VON AGILEN VERFAHREN LERNEN: AUF DEM WEG ZU BEDARFSGERECHTER DOKUMENTATION

In vielen IT-Projekten ist Dokumentation ein leidiges Thema. In manchen Projekten ist Dokumentation nahezu ein Fremdwort, andere ersticken geradezu in Papierbergen, die dann doch nicht gelesen werden. Ein Blick auf die Prinzipien agiler Entwicklung kann helfen, bei der Projektdokumentation planvoll vorzugehen und das richtige Maß zu finden.

der autor

Dr. Andreas Rüping
(E-Mail: andreas.rueping@rueping.info)
ist freiberuflicher IT-Berater mit dem Schwerpunkt Web-Applikationen. Er favorisiert ein agiles Vorgehen im Projektalltag und unterstützt Unternehmen bei der Durchführung agiler Entwicklungsprojekte.

Dokumentation gilt in der Softwareentwicklung traditionell als ungeliebtes Thema. Sie wird oft eingefordert, aber bei weitem nicht immer in der gewünschten Form bereitgestellt. Viele Entwickler haben eine ausgeprägte Abneigung gegen die Erstellung von Dokumentation, andererseits sind die Dokumente, die entstehen, häufig unhandlich oder gar unbrauchbar.

Eine Ursache für dieses Dilemma ist, dass es in vielen Projekten keine ausgewogene Strategie zur Erstellung der Projektdokumentation gibt. Stattdessen wird oft – vermutlich unbewusst – eine von zwei extremen Strategien verfolgt, die einander diametral entgegengesetzt und gleichermaßen ungeeignet sind:

- Vor allem Projekte, die dem klassischen Wasserfall-Modell folgen, arbeiten häufig dokumentenorientiert, d. h. sie versuchen jegliche Information, die in einer bestimmten Projektphase benötigt wird, vorher schriftlich zu fixieren. Dabei entsteht in der Regel eine große Menge an Dokumentation, die aber in ihrer Fülle gar nicht gelesen wird und außerdem schneller veraltet, als es jedem Projektverantwortlichen recht sein kann. Insider sprechen gelegentlich von *Write-Only-Dokumentation*; Tom DeMarco und Timothy Lister schreiben in ihrem Buch Peopleware dazu: „Voluminous documentation is part of the problem, not part of the solution" (vgl. [DeM99]).
- Weniger traditionell durchgeführte Projekte (manchmal auch solche, die sich agil nennen) legen nur geringen Wert auf Dokumentation und behandeln diese demzufolge stiefmütterlich. In manchen Projekten wird nur rudimentär oder überhaupt nicht dokumentiert. Die Problematik dieses Ansatzes wird spätestens dann deutlich, wenn Wissensträger das Projekt verlassen haben, die Funktionsweise der Software nicht mehr nachvollziehbar ist und ihre Wartung oder Weiterentwicklung nahezu unmöglich wird, ohne ein umfangreiches Reverse-Engineering zu betreiben, was letztlich eine aufwändige und wenig dankbare Angelegenheit ist.

Beide Ansätze führen offensichtlich nicht zum Ziel. Wie kann also ein konstruktiver Ansatz zur Lösung des Problems aussehen? Eine vielversprechende Möglichkeit besteht darin, Strategien und Prinzipien aufzugreifen, die von agilen Methoden für IT-Projekte ins Feld geführt wurden. Sich an agilen Verfahren zu orientieren, ist aus zwei Gründen sinnvoll.

Zum einen bieten agile Verfahren dadurch einen Mehrwert, dass sie immer wieder den Sinn und Zweck dessen hinterfragen, was in einem Projekt getan wird. Im Agilen Manifest (vgl. [Cun01]) werden Dinge, die in der Praxis der Softwareentwicklung besonders wertvoll sind, anderen Dingen gegenübergestellt, die als weniger wertvoll angesehen werden (vgl. auch Artikel von Uwe Friedrichsen auf Seite 22 dieser Ausgabe von OBJEKTspektrum). Wie **Abbildung 1** verdeutlicht, wird Dokumentation eher kritisch betrachtet: Sie steht auf der rechten Seite des Agilen Manifests, wird also zu den Dingen gerechnet, die eher ein Mittel zum Zweck als ein eigentliches Ziel darstellen. Dahinter verbirgt sich die Erkenntnis, dass Dokumentation einerseits und die Weitergabe von Wissen andererseits zwei Paar Schuhe sind: Projektwissen steckt häufig in den Köpfen der Beteiligten und bei der Übergabe von Dokumenten kann in der Regel nur ein Teil des vorhandenen Wissens weitergereicht werden.

Allerdings lehnen agile Verfahren eine angemessene Dokumentation durchaus nicht ab. Wenn gelegentlich behauptet wird, agile Methoden versuchten generell ohne Dokumentation auszukommen, so stimmt das nicht. Agile Verfahren betrachten Dokumentation aber eben nicht als Selbstzweck, sondern als etwas, das durch einen bestimmten und nachvollziehbaren Nutzen gerechtfertigt sein muss.

Der zweite Grund, weshalb sich ein Blick auf die agilen Verfahren lohnt, ist die zunehmende Verbreitung dieser Verfahren in der Praxis der Softwareentwicklung.

Besonders wertvoll	Auch wertvoll, aber nicht in dem Ausmaß
• Individuen und Interaktionen	• Prozesse und Werkzeuge
• Funktionierende Software	• Umfassende Dokumentation
• Zusammenarbeit mit dem Kunden	• Vertragsverhandlung
• Reagieren auf Veränderung	• Befolgen eins Plans

Abb. 1: *Rolle der Dokumentation im Agilen Manifest.*

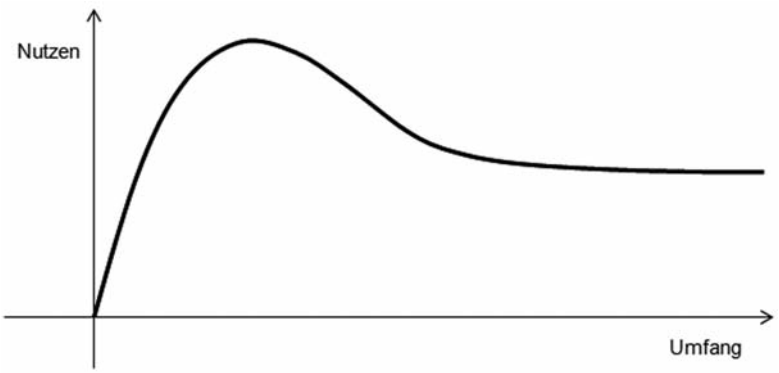

Abb. 2: Fraglicher Nutzen umfangreicher Dokumentation.

Den Anfang hat vor einigen Jahren das *eXtreme Programming (XP)* (vgl. [Bec00]) gemacht, auch *Crystal Clear* (vgl. [Coc05]) hat einen gewissen Bekanntheitsgrad erreicht und heute setzen mehr und mehr Projekte erfolgreich vor allem *Scrum* ein (vgl. [Sch08], [Pic07], [Wir09]).

Mit der zunehmenden Verbreitung agiler Verfahren nimmt automatisch der Druck zu, agile Prinzipien auch auf die Erstellung der Dokumentation anzuwenden. Wird zum Beispiel ein Projekt nach Scrum durchgeführt, ist es nur folgerichtig, wenn die Spielregeln von Scrum auch für die Erstellung der Dokumentation gelten. Mit zunehmender Popularität agiler Verfahren wird daher auch das Thema „agile Dokumentation" immer wichtiger.

Umfang der Dokumentation

Einen ersten Ansatzpunkt für Agilität bildet der Umfang der Dokumentation. Abbildung 2 verdeutlicht eine Erfahrung, die Projektteams immer wieder machen: Mehr ist nicht automatisch besser. Der Gedanke, möglichst alles aufzuschreiben, führt in schöner Regelmäßigkeit in die Irre, weil er die Frage außer Acht lässt, welche Dinge besser in schriftlicher Form festgehalten werden und für welche Dinge die verbale Kommunikation im Team das effektivere Mittel darstellt. In Tabelle 1 habe ich die Vorteile beider Kommunikationsformen einander gegenübergestellt.

Schriftliche Dokumentation ist insbesondere dann sinnvoll, wenn Informationen langfristig verfügbar sein müssen, wenn der Leserkreis räumlich verteilt ist oder wenn mit dem Aufschreiben ein tatsächlicher Erkenntnisgewinn verbunden ist. Für die Projektdokumentation bedeutet dies konkret Folgendes:

- Anforderungen an die Software sollten in dem Maße aufgeschrieben werden, in dem eine schriftliche Darstellung dem Entwicklungsteam den Einstieg in die nachfolgenden Aufgaben erleichtert, also in Design, Implementierung und Test. Scrum sieht hierfür User-Storys vor, in denen in prägnanter Form beschrieben wird, was die Software leisten soll. Für viele Projekte ist das völlig ausreichend – speziell dann, wenn schnelles Feedback dadurch garantiert ist, dass der Fachbereich, der die Anforderungen formuliert, in der Rolle des *Product Owner* direkt ins Projekt eingebunden ist.
- Der Überblick über die Softwarearchitektur sowie wichtige Designaspekte gehören in die Kategorie der Dinge, die schriftlich festgehalten werden sollten, weil sie für die Wartung oder die Weiterentwicklung der Software relevant sind – jedenfalls dann, wenn man davon ausgehen muss, dass zum Zeitpunkt der Weiterentwicklung das ursprüngliche Team in alle Winde zerstreut sein kann (was häufig der Fall ist). Dabei ist es in den allermeisten Fällen unnötig, die Designdokumentation auf die Implementierungsebene auszudehnen: Wenn die Software verändert wird, bleibt den Entwicklern ohnehin nichts anderes übrig, als sich mit dem Code auseinanderzusetzen. Hingegen ist es häufig sinnvoll, die Begründung für das gewählte Design zu dokumentieren. In der Regel ist diese allein anhand des Codes nicht nachvollziehbar, für die langfristige Weiterentwicklung eines Systems spielt sie aber eine wichtige Rolle (vgl. [Rüp03]).
- Die wenigsten Projekte spielen sich auf der grünen Wiese ab. Fast immer gibt es Nachbarprojekte, zu denen Schnittstellen bestehen. Manchmal stehen die Teams der benachbarten Projekte für Diskussionen unmittelbar zur Verfügung, manchmal aber auch nicht. Schnittstellenbeschreibungen sowie weitere projektübergreifende Aspekte sind daher Kandidaten für eine schriftliche Dokumentation.
- In vielen Fällen sind Dokumente zur Nutzung und zum Betrieb der Software erforderlich. Solche Dokumente richten sich an einen Leserkreis außerhalb des Teams und profitieren davon, wenn sie in der Sprache der Leser, die in der Regel keine Softwareentwickler sind, formuliert sind.

Direkte Kommunikation	Dokumentation
Interaktion • persönlicher Umgang miteinander • informeller Informationsfluss	Skalierbarkeit • Nutzung durch viele Personen • Nutzung durch räumlich verteilte Teams
Einschluss von non-verbaler Kommunikation • Körpersprache • Gestik	Schriftliche Ausdrucksmöglichkeit • Vorteil für introvertierte Personen • Erkenntnisgewinn beim Schreiben
Schnelle Verfügbarkeit • kurze Frage-Antwort-Zyklen	Langfristige Verfügbarkeit • Verfügbarkeit nach Ablauf eines Projekts

Tabelle 1: Vorteile direkter Kommunikation vs. Vorteile von Dokumentation.

Wie diese Aufzählung zeigt, gibt es eine Reihe von Projektdokumenten, die durchaus ihren Sinn und Zweck haben. Das bedeutet aber nicht, dass die Dokumente immer auch sehr umfangreich sein müssen. Eine prägnante Beschreibung hilft oft mehr als ein seitenlanger Papierwust. Welcher Detaillierungsgrad letztlich als geeignet

anzusehen ist, hängt natürlich vom einzelnen Projekt ab. Sehr große Projekte sowie Projekte mit hoher Kritikalität (zum Beispiel in der Medizintechnik) erfordern mehr Dokumentation, die meisten Projekte kommen hingegen sehr gut mit relativ wenig Dokumentation aus. In keinem Fall sollte die Dokumentation zum Selbstzweck verkommen: Ihr Umfang und ihr Detaillierungsgrad müssen sich am tatsächlichen Bedarf orientieren.

Zeitpunkt der Erstellung

Ein weiterer Ansatzpunkt hin zu einer „agileren" Dokumentation betrifft den Zeitpunkt der Erstellung. Während das Wasserfallmodell davon ausgeht, dass zu Beginn eines Projekts eine vollständige Spezifikation und vor Beginn der Implementierung eine vollständige Designbeschreibung vorliegen, empfehlen agile Verfahren, Dokumentation so spät wie möglich (aber nicht später) zu erstellen. Die Idee hinter dieser Empfehlung ist klar: Man muss davon ausgehen, dass sich sowohl die Spezifikation als auch das Design im Laufe eines Entwicklungsprojekts noch ändern, denn es ist illusorisch anzunehmen, ein Projekt könne ohne derartige Änderungen auskommen. Je später man dokumentiert, um so seltener muss man daher die Dokumentation aktualisieren.

Aus der Perspektive traditioneller Projekte erscheint die agile Vorgehensweise oft ungewöhnlich. Insbesondere das Fehlen einer vollständigen Spezifikation zu Beginn des Projekts wird gelegentlich als Risiko betrachtet. Kann so etwas gut gehen? Die Erfahrung zeigt: Es kann.

Zunächst einmal schützt auch eine umfassende Spezifikation nicht vor Überraschungen. Der Glaube, ein komplexes Softwaresystem a priori spezifizieren zu können, hat sich in der Praxis allzu oft als Illusion herausgestellt. Viele – insbesondere große – Projekte haben mit einer langen Spezifikationsphase begonnen und sind letztlich doch bei etwas ganz anderem angekommen, als ursprünglich geplant war.

Außerdem bedeutet das Fehlen einer vollständigen Spezifikation nicht, dass es keine Vorstellung von den Projektzielen gibt. Auch in einem agilen Projekt sollten die Stakeholder von Beginn an eine Perspektive für das Projekt entwickeln: die Produktvision. Es ist aber völlig legitim und natürlich, wenn sich diese Perspektive im Laufe der Zeit weiterentwickelt – und genau das wird von agilen Verfahren anerkannt.

Letztlich ist es ehrlicher, ein Projekt mit einer überzeugenden Perspektive, aber einer unvollständigen Spezifikation zu beginnen, als sich der Illusion hinzugeben, sämtliche Anforderungen von vornherein formulieren zu können. Voraussetzung für den Erfolg des agilen Vorgehens ist aber die enge Kooperation zwischen IT und Fachbereich. Nur wenn der Fachbereich ins Projektgeschehen eingebunden ist (wie es z. B. Scrum durch die Rolle des *Product Owner* garantiert), ist auch ein agiler Ansatz bei der Dokumentation von Anforderungen Erfolg versprechend.

Kasten 1: *Projektplanung im Kontext eines agilen Dokumentationsansatzes.*

Wie bestimmt man den spätest möglichen Zeitpunkt? Zum einen muss die Dokumentation natürlich fertiggestellt werden, bevor sie benötigt wird. Zum anderen muss die Dokumentation entstehen, solange die Wissensträger im Projekt noch verfügbar sind. Konkret bedeutet das:

- Spezifikationen müssen, soweit sie erforderlich sind, im benötigten Detaillierungsgrad vor der Implementierung zur Verfügung stehen. Das kann und sollte aber iterativ geschehen: Zu jedem Zeitpunkt müssen nur Spezifikationen vorliegen, die für den anstehenden

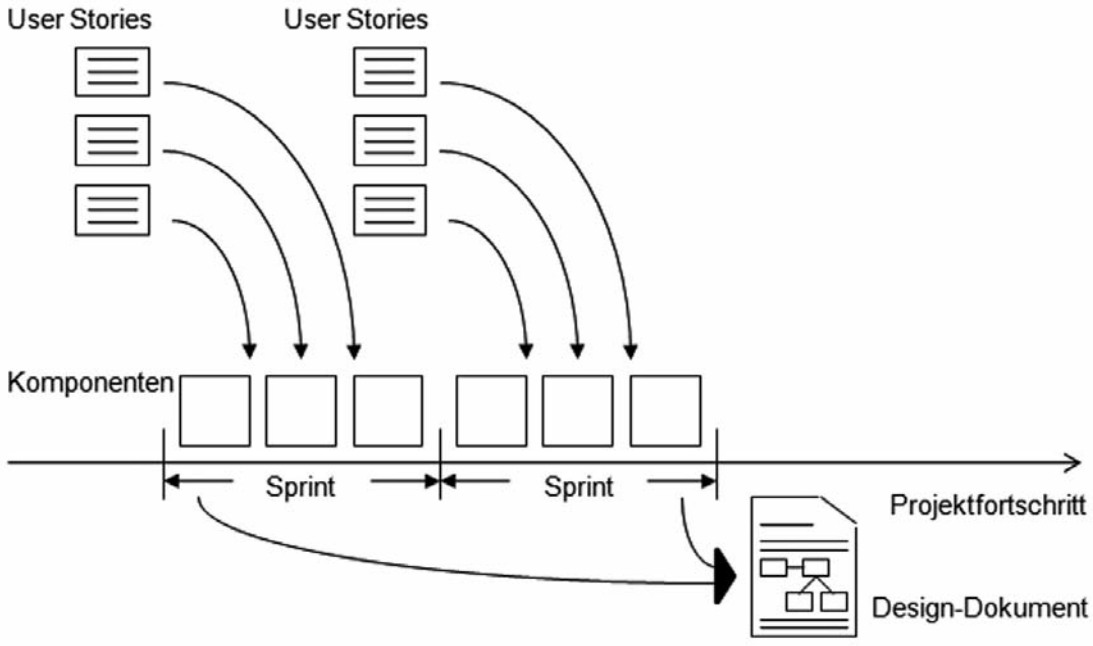

Abb. 3: *Erstellung der User-Storys vor dem Sprint, Erstellung der Designdokumentation im Nachgang.*

Wenn es darum geht, die Erstellung von Dokumentation zu vereinfachen, werden häufig Generierungstechniken ins Gespräch gebracht. Der Wunsch ist verständlich, suggerieren Generierungstechniken doch die Möglichkeit, Dokumentation ohne großen Aufwand herzustellen und automatisch aktuell zu halten.
Das Agile Manifest relativiert die Bedeutung von Werkzeugen (siehe Abbildung 1): Die Erfahrung zeigt, dass sich mit Werkzeugen selten Wunder vollbringen lassen und dass die Bedeutung von Werkzeugen für den Projekterfolg gelegentlich überschätzt wird. Trotzdem sind Werkzeuge (z. B. ein Wiki) natürlich willkommen, wenn sie die Erstellung der Dokumentation einfacher und geradliniger machen. Das gilt auch für Generierungstechniken. Zum Beispiel ist der Einsatz von „JavaDoc" zur Erzeugung der Schnittstellendokumentation in Java-Projekten seit Jahren anerkannter Standard und natürlich ist ein solches Vorgehen auch aus Sicht agiler Verfahren sinnvoll. Gleiches gilt für die Generierung von PDF-Dokumenten aus Wikis heraus. Dennoch ist der kritische Blick auf die Rolle von Werkzeugen berechtigt: Sichten, Aggregate und Formate mögen sich generieren lassen, Dokumentationsinhalte hingegen nicht.
Kern der Dokumentationserstellung ist immer die Frage, welches Wissen schriftlich festgehalten werden soll, sowie die geeignete und prägnante Aufbereitung dieses Wissens. Auf den Inhalt kommt es an. Die Kreativität, die an dieser Stelle gefordert ist, lässt sich nicht durch Generierungstechniken ersetzen.

Kasten 2: *Auf der Suche nach einem effizienten Dokumentationsprozess: Schreiben vs. Generierung.*

Entwicklungsschritt relevant sind. Häufig werden solche Spezifikation in Form ablauffähiger Akzeptanztests aufgeschrieben (vgl. [Lin09]).

- Architektur- und Designdokumente können häufig erst im Laufe eines Projekts entstehen. Zumindest in kleineren Teams, bei denen das Design der Software in den Köpfen oder am Whiteboard entsteht, sind solche Dokumente während der Realisierungsphase nicht erforderlich. Bevor das Team sich auflöst, müssen die notwendigen Dokumente natürlich geschrieben werden. Um in regelmäßigen Abständen aktuelle Designdokumente zu erhalten, hat es sich als sinnvoll erwiesen, ihre Auslieferung an die Auslieferung von Software-Releases zu koppeln. Die Erstellung der Dokumentation macht dann natürlich Arbeit, glücklicherweise aber nur einmal pro Release und nicht permanent.

Scrum trägt dieser Denkweise Rechnung: Es ist geprägt durch ein iteratives Vorgehen, bei dem die gesamte Projektlaufzeit in einzelne Sprints aufgeteilt wird: in Abschnitte von wenigen Wochen, in denen jeweils ein Teil der Gesamtfunktionalität implementiert wird. Wie in Abbildung 3 angedeutet, werden Anforderungen vom *Product Owner* in Form von User-Storys zu Beginn eines Sprints zur Verfügung gestellt. Im nächsten Sprint folgt dann der nächste Satz an User-Storys. Es wird immer nur so viel an Spezifikation bereitgestellt, wie das Team benötigt, um im nächsten Sprint die gewünschte Funktionalität bereitzustellen. Wie detailliert diese Information vorliegen muss, hängt von Know-how und der Zusammenarbeit zwischen Team und *Product Owner* ab: Je besser die funktioniert, umso weniger muss zuvor festgelegt werden (siehe auch Kasten 1). Natürlich existiert auch in einem Scrum-Projekt schon zu Beginn eine Vorstellung davon, wohin die Reise in Summe gehen soll, aber auf den Versuch, die Details von vornherein festzulegen, wird bewusst verzichtet. User-Storys beziehen sich immer auf den aktuellen Stand der Dinge, es gibt keine Spezifikationen, die sich schon erledigt haben, bevor ihre Umsetzung begonnen wird. Unnötiger Aufwand wird vermieden.

Designdokumente können natürlich in einem Scrum-Projekt auch entstehen, insbesondere dann, wenn sie für die langfristige Wartbarkeit der Software erforderlich sind. Üblicherweise erstellt das Team die entsprechende Dokumentation aber erst zu einem Zeitpunkt, zu dem das Design eine gewisse Stabilität erlangt hat (siehe Abbildung 3). Wieder wird unnötiger Aufwand vermieden.

Qualität

In seinem Buch „The Psychology of Computer Programming" schreibt Gerald Weinberg: „The value of documentation is only to be realized if the documentation is well done. If it is poorly done, it will be worse than no documentation at all" (vgl. [Wei98]). Das stimmt natürlich: Wenn es sich lohnt, ein bestimmtes Dokument zu erstellen, weil Leserkreis und Nutzen klar definiert sind (anderenfalls wäre es verzichtbar), lohnt es sich auch, das Dokument in vernünftiger Qualität bereitzustellen.

Qualitativ hochwertige Dokumentation ist nicht mit umfassender Dokumentation zu verwechseln. In vielen Projekten hat sich eine übersichtliche und prägnante Darstellung als vorteilhaft erwiesen. Ziel muss es daher sein, mit vertretbarem Aufwand eine Dokumentation herzustellen, die dem Leser hilft, Informationen schnell und zielsicher nachzuschlagen. Dabei helfen zwei Dinge:

- Zum einen haben sich klar strukturierte Dokumente als hilfreich erwiesen. Eine Reihe von Techniken trägt zum Verständnis bei, z. B. ein vernünftiges (wenn auch schlichtes) Layout, tabellarische Darstellungen sowie ein gesundes Maß an Meta-Informationen (beispielsweise Hinweise für die Leser zum Verständnis eines Dokuments oder auch die Angabe des Releases, auf das sich die Dokumentation bezieht). Tipps zu einer entsprechenden Darstellung finden sich in [Rüp03].
- Zum anderen spielt die Wahl geeigneter Werkzeuge eine Rolle (siehe Kasten 2, speziell im Hinblick auf den Einsatz von Generierungstechniken). Dabei ist zu bedenken, dass traditionelle Dokumente für viele Zwecke nicht das Mittel der Wahl sind. Viele Projekte haben hingegen sehr gute Erfahrungen mit Wikis gemacht (vgl. [Mad08] und den Artikel von Ulrike Parson auf Seite 40 dieser Ausgabe von OBJEKTspektrum).

Für den zweiten Punkt gibt es eine Reihe von Gründen. Zunächst ist für viele Entwickler die psychologische Hürde, ein traditionelles Dokument anzulegen, höher als die, einen Eintrag in einem Wiki zu verfassen. Zweitens bietet ein Wiki auf elegante Weise die Möglichkeit der Zusammenarbeit im Team: Mehrere Projektmitglieder können problemlos gemeinsam an der Dokumentation arbeiten. Drittens erlaubt ein Wiki die Verknüpfung von Informationen. Schließlich bieten Wikis vernünftige Formatierungsmöglichkeiten und können

per PDF-Generierung auch traditionelle Dokumente erzeugen. Wird ein Wiki über die Laufzeit eines Projekts hinweg eingesetzt, entsteht oft eine veritable Ansammlung von Informationen, aus der sich – sofern sie von einem *Wiki-Gardener* gelegentlich in die rechte Form gebracht wird – häufig eine wertvolle Plattform für Wissensmanagement entwickelt.

Ein wichtiger Qualitätsaspekt ist auch die Aktualität der Dokumentation – und auch hier trägt ein Wiki erheblich zur Verbesserung bei. Projektdokumentation veraltet außerordentlich schnell: Projekte sind im Fluss und viele Erkenntnisse reifen erst im Laufe der Zeit. Agile Verfahren betrachten Veränderungen als natürlich und warnen davor, einmal getroffene Entscheidungen um jeden Preis zu zementieren. Von Veränderungen an der Software ist natürlich auch die Dokumentation betroffen. Zwar lassen sich auch traditionelle Dokumente aktualisieren, aber bis die aktuelle Fassung an alle Leser verschickt ist und zur Kenntnis genommen wird, vergeht Zeit. Änderungen am Wiki sind vergleichsweise schnell durchgeführt und stehen sofort zur Verfügung.

Transparenz in der Planung

Auch wenn gute Gründe dafür sprechen, den Umfang der Dokumentation gegenüber dem dokumentenorientierten Ansatz zu reduzieren, gibt es dennoch in praktisch jedem – und eben auch in jedem agilen – IT-Projekt den Bedarf an einem gewissen Maß an Dokumentation. Wenn für diese Dokumentation auch noch eine hohe Qualität eingefordert wird, lässt sich schnell erkennen, dass dies völlig ohne Aufwand nicht zu haben ist.

Um zu verhindern, dass die Scheu vor diesem Aufwand die Erstellung der notwendigen Dokumentation verhindert, hilft nur Transparenz in der Planung. Agile Verfahren zeigen durch ihre konsequente Bewertung und Priorisierung von Anforderungen, wie sich dieses Problem angehen lässt:

- Dokumente sind Artefakte, die im Rahmen eines Projekts entstehen und die genau wie andere Artefakte in die Planung einbezogen sind. Viele Scrum-Teams listen dafür ihre Aufgaben auf dem Taskboard auf und priorisieren diese. Hier dürfen keine Artefakte vergessen werden, die gemäß der „Definition of Done" zu einer produktionsreifen User-Story gehören. Die Erledigung einer Dokumentationsaufgabe lässt sich so genauso einplanen und priorisieren wie andere Aufgaben auch. Wie in Abbildung 4 angedeutet, erscheinen auf dem Scrum-Taskboard letztlich Aufgaben zur Dokumentation neben Aufgaben für Implementierung und Test.
- Durch eine konsequente Planung der Dokumentation wird vermieden, dass die Erstellung wichtiger Dokumente am Ende des Projekts unter den Tisch fällt. Wird Scrum stringent angewendet, entstehen genau die Dokumente, die für den Abschluss der User-Storys erforderlich sind, wobei das Team natürlich die langfristige Wartbarkeit der entwickelten Software im Auge haben muss.

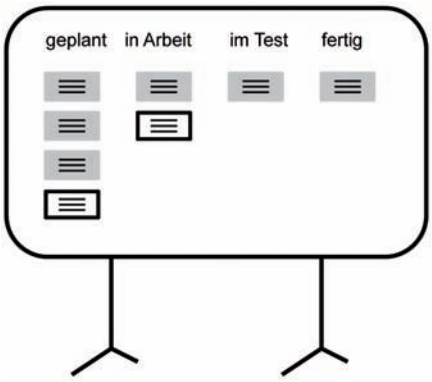

Abbildung 4: Dokumentationsaufgaben (hervorgehoben) zwischen anderen Aufgaben am Taskboard.

Transparenz bezüglich des Aufwands führt ganz automatisch zu der Frage, welche Konsequenzen der Verzicht auf ein Dokument denn hätte. Mit welchen Nachteilen ist zu rechnen, wenn das Dokument nicht oder nur in geringem Detaillierungsgrad angefertigt wird? Falls diese Frage unbeantwortet bleibt oder sich die Nachteile als geringfügig herausstellen, ist der Verzicht auf das Dokument – oder eben die geringere Detailtiefe – offenbar gerechtfertigt. Letztlich hat diese Form der Transparenz den Effekt, dass Dokumentation genau dann erstellt wird, wenn ihr Nutzen größer ist als ihre Kosten. Aus der Gesamtsicht eines Unternehmens ist dieser Effekt zweifellos sinnvoll.

Fazit

Auch Dokumentation kann agil gestaltet werden. Ganz wesentlich dabei ist, dass das Wort „agil" generell keine Technik, sondern eine bestimmte Haltung beschreibt. Auf Dokumentation angewendet, drückt sich diese Haltung darin aus, den Sinn und Zweck möglicher Dokumente zu prüfen, das Vorgehen bei der Erstellung der Dokumentation zu reflektieren und sich letztlich auf die Erstellung solcher Dokumente zu konzentrieren, die ausschlaggebend für den Projekterfolg sind. Dabei ist es völlig legitim, den Erfolg eines Projekts auch über die ursprüngliche Entwicklungszeit hinaus zu messen, d. h. auch Wartung und Weiterentwicklung mit in Betracht zu ziehen. Welche Dokumente im Einzelfall nötig sind und welche Form und welcher Detaillierungsgrad erforderlich sind, hängt vom einzelnen Projekt ab und muss individuell geprüft werden. Profitieren kann die Projektdokumentation von einer agilen Haltung aber in jedem Fall. ■

Literatur & Links

[BecOO] K. Beck, Extreme Programming Explained: Embrace Change, Addison-Wesley 2000

[CocO5] A. Cockburn, Crystal Clear: A Human-Powered Methodology for Small Teams, Addison-Wesley Longman 2005

[CunO1] W. Cunningham et. al., 2001, Manifesto for Agile Software Development, siehe: http://www.agilemanifesto.org/

[DeM99] T. DeMarco, T. Lister, Peopleware: Productive Projects and Teams (2nd Ed.), Dorset House 1999

[LinO9] J. Link, Agile Akzeptanztests: Vision, Praxis und Werkzeuge, in: OBJEKTspektrum 5/2009

[MadO8] S. Mader, wikipatterns, John Wiley & Sons 2008

[PicO7] R. Pichler, Scrum – Agiles Projektmanagement erfolgreich einsetzen. dpunkt.verlag 2007

[RüpO3] A. Rüping, Agile Documentation – A Pattern Guide to Producing Lightweight Documents for Software Projects, John Wiley & Sons 2003

[SchO8] K. Schwaber, M. Beedle, Agile Software Development with Scrum, Pearson Education 2008

[Wei98] G.M. Weinberg, The Psychology of Computer Programming, Dorset House 1998.

[WirO9] R. Wirdemann, Scrum mit User-Stories, Hanser 2009