überall willkommen ist.

Ein weiterer sehr unschöner Aspekt, der sehr oft in agilen Software-Projekten anzutreffen ist, besteht darin, das Konzept der agilen Software-Entwicklung als Begründung für jeden Missstand zu verwenden. Die folgenden Situationen sind leider sehr häufig in agilen Projekten anzutreffen:

- Die Dokumentation ist oftmals noch deutlich schlechter und unvollständiger als in traditionellen Projekten. Laut Aussage des Teams muss das so sein, weil bei agiler Vorgehensweise Software wichtiger ist als Dokumentation. In solchen Fällen sollte man dem Team die genaue Lektüre des Agilen Manifestes nahelegen, da dort nämlich kein Wort davon steht, dass die Dokumentation vernachlässigt werden kann, ganz im Gegenteil. Hier handelt es sich schlicht um eine dreiste Ausrede.
- Datenmodelle ändern sich oft täglich, mitunter sogar stündlich. Gelegentlich kommt es sogar vor, dass bereits gemachte Änderungen zurückgezogen und nur einen Tag später erneut angefordert werden. Fragt man auch hier das Projekt nach dem Grund, dann bekommt man zur Antwort, das müsse so sein, weil man bei agiler Vorgehensweise sehr flexibel agiert. Auch in dieser Situation sollte man sich keinen Bären aufbinden lassen; ein solches Vorgehen hat weniger mit Agilität als mit Kopflosigkeit und schlechter Organisation zu tun. Ein Ziel der agilen Software-Entwicklung besteht nämlich darin, dass die in den einzelnen Sprints erarbeiteten Ergebnisse Bestand haben und nicht ständig überarbeitet werden müssen.
- Prozess-Komponenten werden bewusst als sogenannte „adaptive Black Box" offen gelassen und nur sehr oberflächlich dokumentiert. Die Begründung besteht oft darin, dass das ein Prozessteil sei, der sehr individuell und flexibel gehalten werden müsse, weil die an diesem Teilprozess beteiligten Akteure unheimlich intelligent und kreativ seien und nicht durch eine Prozessvorgabe in ihrer Kreativität eingeengt werden sollten. Auch hier sollte die Aussage kritisch hinterfragt werden. In der Regel war man einfach nur zu bequem, um den Teilprozess sauber zu analysieren und entsprechend hochwertig zu dokumentieren.

Fazit

Agile Methoden haben sich zu Recht für bestimmte Anwendungsfälle neben traditionellen, stark planungsorientierten Verfahren der Software-Entwicklung etabliert. Die Vorteile agiler Software-Entwicklung sind unübersehbar: Komplexe Projekte werden in überschaubare Einheiten unterteilt, die schnell fertiggestellt werden können, Entwickler arbeiten in agilen Umfeldern oftmals effizienter und die strikte Unterteilung in kleinere Arbeitspakete führt zu einer besseren Überwachung der Zielerreichung.

Trotzdem kommen die erwähnten Studien nicht zu der Erkenntnis, dass agile Ansätze in Summe effizienter sind als traditionelle Verfahren. In den meisten Fällen liegt das allerdings nicht an etwaigen Schwächen agiler Verfahren, sondern an der vielfach unzureichenden Implementierung agiler Methoden.

Sebastian Graf
sebastian.graf@promatis.de