Agilität und Adaption – Fluch oder Segen der industriellen Softwareentwicklung

Sebastian Graf
PROMATIS software GmbH
76275 Ettlingen

Schlüsselworte
Agile Softwareentwicklung, Projektmanagement, Business Process Management, Requirements Management, Scrum, XP, Kanban

Einleitung
Vor geraumer Zeit wurde ich bei einem Kunden von der Bereichsleitung gebeten, einem Krisenmeeting beizuwohnen und mir einen Überblick über das zur Diskussion stehende Projekt zu verschaffen. Eigentlich hätte ich die Besprechung nach 10 Minuten wieder verlassen können, da mir recht schnell klar war, dass das Projekt unter gar keinen Umständen zu retten sein würde. Alle Termine waren hoffnungslos überzogen, die zum damaligen Zeitpunkt durchgeführten Integrationstests waren zu 96% fehlerhaft, und die Fachabteilungen hatten das Projekt längst abgeschrieben. Um mir Klarheit über die Ursachen zu verschaffen, habe ich in dem Meeting nach dem Projektplan, dem Datenmodell der Anwendung und der Anforderungsdokumentation gefragt. Geerntet habe ich ungläubige Blicke - aus heutiger Sicht bin ich froh damals nicht des Raumes verwiesen worden zu sein: „Herr Graf, wir arbeiten agil und setzen adaptive Verfahren ein ...” Diese Antwort war damals der Grund dafür, mich etwas eingehender mit dem Thema Agile Softwareentwicklung zu beschäftigen. Und das abseits der üblichen Jubelpublikationen zum Thema. Besonders interessiert haben mich dabei die folgenden Fragen:

- Warum werden in agilen Projekten mehr oder weniger große Teile des Codes drei- bis viermal entwickelt, bis die Anwender einmal damit zufrieden sind?
- Warum ändert sich das Datenmodell in einem agilen Projekt jeden Tag getreu dem Motto: „Und 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 werden fachliche Prozesse in vielen agilen Projekten vollkommen missachtet und finden keinen Eingang in die Anwendung?

Ziel der Präsentation soll sein, den interessierten Zuhörer in die Lage zu versetzen zu entscheiden, ob sich für einen konkreten Einsatzfall agile Methoden tatsächlich lohnen oder ob man mit herkömmlichen, stark planungsorientierten Ansätzen doch besser fahren würde. Ferner soll der Zuhörer in die Lage versetzt werden, die Ausrede „... aber wir arbeiten doch agil, das muss so sein ...“ als solche zu identifizieren und zu erkennen, wenn der Kaiser gar keine Kleider anhat.

Studien zum Thema Agile Softwareentwicklung
Mittlerweile sind agile Verfahren weit verbreitet, und viele Unternehmen investieren größere Summen in die Umstellung ihrer Entwicklungsprozesse auf agile Verfahren. So ist es auch kaum verwunderlich, dass das Internet voll von Studien zum Thema Agile Softwareentwicklung ist, die

selbstverständlich alle zum Schluss kommen, dass agile Methoden den klassischen Verfahren in puncto Qualität der Lösung und Effizienz des Entwicklungsprozesses haushoch überlegen sind. 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 z.B. 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,* welche diverse Studien zum Thema bewertet und vergleicht, zu der klaren Erkenntnis, dass es keinerlei belegbare Hinweise dafür gibt, 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 3,5-mal so viele Lines of Code hervorgebracht hatte, wie das Team welches 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 sich um ein praxiserprobtes Konzept handelt, welches von Praktikern für Praktiker ersonnen wurde.
Im Nachgang des eingangs erwähnten 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 Administratoren eine Auswertung bezüglich der Änderungshäufigkeit mit Bezug auf das Datenmodell in Auftrag gegeben. Diese Untersuchung hat sehr interessante Erkenntnisse zutage gefördert, die in Abbildung 1 beschrieben sind. Bei der Bewertung der Zahlen ist zu berücksichtigen, dass in fraglichem Projekt in einer vorgelagerten Analysephase das komplette der Anwendung zugrunde liegende Datenmodell erstellt werden sollte. Lediglich die Anwendungskomponenten sollten nach agilen Methoden entwickelt werden. Gegen den Protest der Datenbankexperten wurden in der Analysephase aber Detailfragen 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 in Abbildung 1 zu sehen ist, 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 Datenbankexperten eine schiere Horrorvorstellung.

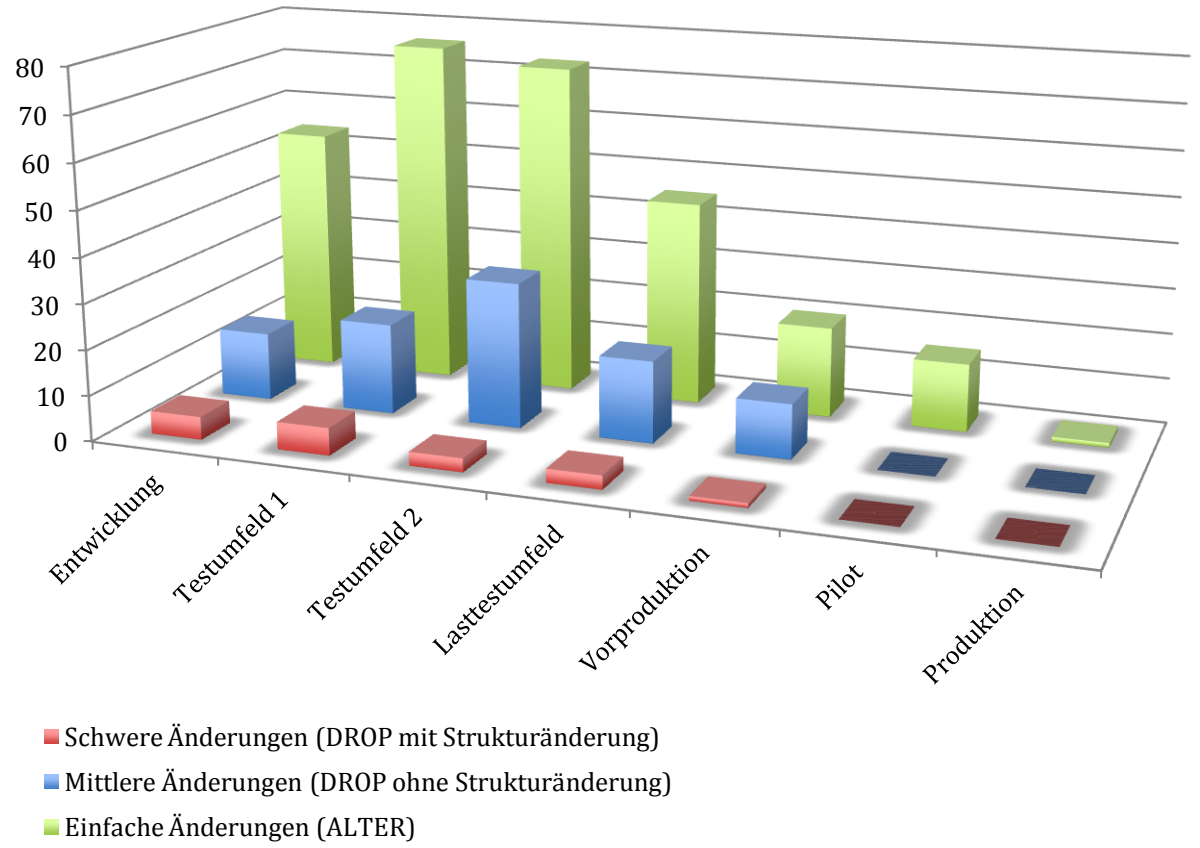

Abb. 1: Wöchentliche Änderungsrate nach abgeschlossener Systemanalyse

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 Softwareentwicklung dargelegt werden.

Grundprinzipien der Agilen Softwareentwicklung

Das Thema Agilität in der Softwareentwicklung ist eigentlich 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 Softwarekrise haben sich einige namhafte Softwareentwickler mit dem Agilen Manifest auf 12 Prinzipien der Softwareentwicklung verständigt. Diese sind einleuchtend und enthalten einfache und klare Botschaften:

1. Unsere höchste Priorität ist es, den Kunden durch frühe und kontinuierliche Auslieferung wertvoller Software zufriedenzustellen.

2. Heisse Anforderungsänderungen 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 im 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 12 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 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 12 Thesen des Agilen Manifestes noch eine detaillierte Erläuterung gibt, wie diese Thesen zu interpretieren sind und in der unter anderem klar gesagt wird, dass saubere Dokumentation sehr wohl wichtig ist, wird von vielen Fans der agilen Softwareentwicklung 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, welches beschreibt, wie ein Softwareprojekt abzuwickeln ist, sondern eben lediglich um 12 gut gemeinte Ratschläge, die nun wirklich nicht im Bereich der Nobelpreis-Verdächtigkeit liegen. Nun stellt sich die Frage, wie das Manifest in einem Projekt umgesetzt werden kann. Hier 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

Wobei aktuell Scrum das Verfahren mit der größten Verbreitung darstellt. Laut dem aktuellen „7th Annual State of Agile Development Survey“ (http://www.versionone.com/pdf/7th-Annual-State-of-Agile-Development-Survey.pdf) verwenden heute mehr als 54% der agilen Softwareprojekte Scrum als Methode.

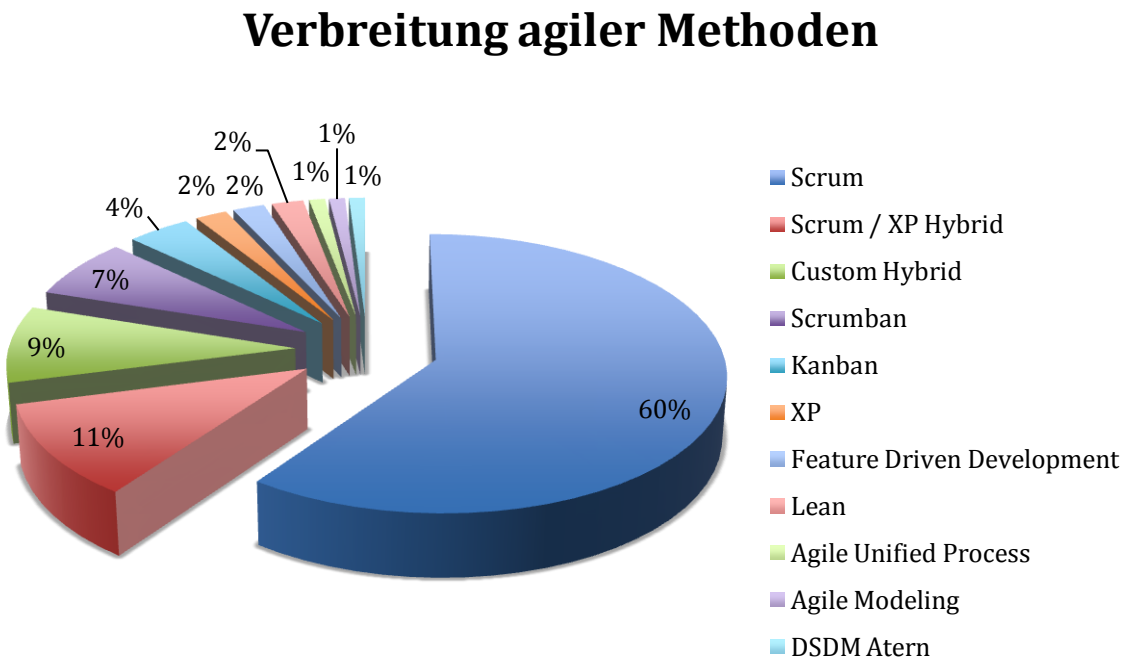

Abb. 2: Verbreitung agiler Methoden

Vereinfacht gesagt wird bei der Scrum Methode ein Projekt in viele kleine Teilprojekte, sog. 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 Projektteam, den Scrum Master, den Product Owner und den Kunden und umschreibt deren Aufgabe relativ detailliert. Die Tasks eines Projekts werden im Product Backlog und im Sprint Backlog gesammelt (vgl. Abbildung 2). Für weitergehende Informationen sei der interessierte Leser auf die einschlägige Literatur verwiesen. Das Hauptaugenmerk bei Scrum liegt auf den kurzen Entwicklungszyklen und der Forderung, zu jedem Zeitpunkt über einen lauffähigen Softwarestand zu verfügen.

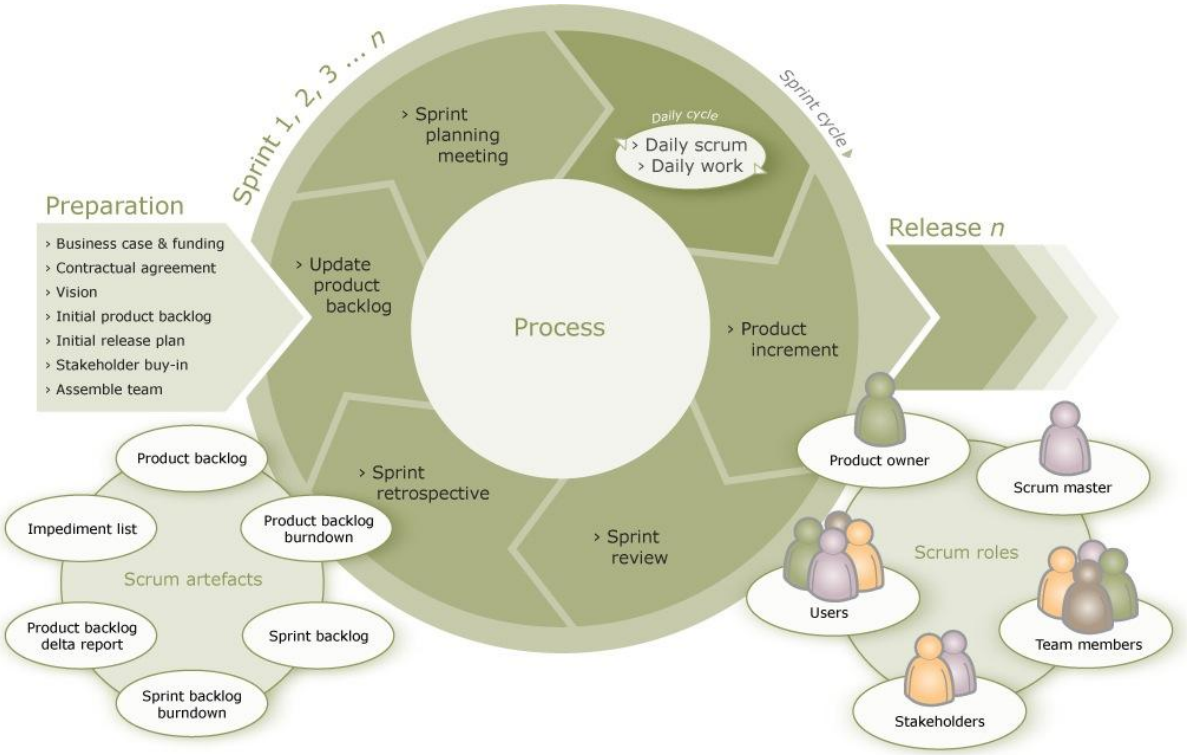

Abb. 3: Schematische Darstellung der Scrum Methode

Agile Stolpersteine – Missachtung des Managements

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, welches eben mal schnell von den Projektmitgliedern eingesetzt wird. Diese „Revolution von unten" fällt dem Management selbstverständlich irgendwann auf. Bohrende Fragen nach den Neuerungen sind die Folge. Und 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 hat sich aller Chancen beraubt, die Vorteile agiler Softwareentwicklung auszuspielen. Da die Umstellung auf ein agiles Verfahren eben 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 des Managements 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, ihre Aufgaben und Pflichten und auch ihre Rechte im Rahmen der gewählten agilen Methode genau kennen. Nicht selten müssen hierfür umfangreiche Budgets für Training bereitgestellt werden, bevor überhaupt an den Beginn eines agilen Entwicklungsprojekts zu denken ist.

Agile Stolpersteine – Agilität ist das Allheilmittel

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 dann der Einsatz agiler Verfahren problematisch, wenn ein Projekt klar (vorher) definierte Anforderungen erfüllen muss und engen Zeit- oder Budgetvorgaben 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, bzw. einem hohen Maß an externen Störfaktoren/Marktveränderungen.
(vgl. 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.

Agile Stolpersteine – Agilität ist einfach
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 nämlich ungeschickt geschnitten, so dass viele Sprints inhaltliche Abhängigkeiten zueinander haben, dann kann das im Verlauf der Bearbeitung der einzelnen Sprints zu endlosen Rework Szenarien und somit das Projekt in den Abgrund führen. 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.

Agile Stolpersteine – Agilität ist gut für jedes Projektartefakt
Betrachtet man die Verfahren agiler Softwareentwicklung so wird schnell klar, dass sich diese Verfahren mitunter recht gut auf das Entwickeln des Anwendungscodes anwenden 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 derlei Artefakte wie ein Datenmodell oder ein Prozessmodell gibt, wird von den Protagonisten agiler Softwareentwicklung 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 z.B. nur vor, dass in einem späten Sprint bekannt wird, dass in einer Tabelle des Datenmodells der Primärschlüssel „leicht“ angepasst werden muss, und dass sich dieser Primärschlüssel 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 auf die komplette Anwendung hat. Betrachtet man die aktuelle Literatur zum Thema agile Softwareentwicklung, dann stellt man fest, dass derlei Fragestellung dort leider komplett ausgeblendet werden. Für datenbankbasierte 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, welches sich nach Meinung des Autors nicht für Anwendung von agilen Verfahren eignet, ist der Geschäftsprozess. Natürlich lassen sich Prozesse je nach gewähltem Modellierungsverfahren sehr schön über Subprozesse 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 Prozessanalyse auf Grund der Modularisierbarkeit optimal für ein agiles Vorgehen eignet, aber dabei 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.

Agile Stolpersteine – Die soziale Komponente

Die Einführung agiler Verfahren stellt weniger einen technologischen Wechsel als vielmehr einen sozialen / kulturellen Wandel dar. Obwohl sich eine überwiegende Mehrzahl der in diversen Studien befragten Entwickler kompromisslos für agile Verfahren entscheiden, scheint vielen der Beteiligten nicht klar zu sein, welche Veränderungen damit verbunden sind. Betrachtet man das Agile Manifest und dessen 12 Thesen, dann kann man eigentlich recht schnell einen Eindruck davon bekommen, 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 im Gespräch von Angesicht zu Angesicht*. Gute und effiziente Kommunikation ist 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 Manifestes. 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. Ein Umstand, der sicher nicht überall mit offenen Armen empfangen wird.

Agile Stolpersteine – Agilität als perfekte Entschuldigung

Ein weiterer sehr unschöner Umstand, der in agilen Softwareprojekten sehr oft anzutreffen ist, besteht darin, das Konzept der agilen Softwareentwicklung 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 erneut zurückgezogen werden nur um einen Tag später erneut angefordert zu werden. Fragt man auch hier das Projektteam, warum dem so ist, dann bekommt man zur Antwort, dass das so sein müsse, 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 Softwareentwicklung besteht nämlich darin, dass die in den einzelnen Sprints erarbeiteten Ergebnisse Bestand haben und nicht ständig überarbeitet werden müssen.

- Prozesskomponenten werden bewusst als so genannte adaptive Black Box offen gelassen und nur sehr oberflächlich dokumentiert. Die Begründung besteht oft darin, dass das ein Prozessteil wäre, der sehr individuell und flexibel gehalten werden müsse, weil die an diesem Teilprozess beteiligten Akteure unheimlich intelligent und kreativ sind 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.

Zusammenfassung
Agile Methoden haben sich zu Recht für bestimmte Anwendungsfälle neben traditionellen, stark planungsorientierten Verfahren der Softwareentwicklung etabliert. Die Vorteile agiler Softwareentwicklung 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 eingangs erwähnten Studien leider nicht zu der Erkenntnis, dass agile Ansätze in Summe effizienter sind als traditionelle Verfahren. In den meisten Fällen liegt das aber nicht an etwaigen Schwächen agiler Verfahren, sondern in der vielfach unzureichenden Implementierung agiler Methoden. Diese, meist menschlich und nicht technisch bedingten Schwachstellen, müssen in einem Projekt konsequent über den gesamten Projektverlauf angegangen werden, um letztlich den Erfolg agiler Softwareentwicklung sicherzustellen.

Kontaktadresse:
Dipl.-Inform. Sebastian Graf
PROMATIS software GmbH
Pforzheimer Str. 160
D-76275 Ettlingen

Telefon:	+49 (0) 7243-2179-0
Fax:	+49 (0) 7243-2179-99
E-Mail	sebastian.graf@promatis.de
Internet:	http://www.promatis.de
	http://www.horus.biz