DARF'S EIN BISSCHEN WENIGER SEIN? ERFOLGREICH DOKUMENTIEREN IN AGILEN ENTWICKLUNGSPROJEKTEN

der autor

Marc Beckers
(E-Mail: marc.beckers@softwareag.com),
Vice President R&D webMethods bei der Software AG, begleitet und gestaltet seit 2009 aus Sicht der Dokumentation, Lokalisierung und Portierung die Umstellung vom Wasserfall-Modell auf agile Entwicklungsprozesse (Scrum) mit.

Agile Prinzipien sind auf die Erstellung von qualitativ hochwertiger Software fokussiert. Folglich bleibt es den technischen Redakteuren und deren Führungskräften häufig selbst überlassen, sich in agilen Entwicklungsumgebungen neu aufzustellen. Angesichts der sich selbst organisierenden Teams liegt darin eine seltene Chance, sich endlich Gehör zu verschaffen. Allerdings müssen hierzu einige Voraussetzungen erfüllt sein.

Agile Entwicklungsmethoden finden zunehmend Verbreitung in der Softwarebranche. Immer mehr IT-Unternehmen bzw. -Abteilungen setzen sie in unterschiedlichen Varianten ein. Dass diese Entwicklung nicht spurlos an der technischen Dokumentation vorbeigeht, zeigt beispielsweise der Umstand, dass auf der Jahrestagung der Gesellschaft für Technische Kommunikation (tekom) in Wiesbaden im November 2010 „Agile Entwicklungsmethoden" ein Hauptthema waren. Sämtliche Veranstaltungen waren überfüllt und von anschließenden regen Diskussionen gefolgt – wobei rege Diskussionen im agilen Umfeld ja in der Natur der Sache liegen.

Klar ist, dass es bei einer Anpassung der Dokumentation an den agilen Entwicklungskontext nicht mit einem Drehen an handwerklichen Stellschrauben getan ist. Der agile Ansatz hat grundlegende Auswirkungen auf die Dokumentationsarbeit an sich und auf das Berufsbild des technischen Redakteurs – und zwar in inhaltlicher, organisatorischer und kultureller Hinsicht. Da das Agile Manifest sowie die öffentlich zugänglichen Diskussionsbeiträge und Erfahrungsberichte doch sehr auf die entbürokratisierte und schnelle Erstellung von robuster Software ausgerichtet sind, gestaltet sich der Umstellungsprozess folglich häufig sehr entwicklungs- und testbezogen. Technische Redakteure fühlen sich in solchen Situationen oftmals an den Rand des Geschehens gedrängt und müssen sich in manchmal hartnäckiger Eigeninitiative einbringen, um am Ende nicht den Zug zu verpassen.

In der weiteren Diskussion ist es wichtig, zwischen den beiden Haupttypen der Dokumentation in Entwicklungsprojekten zu unterscheiden:

- *Projektdokumentation:* Hierzu zählen die internen Unterlagen, die das agile Team benötigt, um seine Arbeit erfolgreich zu verrichten (zum Beispiel Leistungsbeschreibungen, Architekturdiagramme, Spezifikationen, Designpapiere, Use-Cases und Testfälle).
- *Produktdokumentation:* Hierzu gehören die Handbücher und Online-Hilfen, die als Teil des Produkts zusammen mit dem Softwarepaket an den Kunden ausgeliefert werden.

Während im herkömmlichen Wasserfall-Modell der Erfolg der *Produkt*dokumentation sehr stark von Umfang, Qualität und rechtzeitiger Verfügbarkeit der *Projekt*dokumentation abhängt, ist dieser kausale Zusammenhang in agilen Entwicklungsumgebungen so nicht mehr gegeben. Damit haben wir auch schon ein Kernproblem des agilen Ansatzes für die Produktdokumentation identifiziert: Klassische Informationsquellen fallen weg und müssen auf anderem Wege ersetzt werden. Dafür gibt es weder ein Patentrezept noch einen einheitlichen Ansatz – im Gegenteil: Bei der Transformation zu agilen Methoden sollen ja die Teams innerhalb der definierten Prinzipien selbst empirisch erarbeitete Lösungen zum Einsatz bringen. Das ist aus Sicht der Produktdokumentation keineswegs ein zum Scheitern verurteiltes Unterfangen – vorausgesetzt, man geht auftretende Probleme aktiv an und vermeidet die Fallen, die einem unabsichtlich gestellt werden.

Die TOP-5 PROBLEME und deren Bewältigung

Unter den vielen Problemen, die bei der Einführung agiler Entwicklungsmethoden sichtbar werden, soll im Folgenden auf einige für die Produktdokumentation kritische eingegangen werden.

PROBLEM 1:

Funktionsfähige Produkte haben Vorrang vor ausgedehnter Dokumentation

Die schreibende Zunft möge aufatmen: Hier ist natürlich die bereits erwähnte Projektdokumentation gemeint: Und ja – es darf ruhig ein bisschen weniger sein. Also finden Leistungsbeschreibungen sich als unzusammenhängende Einträge im Produkt- bzw. Sprint-Backlog wieder, auf detaillierte Spezifikationen und Designpapiere wird weitgehend verzichtet – sie weichen dem kleinen Dienstweg. Eine Konsequenz dieses Abbaus von Bürokratie jedoch ist, dass sich für die technischen Redakteure die Informationsbeschaffung schwieriger und zeitaufwändiger gestaltet als im herkömmlichen Wasserfall-Modell. Hier müssen sie neue Wege beschreiten.

Abhilfe für die schwierigere Informationsbeschaffung kann in einem weiteren agilen Grundsatz gefunden werden: Individuen und Interaktionen haben Vorrang vor Prozessen und Werkzeugen. Konkret heißt das: Nur wenn der technische Redakteur als aktives Mitglied in das Teamgeschehen eingebunden ist, somit

allen relevanten Projektgesprächen beiwohnt und am informellen Informationsaustausch teilnimmt, kann er das nötige Wissen sammeln um seine Dokumentationsaufträge erfolgreich zu erledigen.

Es wird daher auch oft empfohlen, dass agile Teams auf möglichst engem Raum zusammensitzen sollten. Das ist grundsätzlich richtig, jedoch aufgrund verteilter Teams in der globalisierten Welt der Softwareentwicklung häufig nicht möglich. Darüber hinaus zeigt die Erfahrung, dass eine erfolgreiche Kommunikation im Scrum-Team nicht in erster Linie von geographischen Faktoren abhängt, sondern eher von der Eigenorganisation, der Disziplin und vor allem vom kollaborativen Teamgeist.

PROBLEM 2:

Übergreifende Themen kommen zu kurz

Das ist eine der am meisten geäußerten Beschwerden in agilen Entwicklungsumgebungen. Aus Sicht der Dokumentation bedeutet es, dass bei der Konzentration auf Sprint-Aufgaben häufig die verbindenden, konzeptuellen Inhalte fehlen, die zum Beispiel beschreiben, wie das Zusammenspiel der Produktinkremente insgesamt einen Mehrwert für den Benutzer darstellt. Hier sollte jede Organisation ein wenig experimentieren, um herauszufinden, welcher Ansatz am besten funktioniert.

Grundsätzlich gehören solche übergreifenden Themen ins Backlog und sollten dort auch entsprechend priorisiert werden, um sicherzustellen, dass sie nicht wegfallen und dass man sich um sie kümmert. Dabei sind die übergreifenden Themen einfacher zu handhaben, wenn man sich darauf einigen kann, dass nicht jedes Dokument bei Produktfreigabe sofort verfügbar sein muss, sondern dass es durchaus auch sinnvoll sein kann, bestimmte Dokumente versetzt fertigzustellen und zu publizieren.

PROBLEM 3:

In der Hackordnung sind technische Redakteure ganz unten

Das ist ein brenzliger Punkt. Fakt ist, dass der Beruf des technischen Redakteurs nicht überall das gleiche Ansehen genießt. Dokumentationsarbeit wird mancherorts als „niedere" Arbeit empfunden und technische Redakteure werden nicht überall als gleichwertig behandelt. Es gibt oftmals eine Art Wertigkeitshierarchie: Die Softwareentwicklern stehen ganz oben, gefolgt von Testern – und dann erst kommen die technischen Redakteure. Wenn bei der Einführung von agilen Entwicklungsmethoden die Entwickler, Tester und Redakteure gleichgestellt in einem Scrum-Team zusammenarbeiten sollen, ist das zunächst eine kulturelle Umstellung für alle Beteiligten.

Hier ist das Management gefragt, das die Teams behutsam und mit Fingerspitzengefühl über mehrere Iterationen dahin führt, die agilen Prinzipien immer mehr und immer besser umzusetzen. Agil werden ist ein Prozess, und der Weg, immer besser zu werden, ist ein Großteil des Ziels. Dokumentationsführungskräfte tun gut daran, mit ihren Mitarbeitern und Mitarbeiterinnen zu erörtern, wie sie gewinnbringend für alle zum Ergebnis beitragen und so ihren Stand und ihr Ansehen im agilen Team erhöhen können. Das Gegenlesen von Spezifikationen, das Fotografieren und Speichern von gemalten Bildern auf Tafeln, das Prüfen der Benutzbarkeit der Software, Vorschläge zur intuitiven Menüführung und zu den Texten auf den Dialog-Schaltflächen sowie das Übernehmen der Demo am Ende eines Sprints sind beispielsweise Aktivitäten, die das restliche Team mit Sicherheit gerne entgegennimmt. Im Gegenzug ist das Team dann eher bereit, zum Beispiel auch einmal den einen oder anderen Paragraphen zu einem besonders technischen Zusammenhang beizusteuern.

PROBLEM 4:

Technische Redakteure verbringen zu viel Zeit in Besprechungen

Der Anspruch, dass es mindestens einen technischen Redakteur pro Scrum-Team geben muss, ist in den meisten Unternehmen nicht umsetzbar. So sieht sich das Dokumentationsmanagement häufig mit mehr Scrum-Teams als verfügbaren technischen Redakteuren konfrontiert. Manche technische Redakteure müssen demnach mehr als ein Team betreuen. Das bedeutet prinzipiell aber eine Zunahme der Sprint-Planungsbesprechungen, der täglichen Stand-ups und Demos usw., sowie mehr Aufwand für die Informationsbeschaffung durch mehr persönlichen Kontakt mit den anderen Teammitgliedern.

Hier müssen die technischen Redakteure gezielt in den Teams eingesetzt werden, die an den dokumentationsintensivsten Produktinkrementen arbeiten, bzw. es muss das Backlog durchforstet werden, um zu sehen, in welchen Sprints ein technischer Redakteur am dringendsten gebraucht wird. Die Praxis zeigt, dass ein technischer Redakteur mit mehr als zwei Scrum-Teams überfordert ist. Auch dann ist es bereits eine Herausforderung, die Besprechungen so zu organisieren, dass es keine Überschneidungen gibt. Allerdings muss der technische Redakteur unter Umständen nicht an jedem Meeting teilnehmen, besonders wenn Themen besprochen werden, die keine oder nur minimale Auswirkungen auf die Dokumentation haben. Hier ist eine enge Absprache mit dem Scrum-Master erforderlich, damit der unerwünschte Nebeneffekt – nämlich die Abkoppelung des technischen Redakteurs vom Team – nicht eintritt.

Abschließend kann man aber sagen, dass die Personaldecke in Organisationen, in denen sie bereits beim Wasserfall-Modell dünn war, auch durch eine Umstellung auf Agilität nicht dicker wird. Allerdings wird im agilen Umfeld wesentlich schneller spürbar, an welchen Stellen Personal fehlt.

PROBLEM 5:

Der agile Ansatz fordert intimere Produktkenntnisse

Die Mitarbeit im agilen Team ist mit einem höheren Anspruch an die Rolle des technischen Redakteurs verbunden – sowohl was die Sozialkompetenz betrifft, als auch das technische Wissen. Das Fehlen von ausführlichen Leistungsbeschreibungen, Spezifikationen und Designdokumenten erfordert vom technischen Redakteur, dass er nicht nur näher ans Team, sondern auch näher ans Produkt rückt. Häufig findet man am schnellsten heraus, was sich in der neuesten Produktversion geändert hat, wenn man diese installiert und nachschaut. Natürlich ist das nicht immer nötig, aber es ist unschwer zu erkennen, dass bessere Produktkenntnisse zum einen die Entwickler von detaillierten Erklärungen entlasten, und zum anderen dadurch auch das Ansehen und die Akzeptanz des technischen Redakteurs im Team erhöhen. Daher liegt es auch im eigenen Interesse des technischen Redakteurs, sich mehr mit der Materie auseinanderzusetzen, als es im Wasserfall mitunter der Fall war.

„Don't Follow the Lights"[1]

Es ist eine Grunderkenntnis von agilen Teams, dass die eingesetzte Methodik zunächst keine Probleme löst, sondern sie

[1]) *Dies ist ein Zitat aus Peter Jacksons Verfilmung „The Lord of the Rings", Teil II: „The Two Towers". Die Kreatur Gollum warnt den Ringträger Frodo in den Sümpfen vor den Irrlichtern.*

erst richtig spürbar macht, und dass die Kreativität des Teams gefragt ist, um zu entscheiden, wie man mit den aufgedeckten Problemen umgeht. Viele Wege führen nach Rom – die richtige Lösung ist die, die den Erfolg bringt. Die Problemstellung in unserem Kontext ist wohl: Wie kann das Team vermeiden, dass ein Sprint aufgrund unfertiger Produktdokumentation scheitert?

Die Erfahrung zeigt, dass Teams – gerade wenn es um das Thema Produktdokumentation geht – häufig in Versuchung geraten, Probleme unter Anwendung nicht-agiler Lösungen schnell aus der Welt schaffen zu wollen. Dahinter steckt keine böse Absicht, aber man tut gut daran, nicht immer dem Licht gut gemeinter (Entwickler-)Logik zu folgen, sondern sich auf agile Grundsätze zurückzubesinnen und standhaft zu bleiben. So lassen sich einige Situationen vermeiden, die nicht notwendigerweise eine Falle sein müssen, aber leicht dazu werden können.

FALLE 1:

Der technische Redakteur ist nicht Teil des Scrum-Teams

Besonders in Unternehmen, in denen die Dokumentation zentral organisiert oder aber an Subunternehmen bzw. Freiberufler ausgelagert ist, laufen die technischen Redakteure häufig nebenher und werden nur bei Bedarf hinzugezogen. Bei der Umstellung auf Agilität neigen die Teams dann eher dazu, in Bezug auf die Produktdokumentation nichts zu ändern und die technischen Redakteure als Berater involvieren zu wollen. Lassen Sie das nicht zu! Es gibt viele Beispiele, in denen externe Dokumentationsprofis erfolgreich in agile Teams eingebunden sind. Oder überdenken Sie Ihre Praxis, an dieser Stelle mit Externen zu arbeiten. Ohne am täglichen Geschehen teilzuhaben und ohne die Produktnähe oder den Zugang zum informellen Informationsaustausch hat das agile Projekt keine Chance, die benötigte Dokumentation zu liefern. Die Rolle des technischen Redakteurs muss integraler Bestandteil des Teams sein.

FALLE 2:

Das Dokumentationsteam arbeitet in eigenen Sprints

Unter den Vorschlägen, wie ein agiles Team seine Sprints „sauber" zu Ende führen kann, ist dieser Vorschlag der Klassiker. Es hat auf den ersten Blick zugegebenermaßen etwas Verlockendes – auch für das Dokumentationsteam – ein eigenes Backlog zu pflegen und abzuarbeiten. Die Dokumentationsaufträge sind sehr schön sichtbar und der Fortschritt kann wunderbar nach oben berichtet werden. Tatsächlich bricht diese Praktik aber mit dem Grundprinzip des agilen Ansatzes, nach dem nun einmal das ganze Team für das Ergebnis verantwortlich ist – und dazu zählt eben auch das Dokumentationsinkrement.

Wenn ein Dokumentationsteam an einem eigenen Backlog arbeitet, besteht das Risiko, dass die technischen Redakteure vom Team, das auf andere User-Storys fokussiert ist, abgekoppelt werden. Die einzelnen Redakteure laufen Gefahr, den Zugang zu den Teammitgliedern und deren Informationsaustausch zu verlieren. Bei näherer Betrachtung wird nämlich klar, dass diese Organisation einem Mini-Wasserfall nahe kommt: Der Entwickler entwickelt, der Tester testet, der Dokumentator dokumentiert – und zwar wahrscheinlich in genau dieser Reihenfolge. Woher sonst rührt die Angst, die Dokumentation würde im Sprint nicht fertig? Richtig ist: Es gibt nur einen einzigen Sprint-Backlog und das gesamte Team ist für dessen Abarbeitung verantwortlich.

ABC-456 (8 subtasks)	[PERFORMANCE] Events component must perform fast enough to meet customer expectations
1050	[DEV] Define performance scenario for Events component machine
1051	[DEV] Create scenario for performance group
1052	[DEV] Create command line tool to count events over a long period
1053	[DEV] Create event generator for the performance scenario
1054	[DEV] Monitor Event component server
1055	[DEV] Create event query generator for the performance scenario
1057	[TEST] Create test for error-free running of the performance scenario
1058	[DOC] Write instructions for running the performance test

Abb. 1: User-Story inklusive Dokumentationsauftrag (die rot markierten Aufgaben stehen noch aus). Es wurde zwar der Dokumentationsauftrag ins System aufgenommen, aber nicht die Verifizierung der Dokumentation. Hier würde es sich empfehlen, folgende Aufgaben mit aufzunehmen „[TEST] Run performance scenario using the documentation and provide feedback" sowie „[DOC] Feedback incorporated in documentation".

FALLE 3:

Dokumentationsaufgaben müssen nicht ins Backlog geschrieben werden

Das ist ein sehr zweischneidiges Schwert. Wenn der Dokumentationsauftrag implizit in der Entwicklungs-User-Story enthalten ist, hat man eine übersichtlichere Anzahl von User-Storys und man könnte sich zunächst darauf einigen, dass die User-Story erst dann komplett ist, wenn die entsprechende Dokumentation geschrieben und verifiziert wurde. Im Zweifelsfall sehen die Teammitglieder das aber anders.

Nehmen wir folgendes Beispiel für eine User-Story: „Als Administrator möchte ich Benutzerdefinitionen im System anlegen". Auf jeden Fall ist es sinnvoll, folgenden Dokumentationsauftrag festzuhalten: „Als Administrator und Benutzer möchte ich nachlesen können, welche Rechte mit den verschiedenen Benutzertypen verbunden sind" – sei es als eigene User-Story oder aber als eine der Unteraufgaben der User-Story (vgl. Abbildung 1 mit Abbildung 2). ▸

ABC-789 (11 subtasks)	[USER DOCUMENTATION] As a user I want to know how to configure and operate the Events component
874	[DOC] Provide documentation of Event processing concepts
875	[DOC] Draft "Hello World" excercise for the Getting Started guide
876	[DOC] Add functional overview to the Getting Started guide
877	[DOC] Add description of the sample projects in the Getting Started guide
878	[DOC] Describe the settings and options in the configuration file
879	[DOC] Generate documentation drafts
880	[DEV] Review documentation drafts and provide feedback/corrections
881	[TEST] Verify "Hello World" excercise against the documentation draft
882	[TEST] Verify sample projects against the documentation draft
883	[TEST] Verify configuration file against the documentation draft
884	[DOC] Integrate corrections and regenerate documentation

Abb. 2: User Story bestehend aus Dokumentationsaufgaben (die rot markierten Aufgaben stehen noch aus). Das Beispiel zeigt, dass klar umrissene Dokumentationsaufgaben helfen, die Dokumentation auf das Notwendige zu reduzieren. Zudem ist bei erfolgreichem Abschluss der User-Story im Sprint der komplette Dokumentationsauftrag abgearbeitet, inklusive Verifizierung und Korrektur.

Auch wenn das Hinterlegen von sämtlichen Dokumentationsaufträgen im System die reine Anzahl der Aufgaben im Sprint verdoppeln könnte, hat diese Vorgehensweise entscheidende Vorteile:

- Alle Aufgaben sind klar definiert.
- Es wird nicht mehr und nicht weniger geschrieben, als per Auftrag definiert ist.
- Statistische Auswertungen erfassen auch den Fortschritt der Produktdokumentation.

Eine Story ist erst dann komplett, wenn die Dokumentation geschrieben und verifiziert ist. Um es noch einmal zu betonen: Das gesamte Team ist für das komplette Paket verantwortlich. Scheitern Sprints messbar an der Dokumentation, so kann beziehungsweise sollte das Management prüfen, ob ein Ressourcenproblem oder ein Prozessproblem vorliegen, und entsprechend justieren.

FALLE 4:

Dokumentation kann über Sprint-Grenzen hinweg geschrieben werden

Obwohl dieser Vorschlag bisweilen auch aus der Dokumentationsecke zu hören ist, ist er doch sehr mit Vorsicht zu genießen, denn er birgt ein Entgegenkommen an Teams, die ihre Storys nicht als Team abschließend im Sprint fertigstellen. Hier schimmert erneut die Wasserfall-Denkweise durch: Die Entwickler sind fertig, die Software ist getestet und nur die Dokumentation hinkt – wie so häufig – hinterher.

Auch wenn man sich darauf einigt, dass die Dokumentation im Sprint zwar erzeugt, aber erst im nächsten Sprint verifiziert wird (siehe Abbildung 1), lauert hier die Gefahr, dass das Team mit dem vorigen Sprint gedanklich abgeschlossen hat und für die alten Themen einfach den Kopf nicht frei hat oder dafür keine Geduld mehr hat.

Alternativ könnte man die Verifizierung in die Stabilisierungsphase oder in einen Produktfreigabe-Sprint mit aufnehmen. Hier gilt: Dieser Ansatz mag zwar in ausgereiften Teams funktionieren, aber im Allgemeinen ist es gerade in den frühen Sprints eher ratsam, innerhalb der Iterationen zum Abschluss zu kommen.

Der Weg zum Erfolg

Um in einer agilen Entwicklungsumgebung erfolgreich dokumentieren zu können, müssen einige Voraussetzungen erfüllt sein, die als Erfolgsfaktoren das Ergebnis stark beeinflussen.

Der agile technische Redakteur

Die Mitarbeit im Scrum-Team erfordert ein verändertes oder besser ein erweitertes Profil des technischen Redakteurs. Dabei stehen – neben der Bereitschaft, sich mit dem zu dokumentierenden Produkt auseinanderzusetzen – vor allem Persönlichkeitskriterien in Vordergrund. Folgende Eigenschaften gehören zum Profil des agilen technischen Redakteurs:

- Der agile technische Redakteur bringt sich aktiv und mit Eigeninitiative ein.
- Er integriert sich als gleichberechtigtes Mitglied ins Team.
- Er besitzt Durchsetzungsvermögen.
- Er passt sich schnell und flexibel an veränderte bzw. neue Situationen an.
- Er hat kein Problem damit, bereits geschriebene Abschnitte aufgrund von anderen Anforderungen oder neuen Prioritäten zu verwerfen.
- Er ist in der Lage, sich auch andere Techniken anzueignen, wie zum Beispiel CSS oder Javascript, und ist offen für weitere Aufgaben und Weiterbildungen.

Agiles Management

Agile Entwicklungsumgebungen benötigen ein Management, das eine Kultur der Gleichberechtigung der Aktivitäten beim Kodieren, Testen und Dokumentieren von Software aktiv fördert. Darüber hinaus müssen Dokumentationsführungskräfte aufgrund von Änderungen in der Marktlage und den Produktanforderungen über alle Projekte in der Organisation hinweg begleitend und steuernd tätig werden. Zu den wichtigsten Aufgaben des agilen Managements gehören:

- Sicherstellen, dass je nach Aufgabenumfeld die richtige Person mit den passenden Fähigkeiten dem jeweiligen Team zugeordnet wird.
- Aus Sicht der Dokumentation die sich ändernden Prioritäten und Aufgaben über Sprints hinweg koordinieren und das Personal entsprechend flexibel einsetzen.
- Ständig Fortschritt und Aufwand überwachen.
- Darauf hinwirken, dass die übergreifenden Themen adressiert werden.
- Den technischen Redakteuren beratend zur Seite stehen – sowohl bei Fragen zum Prozess als auch in Fragen der Dokumentationswerkzeuge und -techniken.
- Kontinuierlich die Prozesse beobachten und nach Wegen suchen, sie in kleinen Schritten so zu optimieren, sodass die technischen Redakteure sie als Verbesserung empfinden und nicht den Eindruck erhalten, sie müssten sich ständig verändern.

Automatisierung der Produktion

Technische Redakteure müssen selbstverständlich in der Lage sein, ihre Ergebnisse für interne Verifizierungs- bzw. Vorführzwecke zu jeder Zeit schnell zu erzeugen. Andererseits sollten sie von Endproduktion und Auslieferung der Produktdokumentation weitgehend befreit sein, um sich voll und ganz auf die Erzeugung der Inhalte und deren Qualitätssicherung konzentrieren zu können. Deshalb sollte die Produktion der Auslieferformate automatisiert und die technische Infrastruktur dazu zentral aufgesetzt und verwaltet sein.

Obwohl es Ziel der agilen Entwicklung ist, am Ende eines Sprints ein potenziell auslieferbares Produkt zu haben, gehören schon eine sehr ausgereifte Organisation und Infrastruktur dazu, um hier die gesamte Produktdokumentation mit einzuschließen. Natürlich muss man sicherstellen, dass Produktion und Publizierung kontinuierlich und reibungslos funktionieren. Es reicht aber aus, wenn das zunächst regelmäßig im Laufe eines Projekts passiert. Es ist dann eine Managementaufgabe, im Sinne der Prozessoptimierung dafür zu sorgen, dass die Produktionszyklen effizienter gestaltet und damit verkürzt werden können.

Tipps für die Praxis

Neben den genannten Erfolgsfaktoren gibt es eine Reihe bewährter Praktiken, die man technischen Redakteuren an die Hand geben kann, damit sie im agilen Alltag besser bestehen:

- Gehen Sie minimalistisch vor. Gerade für die Produktdokumentation gilt: Weniger ist mehr. Präzise formulierte Dokumentationsaufträge auf dem Backlog sorgen dafür, dass nicht mehr und nicht weniger dokumentiert wird als absolut notwendig. Fragen Sie nicht, „was kann ich noch dokumentieren",

sondern fragen Sie, „was kann ich noch weglassen"? Der Arbeitsberg wird dadurch vielleicht nicht so hoch und Ihre Kunden werden es Ihnen danken (siehe Abbildung 2).

- Wählen Sie den aufgabenorientierten Dokumentationsansatz, indem Sie – in Anlehnung an die User-Storys – Anleitungen schreiben, die der Benutzer befolgen muss, um die Software zweckmäßig und gezielt einzusetzen.
- Wirken Sie darauf hin, dass die Review-Aufgaben für die Qualitätssicherung auch auf das Backlog geschrieben werden, entweder als separate User-Story oder als Teilaufgabe einer Story.
- Wählen Sie ein Quellformat und Autorensystem, das es Ihnen ermöglicht, kollaborativ und modular vorzugehen. Sehr geeignet für agile Entwicklungsprojekte ist eine stringente XML-Architektur wie die *Darwin Information Type Architecture (DITA)*, die die Informationen in die Typen Aufgabe, Konzept und Referenz gliedert und für jeden Typ die entsprechende Semantik liefert (vgl. http://docs.oasis-open.org/dita/v1.2/spec/DITA1.2-spec.html). Unabhängig voneinander erzeugte DITA-Module lassen sich leicht für die Produktion zusammenfügen und strukturieren, um als Online-Hilfe, Web-Seiten oder PDF ausgegeben zu werden.
- Seien Sie mehr als „nur" der Redakteur im Team. Verfassen Sie Protokolle, ziehen Sie den Entwurf von grafischen Benutzungsoberflächen an sich, revidieren Sie Spezifikationen und geben Sie entsprechend Feedback an das Team. Sie „verkommen" dadurch keineswegs zum Teamsekretär, sondern machen sich letztendlich Ihren Job leichter und erhöhen gleichzeitig Ihren Stand im Team. Und das bedeutet: Sie werden gehört.
- Demonstrieren Sie den Stand der Produktdokumentation am Sprint-Ende. Damit erhöhen Sie die Sichtbarkeit Ihrer Arbeit.
- Seien Sie experimentierfreudig, auch wenn Ihre Versuche ab und an fehlschlagen. Dadurch lernen Sie, die passende Positionierung und Rolle im Team zu finden.

Fazit: Scrum unterstützt die Produktdokumentation

Richtig umgesetzt, kommt der agile Ansatz den Interessen der technischen Redakteure durchaus entgegen und bietet gegenüber dem klassischen Wasserfall-Modell entscheidende Vorteile:

- Teams sind sich darüber im Klaren, dass die Produktdokumentation Teil des Sprint-Ergebnisses ist.
- Informationen werden schnell und zeitnah verfügbar gemacht.
- Die täglichen Stand-ups und das retrospektive Meeting am Sprint-Ende bieten technischen Redakteuren eine Plattform, um ihren Standpunkt, ihre Interessen und ihre Bedürfnisse zu vertreten.
- Die Mitarbeit im Scrum-Team hilft technischen Redakteuren dabei, Beziehungen zu den anderen Teammitgliedern aufzubauen.
- Die Antwortzeiten für die Qualitätssicherung der Dokumentation sind kürzer.
- Der Arbeitsfluss ist konstanter: Es gibt kaum noch Leerlaufzeiten, gefolgt von Bergen von Arbeit in der Endphase eines Projekts.

Wenn es den einzelnen Mitgliedern eines agilen Team gelingt, über den eigenen Tellerrand zu schauen, zuweilen auch über den eigenen Schatten zu springen und sich die agilen Prinzipien zu eigen zu machen, wird man erstaunt sein, zu welcher Produktivität das Team in der Lage ist. Es gibt keinen Grund, warum am Ende nicht alle gewinnen sollten. ■

Literatur & Links

[Amb01] S.W. Ambler, Agile/Lean Documentation: Strategies for Agile Software Development, 2001-2010, siehe: http://www.agilemodeling.com/essays/agileDocumentation.htm

[Gent07] A. Gentle, Writing End-User Documentation in an Agile Develepment Environment, 2007, siehe: http://justwriteclick.com/2007/07/02/writing-end-user-documentation-in-an-agile-development-environment/

[Marg08] P. Margutti, Writing Software Documentation in Agile „Scrum" Teams, 2008, siehe: http://www.infomanagementcenter.com/enewsletter/200801/second.htm

[Maz11] J.-L. Mazette, Agile Technical Documentation, 2011, siehe http://www.writersua.com/articles/Agile_doc/index.html