AGILES VORGEHEN IM PROJEKTGESCHÄFT: WUNSCH ODER REALITÄT?

Produkthersteller und interne IT-Abteilungen haben häufig eine bessere Kontrolle über ihre Vorgehensweise als Softwarehäuser, die Projektgeschäft betreiben. Während erstere selbstständig – unter Einbeziehung der Fachabteilungen – über die Einführung einer agilen Methode entscheiden können, müssen letztere stärker die Gegebenheiten ihrer Kunden berücksichtigen. Dieser Artikel beschreibt, wie die zwölf Prinzipien, die hinter dem Agilen Manifest stehen, im Alltag eines Projekts weitestmöglich umgesetzt werden können, um die Brücke zwischen einem optimalen agilen Vorgehen und den jeweiligen Gegebenheiten auf einer nicht-agilen Kundenseite zu schlagen. Im Ergebnis wird ein flexibles Projektmodell erarbeitet, das in der Praxis in verschiedenen Situationen angewendet werden kann und das der agilen Idee bestmöglich gerecht wird. Darüber hinaus werden Praxistipps für dessen Umsetzung gegeben.

der autor

Stefan Fleckenstein
(E-Mail: stefan.fleckenstein@mwea.de)
ist CTO bei der MaibornWolff et al GmbH in München. In seinen 17 Berufsjahren hat er sich intensiv mit Softwarearchitekturen und Vorgehensmodellen auseinandergesetzt.

Seit der Veröffentlichung des Agilen Manifests vor zehn Jahren sind agile Methoden mittlerweile Teil des Mainstream geworden. Lag der Fokus anfangs hauptsächlich noch auf *eXtreme Programming (XP)*, ist Scrum (vgl. [Scr09]) derzeit das Mittel der Wahl. Aber auch andere Methoden, wie beispielsweise Kanban (vgl. [Roo10]), finden immer weitere Verbreitung.

Während ein Produkthersteller die Ausgestaltung der Vorgehensweise selbst in der Hand hat und über einen längeren Zeitraum den Prozess anpassen und verbessern kann, stehen Softwarehäuser im Projektgeschäft vor einer anderen Herausforderung: Alle Kunden sind unterschiedlich und viele haben feststehende Vorgehensweisen, an die sie sich über die Zeit hinweg gewöhnt haben. Häufig müssen viele Stakeholder (Fachbereiche, Unternehmensarchitektur, Qualitätssicherung, Wartungsteams und Betrieb) frühzeitig einbezogen werden und bestehen auf abgenommene Fach- und DV-Konzepte. Oft sind zudem die IT-Abteilungen der Kunden nur mäßig flexibel und der Paradigmenwechsel durch ein komplett agiles Projektmodell ist meist zu schwierig, um ihn in einem einzelnen Projekt zur Realisierung eines Anwendungsprogramms durchführen zu können.

Wie also kann man hier als Softwarehaus den Spagat zwischen Wunsch und Realität bewerkstelligen? Wie können in diesem Umfeld möglichst viele Elemente der agilen Vorgehensweisen eingesetzt und ihre Vorteile für den Projekterfolg genutzt werden?
Häufig ist die Kluft gar nicht so groß, wie sie zu Beginn scheint. Auch mit agilen Methoden werden Anforderungen erfasst, eine gut entworfene und qualitativ hochwertige Software erstellt und getestet. Ebenso wird ein regelmäßiger Planungsprozess durchgeführt. Die Elemente der klassischen Durchführung von Softwareprojekten sind also auf sehr abstraktem Level weiterhin vorhanden, werden jedoch anders angeordnet, mit anderen Techniken durchgeführt und unterschiedlich gewichtet.

Prinzipien als Grundlage für ein Vorgehensmodell

Die vier Werteabwägungen des Agilen Manifests sind sehr prominent. Deutlich weniger im Fokus stehen jedoch die dahinterliegenden zwölf Prinzipien (vgl. [Agi]). Dabei liefern aber gerade diese eine wertvolle Hilfe, wie agile Vorgehensweisen in unterschiedlichen Umfeldern funktionieren und helfen, den Projekterfolg zu sichern.

Auch wenn es in der Auflistung im Agilen Manifest nicht auf den ersten Blick ersichtlich ist, so stehen die Prinzipien doch in einem engen inhaltlichen Zusammenhang. Viele bedingen sich gegenseitig, sodass man keines der Prinzipien einfach über Bord werfen kann, ohne das gesamte Ziel eines besseren Vorgehens durch Agilität zu gefährden.

In Abbildung 1 habe ich eine Gruppierung der Prinzipien vorgenommen. Anhand dieser Einteilung werden sie in diesem Artikel auf ihren Einsatz in einem nicht hundertprozentig agilen Umfeld untersucht. Dabei wird Stück für Stück eine Werkzeugkiste mit Methoden zur Steuerung von Entwicklungsprojekten gefüllt. Die Summe der Inhalte dieser

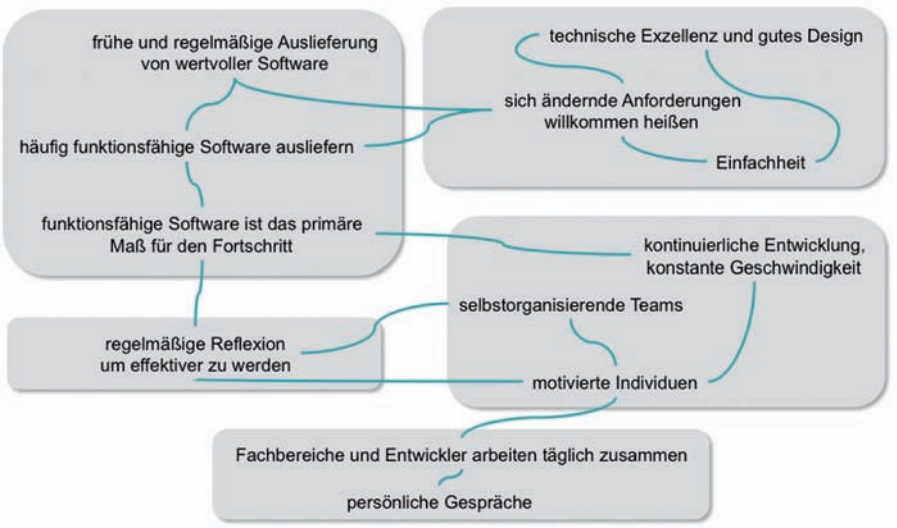

Abb. 1: Zusammenhang und Gruppierung der zwölf Prinzipien des Agilen Manifests.

Werkzeugkiste kann dann genutzt werden, um beim Einsatz in einem konkreten, nicht-agilen Kundenumfeld ein passendes Vorgehensmodell zu modellieren. Dies kann in zwei Richtungen erfolgen:

- Zum einen kann der Kunde ein Vorgehensmodell vorgeben, das nicht agil ist. Dann sollten Sie zu Beginn des Projekts ein intelligentes Tailoring des vorgegebenen Modells durchführen, um die Prinzipien möglichst weitgehend umzusetzen.
- Die zweite Möglichkeit besteht darin, dass der Kunde keine konkreten Vorgaben zum Vorgehensmodell macht, aber viele nicht-agile Inhalte erwartet, wie etwa abgenommene Konzepte, bevor mit der Realisierung begonnen werden darf. Wenn die Organisation des Kunden nicht bereit für ein agiles Projekt ist, können Sie auf einer agilen Vorgehensweise wie Scrum aufsetzen und diese so anpassen, dass die Ansprüche des Kunden gewahrt bleiben.

In den folgenden Abschnitte orientiere ich mich an der oben vorgenommen Gruppierung. Ich beginne jeweils mit einer Auflistung der Prinzipien, deren Bedeutung für das Projektmodell dann im weiteren Verlauf des Abschnitts diskutiert wird.

Ein frühzeitig verfügbares, benutzbares Anwendungssystem

- Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
- Working software is the primary measure of progress.

Der früher so häufig zitierte Wasserfall, in dem ein Projekt nach langen Fach- und DV-Konzeptphasen erst spät in die Realisierungs- und Testphasen eintritt, ist schon lange nicht mehr aktuell. Auch in klassischen Modellen werden Projekte in kleinere und einfacher zu handhabende Schritte unterteilt. Kunden sind typischerweise daran gewöhnt, dass Projekte in mehrere Stufen aufgeteilt werden. Die einzelnen Phasen werden dabei in kürzeren Zyklen

Abb. 2: *Vom Wasserfall zur Kaskade.**

**) Beide Bilder sind Wikipedia Commons entnommen und dort als Public Domain lizensiert:*
http://commons.wikimedia.org/wiki/File:2005SomeWaterfallInIceland.jpg
http://commons.wikimedia.org/wiki/File:Oirase_Waterfall.jpg

durchlaufen und damit wird die Software in Inkrementen erstellt (siehe Abbildung 2).

Das können Sie sich zunutze machen, indem Sie beim Tailoring des Vorgehens ein wenig radikaler vorgehen, als es der Kunde vielleicht gewöhnt ist. Machen Sie jede Stufe so kurz, wie es irgendwie geht.

Besonders kritisch ist immer die erste Stufe, da hier viele Grundlagen erstellt werden müssen. Häufig schlägt sich dies in langen Phasen mit ausgedehnten Konzepten und einer großen ersten Softwareauslieferung mit entsprechend umfangreichen Tests nieder. Dass dabei das Vertrauen des Kunden lange Zeit auf die Probe gestellt wird, da er monatelang nur Papier bewerten kann, war auch schon vor der Erstellung des Agilen Manifests ein bekanntes Problem.

Machen Sie es sich zum Ziel, eine *erste Stufe nach spätestens einem Monat* an den Kunden auszuliefern. Muss die erste Stufe wirklich in Produktion gehen? „Valuable, working software", so wie es die agilen Prinzipien fordern, heißt nicht notwendigerweise auch produktiv einsetzbare Software. Ein erstes Release kann beispielsweise auch mit sehr eingeschränkter Funktionalität auf eine Testumgebung des Kunden geliefert und dort begutachtet werden. Das ist sicher auch in großen Projekten und mit einem entsprechend kurzen Zyklus der Konzeption möglich.

Weitere Stufen sollten Sie dann in regelmäßigen Zyklen *mit einer Länge von maximal einem Monat* im *Timeboxing*-Verfahren an den Kunden ausliefern. Legen Sie dabei großes Augenmerk auf eine möglichst frühe produktive Stufe, die ein Mindestmaß an fachlicher Funktionalität besitzt. Eine gute Möglichkeit ist beispielsweise die Nutzung eines fachlichen Piloten, der zwar noch eingeschränkt sein kann (fachlich wie technisch) und nur für eine kleine Gruppe von Anwendern zur Verfügung gestellt wird. Damit können Konzepte in einem kontrollierten Umfeld im Echtbetrieb beobachtet werden. Die daraus gewonnenen Erkenntnisse können häufig schon zu ersten Justierungen des fachlichen Ziels führen, für deren Behandlung im Fall eines Festpreis-Projekts ein *Change-Request*-Verfahren aufgesetzt sein sollte.

Neben dem frühen Feedback des Kunden auf die ersten gelieferten Stufen gibt es noch einen weiteren wichtigen Effekt: Die Arbeit mit dem Fachbereich wird verstetigt. Im klassischen Wasserfall wird der Fachbereich nur zu Beginn mit dem Fachkonzept und am Ende mit umfangreichen Tests gefordert. Dazwischen wird er quasi

nicht beansprucht. In einem agileren Modell bedarf es einer stetigen Mitarbeit von Fachbereichsmitarbeitern für fachlichen Input und Tests. Dafür werden sie aber nur für kürzere Zeiteinheiten pro Woche involviert und es gibt weniger Konfliktpotenzial beim Einsatz dieser vielbeschäftigten Mitarbeiter. Das ist ein wichtiges Argument, wenn Sie mit dem Kunden über ein Vorgehen mit häufigeren, dafür aber kürzeren Phasen diskutieren.

Einfachheit und Qualität als Schlüssel zur Flexibilität

- Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
- Simplicity – the art of maximizing the amount of work not done – is essential.
- Continuous attention to technical excellence and good design enhances agility.

Agile Vorgehensweisen legen weniger Gewicht auf schriftliche Dokumentation als traditionelle Verfahren. Damit wird Ballast reduziert, der bei der Umsetzung von Änderungsanforderungen obsolet wird oder angepasst werden muss. In der Produktentwicklung führen typischerweise stabile Teams die Entwicklung und weitere Betreuung durch. Diese Teams kennen die Anwendung gut – sowohl fachlich als auch technisch. In Kundenprojekten sieht die Lage häufig anders aus: Ein Team des Auftragnehmers entwickelt die Anwendung und legt sie am Ende des Projekts in die Hände des Kunden. Die Wartung übernimmt dann entweder der Kunde selbst, der bisherige Auftragnehmer oder aber ein dritter Partner. Im Allgemeinen möchten sich die Kunden jedoch nicht vom ursprünglichen Auftragnehmer abhängig machen. Allein deshalb sollte Wert auf eine angemessene Menge fachlicher und technischer Dokumentation gelegt werden, die dem zukünftigen Wartungsteam hilft, die dem Code zu Grunde liegenden Zusammenhänge und Konzepte besser zu verstehen.

Ein weiterer Grund für ein Mindestmaß an Schriftlichkeit ist, dass Flexibilität nicht mit Beliebigkeit verwechselt werden darf. Ein Projekt muss zu jedem Zeitpunkt ein klares Ziel verfolgen. Dafür ist es von Beginn an notwendig, die funktionalen und nicht-funktionalen Anforderungen an das neue System zu erfassen. Ein *inkrementelles Vorgehen* – im Zusammenhang mit den oben beschriebenen Zyklen für die Auslieferung von Software – kann wie folgt aussehen:

- Erstellen Sie im Rahmen der ersten Phase eine leichtgewichtige Version der fachlichen Konzeption. Diese liefert einen wertvollen Überblick über die funktionalen Anforderungen des gesamten geplanten Systems. Eine sehr gute Praxis dafür ist eine Sammlung von kurzen User-Storys zusammen mit einem groben, aber dennoch vollständigen Modell der Geschäftsprozesse. Damit geben Sie zum einen dem Kunden sehr schnell das Gefühl, dass Sie sein Problem verstanden haben. Zum anderen legen Sie eine natürliche Struktur fest, die Sie für die Planung der einzelnen Iterationen nutzen können.
- Für jede geplante Auslieferung müssen die dafür vorgesehenen User-Storys genauer spezifiziert werden. Eine gute Möglichkeit der Strukturierung von User-Storys gibt es beispielsweise von Jeff Patton mit Hilfe von *Story Maps* (vgl. [Pat09]). Die detailliertere Beschreibung kann in unterschiedlichen Formen erfolgen: Neben Freitext stellen auch Akzeptanzkriterien oder Business-Regeln eine gute Möglichkeit dar. Auch ein detaillierteres Prozessmodell hilft oft bei der Erarbeitung und Dokumentation der Zusammenhänge. Diese Arbeit findet zu Beginn oder im Lauf einer Iteration statt. Eine weitere Möglichkeit ist, dass ein Teil des Teams innerhalb einer Iteration bereits die Detaillierung für die Themen der nächsten Iteration vornimmt.

Im Gegensatz zu einer aufwändigen Spezifikation zu Projektbeginn wächst so die fachliche Dokumentation stetig mit der Projektlaufzeit. Das kommt auf der einen Seite den Wünschen des Kunden nach schriftlicher Dokumentation entgegen, zum anderen erhöht die inkrementelle Vorgehensweise aber die Flexibilität bei Änderungen zur Projektlaufzeit. Dadurch kann auch die Gesamtmenge an fachlicher Dokumentation deutlich geringer ausfallen als bei einer klassischen Spezifikation im Wasserfall. Der Grund dafür ist, dass deutlich weniger Redundanzen zwischen Dokumentation und Code notwendig sind. Beispielsweise müssen keine Bildschirmmasken als Prosatext spezifiziert werden, da stattdessen direkt auf die Entwicklung verwiesen werden kann.

Das gleiche inkrementelle Vorgehen können Sie zur Erstellung einer technischen Dokumentation wählen. Zur Diskussion und Darstellung der Modularisierung genügt in der Startiteration eine Architekturvision mit wenigen Komponenten-Diagrammen und Laufzeit-Sichten sowie einem Deployment-Diagramm. Da Sie sich frühzeitig auch mit querschnittlichen Themen – wie Fehlerbehandlung und ähnlichem – auseinandersetzen müssen, können Sie diese auch gleich kurz dokumentieren. Eine möglichst kurze Fassung in der Struktur von [Hru10] gibt einen guten Überblick über die Architektur und dient zudem als Einstieg in den Quellcode. Im Laufe der einzelnen Stufen kann die technische Dokumentation bei Bedarf detailliert werden, um Konzepte der Modularisierung besser verständlich zu machen. Wie bei der fachlichen Dokumentation sollten Sie sich auch hier auf das absolut notwendige Mindestmaß beschränken, damit bei Änderungen nicht zu viel Aufwand für das Anpassen der Dokumentation erforderlich ist.

Viele Hinweise und Ideen, wie viel Modellierung und Dokumentation zu welchem Zeitpunkt durchgeführt werden kann, hat auch Scott Ambler auf seiner Website zur agilen Modellierung zusammengetragen (vgl. [Amb10]).

Je besser das Design der Software sowie die Qualität und die Kommentierung des Codes sind, umso kürzer kann die Menge an benötigter technischer Dokumentation außerhalb ausfallen. Auch fehlerfreier Code unterstützt die Agilität, da so das Team nicht mit der aufwändigen und riskanten Behebung von Fehlern beschäftigt ist. Stattdessen kann es sich um die Entwicklung neuer Funktionalität kümmern. Zwei Techniken helfen Ihnen dabei, über die gesamte Projektlaufzeit ein hohes Qualitätsniveau zu halten:

- *Regelmäßige Code- und Architektur-Reviews* sind ein wichtiger, in der Breite noch viel zu wenig genutzter Beitrag zur Qualität. Eine Möglichkeit zum regelmäßigen Code-Review ist beispielsweise das mit XP eingeführte *Pair Programming*. Darüber hinaus gibt es auch leistungsstarke Tools, die in den kontinuierlichen Build-Prozess einge-

bunden werden und mit Hilfe von statischen Analysen auf Probleme in der Programmierung (z. B. „Sonar", vgl. [Son]) und in der Einhaltung der Architektur (z. B. „Structure101", „SonarJ") hinweisen.

- *Laufendes Refactoring* gewährleistet, dass die Struktur des Codes in jeder Stufe in einem guten Zustand ist, damit die effiziente Weiterentwicklung nicht durch Degeneration gefährdet wird.

Generell ist ein durchgängiges Qualitäts- und Testmanagement auf allen Ebenen – von der Entwicklung bis zur Abnahme – unabdingbar. Der Umfang und die Formalität der Maßnahmen hängen dabei stark von der Größe und Kritikalität des Projekts ab.

- Ein gutes Mittel zur Definition der erwarteten Qualität sind so genannte *Definitions of Done (DoD)*. In diesen wird für jedes Arbeitsergebnis, typischerweise eine User-Story, festgelegt, wann der Status „fertig" erreicht ist. Aber auch für technische Ergebnisse ist eine DoD sinnvoll: Für ein Deployment könnte das beispielsweise das Packen des Systems in einer mit dem Auftraggeber vereinbarten Form, die erfolgreiche Durchführung von Integrationstests sowie die Aktualisierung von Release-Notes und Installationsanleitung sein.
- Für das Testmanagement sind Unit-Tests in der Entwicklung, Integrationstests für größere Einheiten und ein Abnahmetest des Kunden für produktive Stufen das Mindestmaß. Für Systeme mit hoher Last sind Last- und Performancetests auch dringend angeraten. Streben Sie für alle Teststufen einen hohen Automatisierungsgrad an, um die häufigen Releases effizient testen zu können. Eine gute Vorgehensweise für automatisierte Akzeptanztests, im Zusammenspiel mit Akzeptanzkriterien zur fachlichen Spezifikation, hat beispielsweise Johannes Link in seinem Artikel über agile Akzeptanztests beschrieben (vgl. [Lin09]).

Enge Zusammenarbeit für hohe Effizienz

- Business people and developers work together daily throughout the project.
- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

Auch die umfangreichste Konzeption wird nicht helfen, wenn die Fachleute des Auftraggebers und die Entwickler des Auftragnehmers kein gemeinsames Verständnis von der Aufgabe entwickeln. Je direkter der Kontakt ist, umso weniger Informationen gehen in der Kommunikation verloren. Der Idealfall ist eine *direkte Face-to-Face Abstimmung mit allen Beteiligten und Stakeholdern*.

Neben einer engen Kommunikation im gemeinsamen Team von Auftraggeber und -nehmer gibt es ein weiteres wichtiges Strukturelement für die Konsistenz und Vollständigkeit der Lösung. Aus der Erfahrung gerade aus mittleren und großen Projekten hat sich die Implementierung der *Rollen eines fachlichen und eines technischen Chefdesigners* bewährt. In großen Projekten oder Programmen können das durchaus auch Vollzeit-Rollen sein. In kleineren oder mittleren Projekten hingegen werden die Rollen häufig in Personalunion mit anderen Aufgaben ausgefüllt, beispielsweise durch den erfahrensten Entwickler des Teams. Beide Chefdesigner müssen eine große Übersicht über ihren jeweiligen Themenbereich behalten, um dem Team helfen zu können, Probleme frühzeitig zu erkennen und ihre Lösung herbeizuführen. Die Arbeitsweise der Chefdesigner wird stark durch Moderation geprägt. Sie geben durch ihre Erfahrung Leitlinien vor und helfen dem Team, ein gutes Design zu finden, das die aktuellen Anforderungen erfüllt und für zukünftige Änderungen flexibel ist.

Ein weiteres wichtiges Element, um eine enge Kommunikation zu erreichen, sind Zeremonien, die nicht umsonst in allen agilen Verfahren eine große Rolle spielen. *Regelmäßige Meetings zu inhaltlichen Themen* mit festgelegten Terminen und einem definierten Teilnehmerkreis können sehr gut dabei helfen, dass die nötigen Abstimmungen nicht dem Zeitdruck und vollen Terminkalendern zum Opfer fallen. Dabei gilt es in unserem Kontext, Folgendes zu beachten: Im Projekt arbeiten nicht nur ein enger Kreis von Mitarbeitern des Fachbereichs und Entwickler zusammen. Bereits im Vorfeld sind viele Abstimmungen mit weiteren Stakeholdern beim Kunden nötig, die häufig nur sehr punktuell ins Projekt eingebunden sind. Auch im weiteren Projektverlauf und nach der Auslieferung an den Kunden wird für viele verschiedene Nutzerkreise fachliche und technische Dokumentation benötigt (siehe Abbildung 3). Das richtige Maß an Schriftlichkeit unterstützt die Effizienz in der Kommunikation, da Entscheidungen im fachlichen und technischen Design im Gesamtkontext dargestellt werden. Außerdem können so Abstimmungen mit den einzelnen Stakeholdern besser vorbereitet werden.

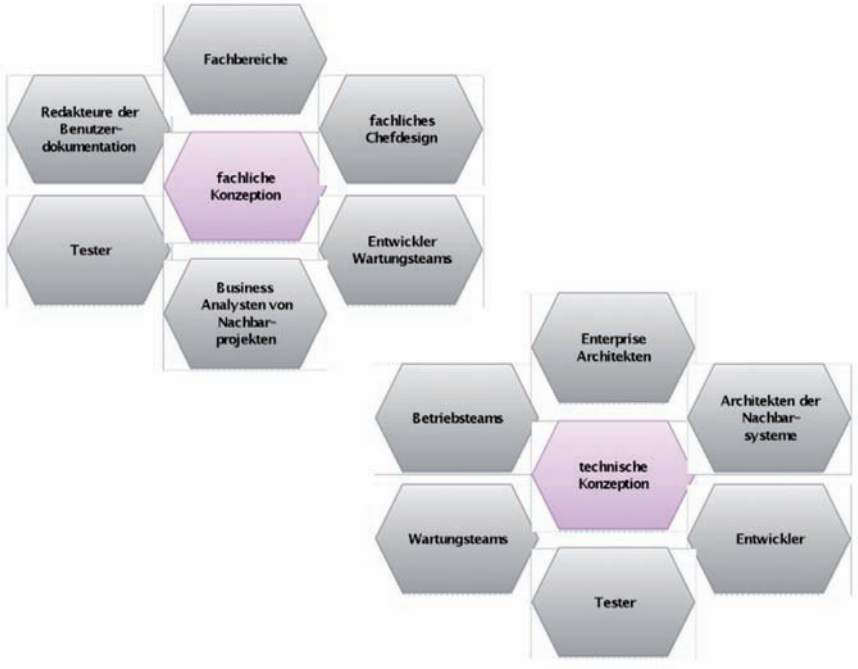

Abb. 3: Nutzerkreise für fachliche und technische Dokumentation.

Motivierte Teams erzielen bessere Ergebnisse

- Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
- The best architectures, requirements and designs emerge from self-organizing teams.
- Agile processes promote sustainable development. The sponsors, developers and users should be able to maintain a constant pace indefinitely.

Eigentlich ist es eine Selbstverständlichkeit, nichtsdestotrotz häufig aber doch ein Quell stetigen Ärgernisses: Letztendlich werden Softwareprojekte von Menschen durchgeführt und je motivierter diese sind, desto bessere Ergebnisse kann man von ihnen erwarten. In der Praxis sieht das aber oft anders aus und die Teammitglieder vergeuden unnötig Energie, indem sie sich mit vermeidbaren Problemen beschäftigen. Was können Sie tun, damit die Mitarbeiter sich auf ihre Arbeit konzentrieren können und wollen?

- Stellen Sie dem Team *angemessene Arbeitsbedingungen* zur Verfügung. Das fängt bei Räumlichkeiten an, in denen sich Ihre Mitarbeiter wohl fühlen können. Sie sollten so zusammenhängend sein, dass die direkte Kommunikation ohne Behinderungen gewährleistet ist. Auch die Ausstattung des Teams mit den notwendigen Ressourcen (Arbeitsplatzrechner, Server, Software-Tools) gehört dazu. Binden Sie daher das Team in die Entscheidungen hinsichtlich der Auswahl und Gestaltung seines Umfelds weitestmöglich mit ein.
- *Das Team führt gemeinsam die Stufenplanung und die Verteilung der Aufgaben durch*. Je mehr Sie die Mitarbeiter an Entscheidungen im Projekt beteiligen, desto mehr geben Sie ihnen die Möglichkeit, sich den gemeinsam definierten Zielen zu verpflichten. Außerdem wird sich die inhaltliche Qualität gegenüber der Planung durch einen Projektleiter verbessern, da das Wissen aller Beteiligten direkt darin einfließen kann.

Typischerweise erwartet der Kunde auf Ihrer Seite einen Projektleiter, der das Projekt ihm gegenüber vertritt. Auch innerhalb ihrer Firma wird es jemanden geben müssen, der Aufgaben wie die Vorbereitung der Rechnungsstellung oder Reporting übernimmt. Dafür ist aber kein klassischer Projektleiter nötig, die Rollenbeschreibung ist am besten als „Außenminister" des Projekts definiert. In Scrum kann diese Rolle beispielsweise der *Product Owner (PO)* übernehmen.

Eine exzellente weiterführende Literatur ist das Buch „Wien wartet auf dich!", in dem Tom DeMarco und Timothy R. Lister bereits 1987 den Begriff „Peopleware" geprägt haben. Der Begriff soll ausdrücken, dass neben Hard- und Software der Mensch ein entscheidender Faktor in der Softwareentwicklung ist. Ihr Buch ist auch heute noch lesenswert, um etwas über die Motivation und Führung von Mitarbeitern in Softwareprojekten zu lernen (vgl. [DeM99]).

Ein hehres Ziel, das in klassischen Projektmodellen jedoch nur extrem selten erreicht wird, ist eine kontinuierliche Geschwindigkeit, die das Team über die gesamte Projektlaufzeit beibehalten kann. Typischerweise steigt die Kurve der Arbeitslast zu Auslieferungsterminen von Konzepten oder Software stark an. Das hat zur Folge, dass die Qualität in diesen Zeiträumen eher sinkt. Zudem sind die Teammitglieder ausgebrannt und benötigen zu Beginn der nächsten Phase erst einmal Erholung. Das bedeutet, dass nur mit geringerer Last gestartet werden kann – ein Teufelskreis, da für die nächste Phase Zeit verloren geht und zusätzlich auch noch die Fehler der Vorphase ausgebügelt werden müssen. Auch hier stellt sich wieder die Frage nach den Mitteln, mit denen Sie eine höhere Kontinuität erreichen können.

Eine Antwort habe ich bereits zu Beginn dieses Artikels diskutiert. Indem Sie den Zyklus der Stufen möglichst kurz halten, bleibt die Kurve der Arbeitslast geringer, weil das Team nicht mehr über einen längeren Zeitraum auf ein großes Ziel hinarbeitet. Als zweites Mittel können Sie eine *laufende Verfolgung des Projektfortschritts auf Basis eng getakteter Aufgaben einführen*. Die Aufgaben müssen so geschnitten sein, dass sie mindestens innerhalb einer Stufe komplett abzuschließen sind. Idealerweise sind die User-Storys bereits so fein-granular, dass eine Story in wenigen Tagen erledigt werden kann. Wenn das nicht sinnvoll möglich ist, sollte das Team Unteraufgaben definieren. Mit den granularen User-Storys haben Sie sowohl die Einheiten, in denen die Projektplanung pro Stufe stattfindet, als auch die Einheiten, in denen Sie den Projektfortschritt verfolgen (mit den Status „geplant", „in Arbeit" und „abgeschlossen"). Das entspricht in Scrum dem *Sprint Backlog*. Dieses Mittel ist aber nicht auf Scrum beschränkt, sondern kann auch in einem mehr konventionellen Projektmodell gewinnbringend eingesetzt werden. Machen Sie den Projektfortschritt für alle Beteiligten transparent! Dies kann etwa durch den Einsatz eines physikalischen Taskboards erfolgen oder auch durch Softwarewerkzeuge, die dies sehr gut unterstützen.

Effizienzsteigerung durch regelmäßige Reflektion

- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Ein wohl definiertes und konsequent gelebtes Projektmodell ist ein wichtiger Faktor für den Projekterfolg. Genauso konsequent müssen Sie aber beobachten, ob Sie noch auf dem richtigen Weg sind, und Ihre Vorgehensweise bei Bedarf justieren. Dazu führt das Team *regelmäßige Retrospektiven für den Prozess und die Rahmenbedingungen* durch. Mindestens einmal pro Monat – sinnvollerweise synchronisiert mit der Auslieferung der Stufen – sollte sich das Team zusammen mit den Projektverantwortlichen Zeit nehmen und sich folgende Fragen stellen:

- *Ist unsere Vorgehensweise noch optimal?* Alle Vorgehensweisen, die in diesem Artikel angesprochen wurden, werden auf den Prüfstand gestellt und die Punkte angepasst, die den Projekterfolg nicht mehr optimal unterstützen.
- *Gibt es Änderungen an den Rahmenbedingungen, auf die wir reagieren sollten?* Dazu gehört beispielsweise das Risikomanagement, aber auch das Beobachten von Chancen, wo der Kunde noch besser unterstützt werden kann. Ein weiteres Beispiel wären Änderungen in der Organisation oder der Vorgehensweisen des Kunden, die häufig Einfluss auf das aktuelle Projekt haben können.

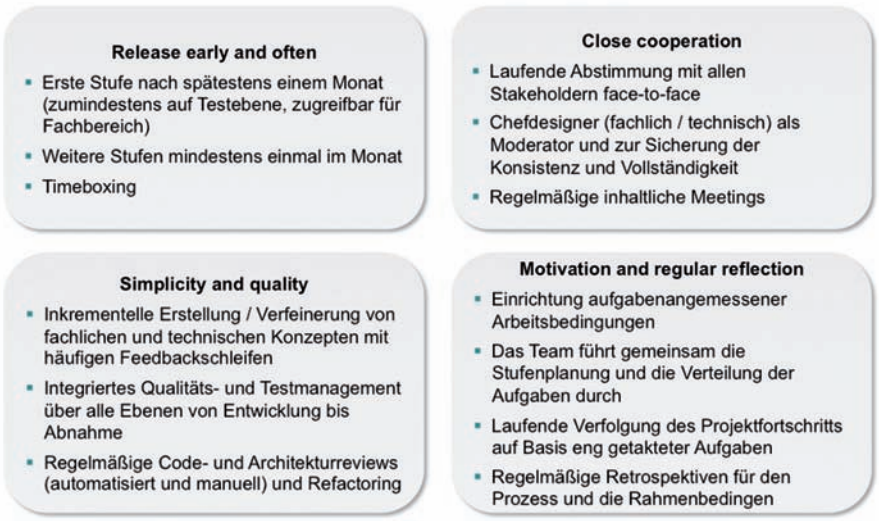

Abb. 4: Checkliste für ein modernes Projektmodell.

Fazit

Es gibt viele verschiedene Möglichkeiten, um ein Projekt durchzuführen. Im Projektgeschäft stehen wir häufig vor der zusätzlichen Herausforderung, dass der Auftragnehmer das Vorgehensmodell nicht alleine bestimmen kann. Stattdessen hat der Auftraggeber mehr oder weniger genaue Vorgaben, wie vorgegangen werden muss und welche Ergebnisse er zu welchem Zeitpunkt erwartet. Auch wenn die Vorteile der agilen Modelle sich immer weiter verbreiten, lassen es die Umstände im konkreten Projekt nur selten zu, ein wirklich reines agiles Vorgehen wie z. B. Scrum umzusetzen.

Die zwölf Prinzipien, die dem Agilen Manifest zu Grunde liegen, liefern eine wertvolle Hilfestellung dahingehend, welche Rahmenbedingungen ein modernes Projektmodell erfüllen muss, um einen optimalen Projekterfolg zu gewährleisten. Diese Rahmenbedingungen werden Sie zu großen Teilen auch im Tailoring einer vorgegebenen Vorgehensweise umsetzen können.

Zusammengefasst lassen sich diese Rahmenbedingungen in vier Blöcken darstellen, die als konkrete Checkliste für das Aufsetzen der Vorgehensweise im Projekt dienen können (siehe Abbildung 4). ■

Literatur & Links

[Agi] Principles behind the Agile Manifesto, siehe http://agilemanifesto.org/principles.html

[Amb10] S.W. Ambler, Agile Modeling Homepage, 2010, siehe http://www.agilemodeling.com/

[DeM99] T. DeMarco, T. Lister, Wien wartet auf Dich!, Hanser 1999

[Hru10] P. Hruschka, G. Starke, arc42 – Ressourcen für Software-Architekten, 2010, siehe http://www.arc42.de

[Lin09] J. Link, Agile Akzeptanztest: Vision, Praxis und Werkzeuge, in: OBJEKTspektrum 5/2009

[Pat09] J. Patton, The new user story backlog is a map, 2009, siehe http://www.agileproductdesign.com/blog/the_new_backlog.html

[Roo10] S. Roock, KANBAN in der Softwareentwicklung: Schlanke Softwareentwicklung im Fluss, in: OBJEKTspektrum 2/2010

[Scr09] Srum.org, The Scrum Guide, 2009, siehe http://www.scrum.org/scrumguides/

[Son] Sonar Homepage, siehe http://www.sonarsource.org/