mehr zum thema:
http://blog.codecentric.de/

WAS MUSS, WAS KANN UND WAS GEHT GAR NICHT? OPTIMALE SYSTEMDOKUMENTATION MIT AGILEN PRINZIPIEN

Die Diskussion ist fast so alt wie die IT: Wie viel Dokumentation benötigt ein System? Während es für den Kunden oft nicht genug sein kann, möchte der typische Entwickler am liebsten ausschließlich Code erzeugen. Eine leicht falsch interpretierbare Aussage aus dem Agilen Manifest zum Thema Dokumentation hat die Diskussion noch weiter aufgeheizt. Guter Rat scheint teuer. Aber so schwierig ist es letztlich gar nicht und ein wenig Systematik bei den Dokumentationsarten sowie gerade die agilen Prinzipien helfen, Inhalt und Umfang einer guten Systemdokumentation zu optimieren. In diesem Artikel werden zwei einfache Werkzeuge vorgestellt, die helfen, Dokumentation nicht mehr „auf Verdacht" zu erstellen, sondern zielgerichtet nach Sinn und Wert zu optimieren.

der autor

Uwe Friedrichsen
(E-Mail: uwe.friedrichsen@codecentric.de)
hat langjährige Erfahrungen als Architekt, Projektleiter und Berater. Aktuell ist er bei der codecentric AG für die strategische Entwicklung neuer Paradigmen und Technologien verantwortlich und beschäftigt sich in diesem Kontext insbesondere mit agilen Verfahren und neuen Architekturansätzen.

Zum Thema Systemdokumentation gibt es immer wieder lange und leidenschaftlich geführte Debatten. Von den Vertretern ausufernder, viele hundert und tausend Seiten langer Dokumentationen bis hin zu den radikalen „Code-only"-Verfechtern findet man in der IT alle Abstufungen, die ihre Positionen vehement vertreten. Gerade im Verhältnis Kunde-Dienstleister scheinen die Gegensätze häufig aufeinanderzuprallen: Während der Kunde eine umfassende Dokumentation des zu liefernden Systems fordert – häufig basierend auf ausführlichen Dokumentations-Templates, die Teil des kundeneigenen Softwareentwicklungsprozesses sind –, würden viele Dienstleister am liebsten nur die Software ohne jede Dokumentation ausliefern. Schließlich ist der Code doch die einzig zuverlässige Informationsquelle, oder?

Das agile Missverständnis

Zu zusätzlichen Irritationen führte ein Wert aus dem agilen Manifest (vgl. [Cun01]): „[...] we have come to value: Working software over comprehensive documentation" (etwa: „Funktionierende Software ist uns wichtiger als umfassende Dokumentation"). Dieser Wert wurde sowohl von Befürwortern als auch von Gegnern der Agilität häufig dahingehend (miss)interpretiert, dass die agile Softwareentwicklung nur noch lauffähige Software ausliefert und überhaupt keine Dokumentation mehr. Der relativierende Satz, „That is, while there is value in the items on the right, we value the items on the left more" (etwa: „Während wir die Dinge auf der rechten Seite durchaus als wertvoll ansehen, finden wir die Dinge auf der linken Seite wertvoller"), der den vier Werten im Manifest folgt, wird dabei meist geflissentlich überlesen.

An dieser Stelle hilft vielleicht noch einmal eine kleine Klarstellung zu der Motivation hinter dem Wert aus dem Manifest: Dokumentation hat auch in der Agilität ihren festen Platz und kein seriöser Befürworter von agiler Softwareentwicklung würde Dokumentation pauschal als unnötig bezeichnen. Der Wert an sich ist am besten zu verstehen, wenn man das Agile Manifest im Kontext seiner Entstehungszeit betrachtet: Die späten 90er Jahre waren geprägt von großen, eher schwergewichtigen Prozessen, die häufig Unmengen an Dokumentation jedweder Art produzierten. Die eigentliche Software wurde darin zur minder wichtigen Randerscheinung degradiert. Der Prozess und die ganzen Dokumente drohten zum Selbstzweck zu verkommen.

Davon distanzierte sich die agile Bewegung ganz klar. Sie stellte die Wertschöpfung in das Zentrum ihres Interesses – und der primäre Wert in der Softwareentwicklung ist eine lauffähige und qualitativ hochwertige Software. Konsequenterweise wurde die Software an sich wieder zurück in den Fokus gerückt und erhielt so ihren verdienten Stellenwert zurück. Dokumentation hingegen wurde nur als wichtig erachtet, wenn sie einen erkennbaren Mehrwert für das Gesamtergebnis beisteuert. Dieser Schritt war zu seiner Zeit sehr wichtig und lieferte wertvolle Impulse auch außerhalb der agilen Bewegung. In der Agilität geht es also nicht um eine pauschale Dokumentationsvermeidung, sondern um eine wertbasierte Abwägung, die ich im weiteren Verlauf dieses Artikels noch nutzen werde, um die benötigte Dokumentation zu optimieren.

Dokumentation ist wichtig

Um uns einer optimalen Systemdokumentation anzunähern, sollten wir vielleicht am besten mit der Frage beginnen, wie wichtig Dokumentation tatsächlich ist. Schauen wir uns dafür zwei Beispiele an:

- Als Administratoren sollen wir eine große, mehrere Dutzend Personenjahre schwere Individualsoftware in Betrieb nehmen. Leider gibt es keine Zeile Dokumentation dazu und es ist auch niemand zu finden, der uns etwas zu der Software sagen kann. Wir haben nur eine CD – leider ohne irgendeinen Installer – und das war's.
- Wir sind ein Projektteam, das seinen Projektstatus komplett transparent im Unternehmens-Wiki verwaltet. Anhand einer einfachen Darstellung kann man zu jedem Zeitpunkt sehen, was wie weit ist und ob der nächste Termin gefährdet ist. Außerdem arbeiten wir mit kurzen Iterationen von zwei Wochen und zeigen am Ende jeder Iteration die fertig gestellte Software auf dem Zielsystem. Trotzdem will unser Management, dass wir jede Woche einen Projekt-Statusbericht auf Basis einer 15-seitigen Vorlage anfertigen, damit es den Status des Projekts bewerten kann.

Mit diesen – zugegebenermaßen recht extremen – Beispielen haben wir die Band-

breite der Wichtigkeit von Dokumentation recht weit ausgelotet. Während die Dokumentation im ersten Beispiel essenziell ist (ohne sie wird der arme Administrator seine Aufgabe wohl nicht erfüllen können), ist sie im zweiten Fall eher ärgerlich und ohne erkennbaren Mehrwert.

Andererseits kann man bei einem einfachen Dienstprogramm, das mit einem fertigen Installer kommt, vielleicht komplett auf Dokumentation verzichten, während es sicherlich auch Projekte gibt, bei denen ein regelmäßiger Statusbericht die einzige Chance für das Management ist, sich einen rudimentären Einblick in den Status des Projekts zu verschaffen. Die Wichtigkeit von bestimmten Dokumenten kann man also nicht pauschal beantworten, sondern muss sie individuell bewerten. Aber wie kommen wir zu einer individuellen Bewertung der Wichtigkeit von Dokumentation? Auf dem Weg dahin fehlen noch zwei Bausteine:

- Die Aufgabe von Dokumentation: Wofür benötigen wir überhaupt Dokumentation?
- Die Arten von Dokumentation: Wie sieht eine minimale Systematik aus?

Der erste Baustein hilft uns, die Werthaltigkeit eines Dokuments zu ermitteln. Wenn ich den Zweck von Dokumentation im Allgemeinen verstanden habe, kann ich bewerten, inwieweit ein bestimmtes Dokument zu diesem Zweck beiträgt. Den zweiten Baustein benötigen wir, da es je nach Dokumentationsart unterschiedliche Herangehensweisen und Handlungsmöglichkeiten gibt.

Ziele von Dokumentation

Beginnen wir mit dem ersten Baustein: Wofür benötigen wir Dokumentation überhaupt? Was ist ihre Aufgabe? Auf diese Frage gibt der IEEE-Standard 1471–2000 (vgl. [IEEE]) eine sehr gute Antwort. Eigentlich bezieht sich dieser Standard zwar auf die Dokumentation von Architekturen, aber der hieraus relevante Teil (siehe Abbildung 1) ist für jede Art von Dokumentation zutreffend: Stakeholder haben Fragestellungen, die für sie wichtig sind und auf die sie gerne Antworten haben möchten bzw. haben müssen. Die Aufgabe der Dokumentation ist es, diese Fragen zu beantworten. Das ist die Kernaussage.

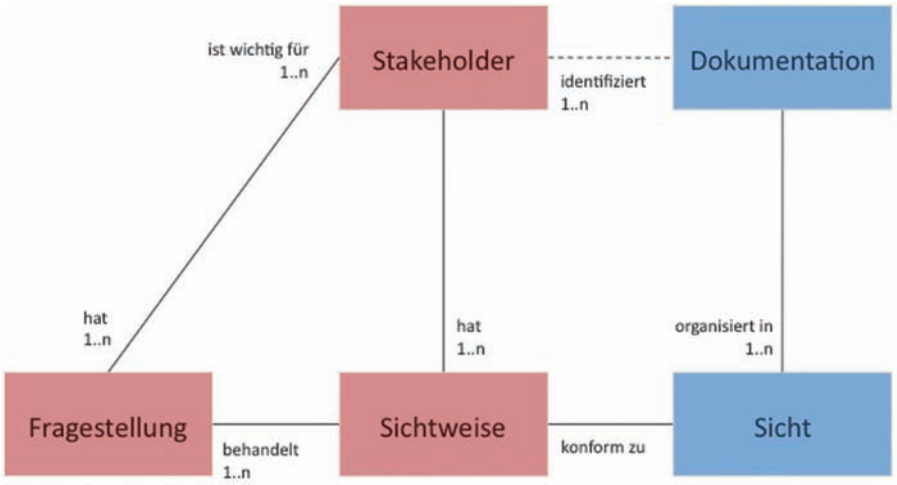

Abb. 1: Der Zusammenhang zwischen Stakeholdern, ihren Fragen und Dokumentation (nach [IEEE]).

Ergänzend empfiehlt der Standard noch, die Fragestellungen in Sichtweisen zu bündeln und die Dokumentation anhand dieser Sichtweisen zu strukturieren. Das ist eine nützliche Empfehlung, weil sie hilft, die Dokumentation zielgruppen- und fragestellungsgerecht zu organisieren.

Die Aufgabe der Dokumentation ist damit klar. Wie sieht es aber mit dem Umkehrschluss aus: Wenn Dokumentation die Fragen von Stakeholdern beantwortet, muss ich dann für jede mögliche Frage eines Stakeholders ein Stück Dokumentation schreiben? Nein, natürlich nicht.

Zum einen kann es in vielen Fällen hinreichend (und häufig auch effektiver) sein, Fragen direkt im Dialog zu klären, anstatt dafür ein Stück Dokumentation zu schreiben. Wenn etwa der Sicherheitsbeauftragte eines Unternehmens Fragen zu der neu einzuführenden Software hat, dann ist es häufig für beide Seiten wesentlich effizienter, sich einmal zusammenzusetzen und die Fragen im Gespräch zu klären, als dafür ein großes Dokument zu schreiben, das in der Regel dann doch nicht alle Fragen klärt. Wenn dieser Weg also sinnvoll ist, dann sollte man ihn bevorzugen.

Zum anderen kann auch das Produkt selbst diverse Fragen direkt beantworten. Dafür muss dann keine redundante Dokumentation geschaffen werden. So brauche ich z. B. in der Regel keine Dokumentation von Klassen und Methoden außerhalb des Quellcodes, wenn diese Information ausschließlich für die Entwickler des Systems von Interesse ist, was meistens der Fall ist. Diese Informationen können sich die Entwickler auch direkt aus dem Quellcode beschaffen. In Abbildung 2 ist das noch einmal bildlich zusammengefasst.

Projekt- und Systemdokumentation

Kommen wir zu dem zweiten Baustein: den Arten von Dokumentation. Als minimale Systematik können wir zwei grundlegende Arten von Dokumentation im Kontext der meisten Softwareentwicklungsprojekte unterscheiden:

- Projekt- bzw. Prozessdokumentation
- Produkt- bzw. Systemdokumentation

Die Projekt- bzw. Prozessdokumentation umfasst Dokumente, die an das jeweilige Projekt und sein Vorgehensmodell (den Prozess) gebunden sind. Das sind die Artefakte, die die typischen Vorgehensmodelle an den Übergabepunkten zwischen verschiedenen Rollen oder Aktivitäten vorschreiben: beispielsweise Anforderungsspezifikationen, Konzepte und Modelle, aber auch Projektpläne, Projektstatusberichte und Meeting-Protokolle. All diesen Dokumenten ist eines gemeinsam: Ist das Projekt beendet, sind sie üblicherweise nicht mehr relevant.

Die Produkt- bzw. Systemdokumentation hingegen beschreibt Dokumente, die direkt zum fertigen System gehören. Das sind z. B. Installationsanleitungen, Betriebs- und Administrationshandbücher, eventuell rechtlich vorgeschriebene Zertifikate oder auch Benutzerhandbücher. Alle Dokumente sind integraler Bestandteil des finalen Produkts, d. h. ohne sie ist das Produkt nicht voll-

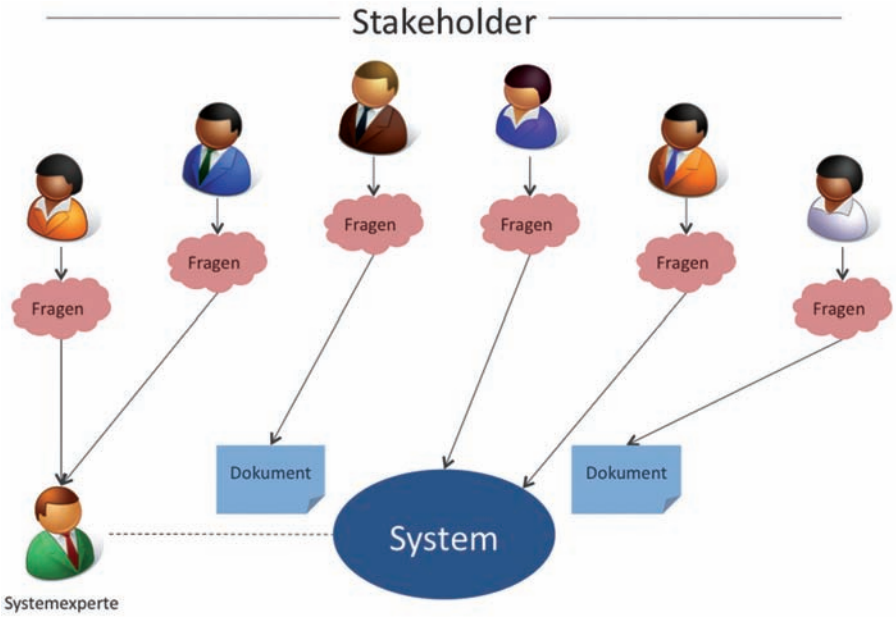

Abb. 2: Verschiedene Quellen zur Beantwortung von Stakeholder-Fragen.

ständig. Die lauffähige Software ist natürlich der Kern des Produkts, aber ohne die zugehörige Systemdokumentation ist die Software in der Regel nicht einsetzbar – gerade im Bereich der Individual-Softwareentwicklung.

Bei dieser Unterscheidung sind noch zwei weitere Aspekte von Interesse. Der erste Aspekt betrifft den Produkt-Lebenszyklus: In der Regel kann man die Erstellungsphase einer Software und ihre Betriebs- und Wartungsphase unterscheiden. Das ist nicht immer ganz einfach, z. B. in agilen Projekten, in denen die Software häufig schon produktiv gesetzt wird, während parallel dazu noch die initial geplante Funktionalität weiterentwickelt wird. Aber auch in diesen Fällen kann man die beiden Phasen zumindest inhaltlich unterscheiden, wenn auch nicht zeitlich. Das eigentliche Projekt, in dem eine Software entwickelt wird (Erstellungsphase), ist üblicherweise bezüglich Zeit und Aufwand im Verhältnis zur Betriebs- und Wartungsphase sehr klein.

Hierzu gibt es verschiedene Untersuchungen (eine Übersicht findet sich z. B. in [Kos03]). Man kann heute davon ausgehen, dass die Betriebs- und Wartungsphase bezüglich Zeit und Aufwand im Schnitt mindestens um den Faktor 5 größer ist als die Erstellungsphase (Tendenz steigend). In agilen Projekten dürfte der Faktor noch deutlich größer sein, weil diese Projekte üblicherweise sehr früh in Produktion gehen.

Die Projektdokumentation bezieht sich auf die Erstellungsphase. Das ist die Dokumentation, die man zur ordnungsgemäßen Abwicklung dieser Phase benötigt. Die Systemdokumentation benötigt man insbesondere für die Zeit nach dem Projekt, d. h. für die Betriebs- und Wartungsphase. In dieser Zeit existiert das Projektteam in der Regel nicht mehr und auch viele andere Ansprechpartner sind nicht mehr verfügbar. Die Systemdokumentation sollte daher mindestens alle Informationen umfassen, die benötigt werden, um das System auch ohne die bei der Erstellung verfügbaren Personen betreiben, nutzen und warten zu können. Nebenbei unterstreicht das Verhältnis zwischen Erstellungsphase sowie Betriebs- und Wartungsphase eindrucksvoll die Wichtigkeit der Systemdokumentation – und die relative Unwichtigkeit der Projektdokumentation (**siehe Abbildung 3**). Eine etwas andere Situation liegt vor, wenn das System über seinen ganzen Lebenszyklus von einem festen Team entwickelt und gewartet wird, wir es also zumindest inhaltlich (kommerziell kann das anders geregelt sein) nicht mehr mit abgeschlossenen Projekten zu tun haben, sondern einer kontinuierlichen Systementwicklung. Hier stünden prinzipiell Ansprechpartner über den gesamten Lebenszyklus zur Verfügung, womit sich die zuvor beschriebene Wichtigkeit von Systemdokumentation wieder etwas relativieren würde. Allerdings sind – insbesondere in Kunde/Dienstleister-Beziehungen – abgeschlossene Projekte der Regelfall und eine durchgängige Begleitung eines Systems durch ein festes Team über den gesamten Lebenszyklus ist eher die Ausnahme.

Der zweite Aspekt bezieht sich auf das Verhältnis Kunde-Dienstleister: Viele Kunden bestehen auf einer ganzen Menge an Dokumentation, die sie zusammen mit einem Softwaresystem geliefert haben wollen und ohne die sie das System nicht abnehmen. Hier muss man genauer hinschauen: Handelt es sich dabei wirklich immer um Systemdokumentation gemäß der obigen Definition oder besteht der Kunde auf dem Dokument, weil es Teil seines Prozessmodells ist bzw. er es aus anderen projekttechnischen Gründen haben will?

Das muss man genauer hinterfragen, weil es häufig nicht differenziert wird. Nicht jeder vom Prozessmodell vorgeschriebene

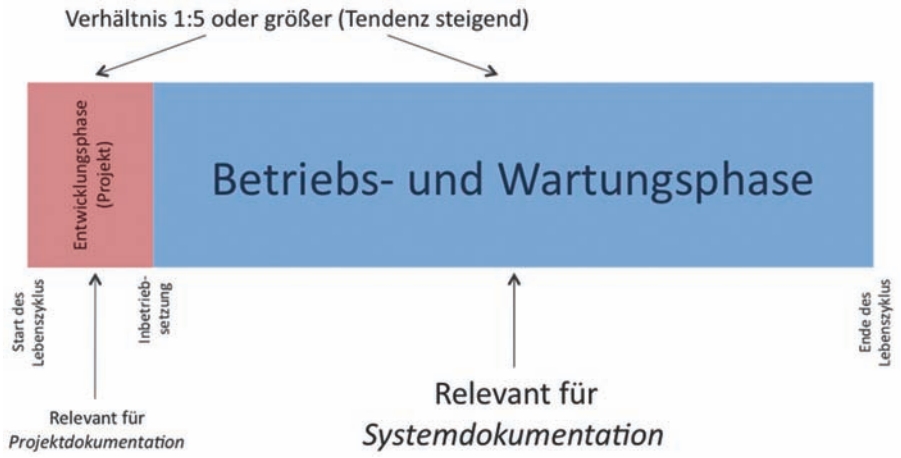

Abb. 3: Relevanz der Projekt- und Systemdokumentation im Lebenszyklus eines Systems.

Artefakt ist automatisch Projektdokumentation, sondern kann sehr wohl ein Systemdokument beschreiben. Und nicht jedes gemäß Prozessmodell abnahmerelevante Dokument ist automatisch Teil der Systemdokumentation. Für diese Unterscheidung sollte man die zuvor beschriebenen Definitionen heranziehen und jedes der geforderten Dokumente daraufhin hinterfragen. Dieses Hinterfragen kann bei der Bestimmung der optimalen Systemdokumentation sehr hilfreich sein, wie wir gleich sehen werden.

Optimale Systemdokumentation

Damit haben wir alle Informationen, die wir benötigen, um uns Gedanken über eine optimale Systemdokumentation zu machen: Wir kennen die Aufgabe von Dokumentation und können Projekt- und Systemdokumentation voneinander unterscheiden.

Für die Bewertung der Dokumentation ziehen wir den agilen Mehrwert-Gedanken heran: Wir prüfen den Wertbeitrag der Dokumentation, bezogen auf ihre Aufgabenstellung und ihren Kontext (Projekt- oder Systemdokumentation), und vermeiden Dokumentation mit geringem Wert. Das Prinzip lautet also nicht, was möglich ist, sondern was nötig ist. Dafür möchte ich zwei einfache, aber bewährte Vorgehensweisen anbieten:

- Vermeidung von Projekt- bzw. Prozessdokumentation
- Produkt- bzw. Systemdokumentation – stakeholder-basierte Analyse

Beginnen wir mit der Projektdokumentation: Diese ist keine Systemdokumentation. Streng genommen gehört dieser Teil daher gar nicht in diesen Artikel. Da hier aber erfahrungsgemäß eine Menge Optimierungspotenzial vorhanden ist, möchte ich trotzdem darauf eingehen.

Dokumentationsvermeidung

Im Projektkontext ist es wichtig, die Fragestellungen der verschiedenen beteiligten Stakeholder aktiv zu managen. Welche Fragen und Sorgen haben sie und was brauchen sie, damit sie diese Sorgen nicht mehr haben? Sofern es sinnvoll möglich ist, sollte man immer versuchen, hierfür die direkte Kommunikation und keine Dokumentation zu nutzen. Erstens ist das fast immer mit weniger Aufwand für alle beteiligten Parteien verbunden, zweitens ist es fast immer schneller und drittens ist eine direkte Kommunikation deutlich weniger fehleranfällig (im Sinne von Missverständnissen und Fehlinterpretationen) als ein Dokument.

Nur wenn der Weg der direkten Kommunikation nicht möglich ist, sollte Dokumentation eingesetzt werden. Dafür, dass eine direkte Kommunikation nicht möglich ist, kann es verschiedene Gründe geben:

- *Großes Projekt:* Je größer das Projekt ist, desto mehr formale Kommunikationsstrukturen benötigt man, da die direkte Kommunikation nicht gut skaliert. Das umfasst dann auch einen verstärkten Einsatz von schriftlicher Kommunikation, insbesondere auch Dokumentation.
- *Verteiltes Projekt:* In einem über mehrere Standorte verteilten Projekt ist direkte Kommunikation häufig nur sehr schwer möglich. Auch hier steigt die Notwendigkeit, mehr Dokumentation zur Kommunikation zu nutzen.
- *Hohes Risiko:* Ein riskantes Projekt erfordert Maßnahmen, um das Risiko beherrschbar zu halten. Das erfordert in der Regel auch mehr Dokumentation von Beschlüssen und Aktivitäten, um Transparenz und Nachvollziehbarkeit sicherzustellen.
- *Gesetzliche Anforderungen:* In verschiedenen Umfeldern gibt es gesetzliche Anforderungen, die die Dokumentation von Projektaktivitäten und Beschlüssen erforderlich machen, auch wenn sie ansonsten keinen weiteren Wertbeitrag liefern, als die gesetzlichen Vorgaben zu erfüllen (vgl. Beitrag von Engler et. al. in dieser Ausgabe von OBJEKTspektrum auf Seite 44).

Aber auch, wenn wir es wahrscheinlich in der Regel nicht schaffen werden, komplett ohne Projektdokumentation auszukommen, sollten wir doch immer den Wertbeitrag kritisch hinterfragen und prüfen, ob es nicht auch ohne das jeweilige Dokument geht oder zumindest mit einem reduzierten Dokument. Hier ein paar Beispiele:

- Brauchen wir eine detaillierte Spezifikation? Reichen nicht auch User-Storys und eine *Just-in-Time*-Abstimmung zwischen Entwickler und Fachanwender aus?
- Benötigen wir regelmäßig den erwähnten 15-seitigen Statusreport? Reicht nicht auch eine wöchentliche Telefonkonferenz aus?
- Hätte eine Vorführung lauffähiger Software alle zwei, drei oder vier Wochen nicht wesentlich mehr Aussagekraft (und würde für mehr Vertrauen bei den Projektsponsoren sorgen) als jeder noch so detaillierte Statusreport?

Ergänzend hilft es, regelmäßig zu prüfen, ob ein Dokument noch werthaltig ist. So kann eine Architekturskizze zu Beginn des Projekts sehr wertvoll gewesen, mittlerweile aber überholt sein. Anstatt die Skizze nachzupflegen, ist es häufig sinnvoller, sie als „veraltet" zu kennzeichnen und in ein Archiv zu verschieben.

Mit diesen Empfehlungen sollte es möglich sein, ein gutes Maß an Projektdokumentation festzulegen. Hier besteht bei traditionellen Vorgehensmodellen häufig das größte Einsparpotenzial – und genau das wurde auch primär im agilen Manifest adressiert. Das geht nicht von allein und Sie werden – je nach Umgebung – sicherlich eine Menge Überzeugungskraft aufbringen müssen, um lieb gewonnene, aber eigentlich wertlose Projektdokumente loszuwerden, aber den Aufwand ist es aus meiner Erfahrung wert. Außerdem ist überflüssige Dokumentation sehr teuer: Sie kostet (häufig viel) Geld, ohne einen Nutzen zu bringen. Deshalb ist allein schon aus wirtschaftlichen Gründen ein „im Zweifelsfall gegen die Dokumentation" angebracht.

Soviel als Empfehlung zur Behandlung von Projektdokumentation. Wie sieht es mit der Produkt- bzw. Systemdokumentation aus? Wie viel benötigen wir da?

Stakeholder-basierte Analyse

Ganz so einfach geht es für die Systemdokumentation nicht, weil wir ja insbesondere die Zeit nach dem Projekt betrachten müssen. Wie finden wir dafür einen optimalen Dokumentationsumfang, der auf der einen Seite alle relevanten Informationen enthält, auf der anderen Seite aber vom Umfang her so gering wie möglich ist?

Meine Empfehlung dafür ist eine stakeholder-basierte Analyse. Diese orientiert sich ebenfalls am agilen Mehrwertprinzip, indem sie den tatsächlichen Informationsbedarf systema-

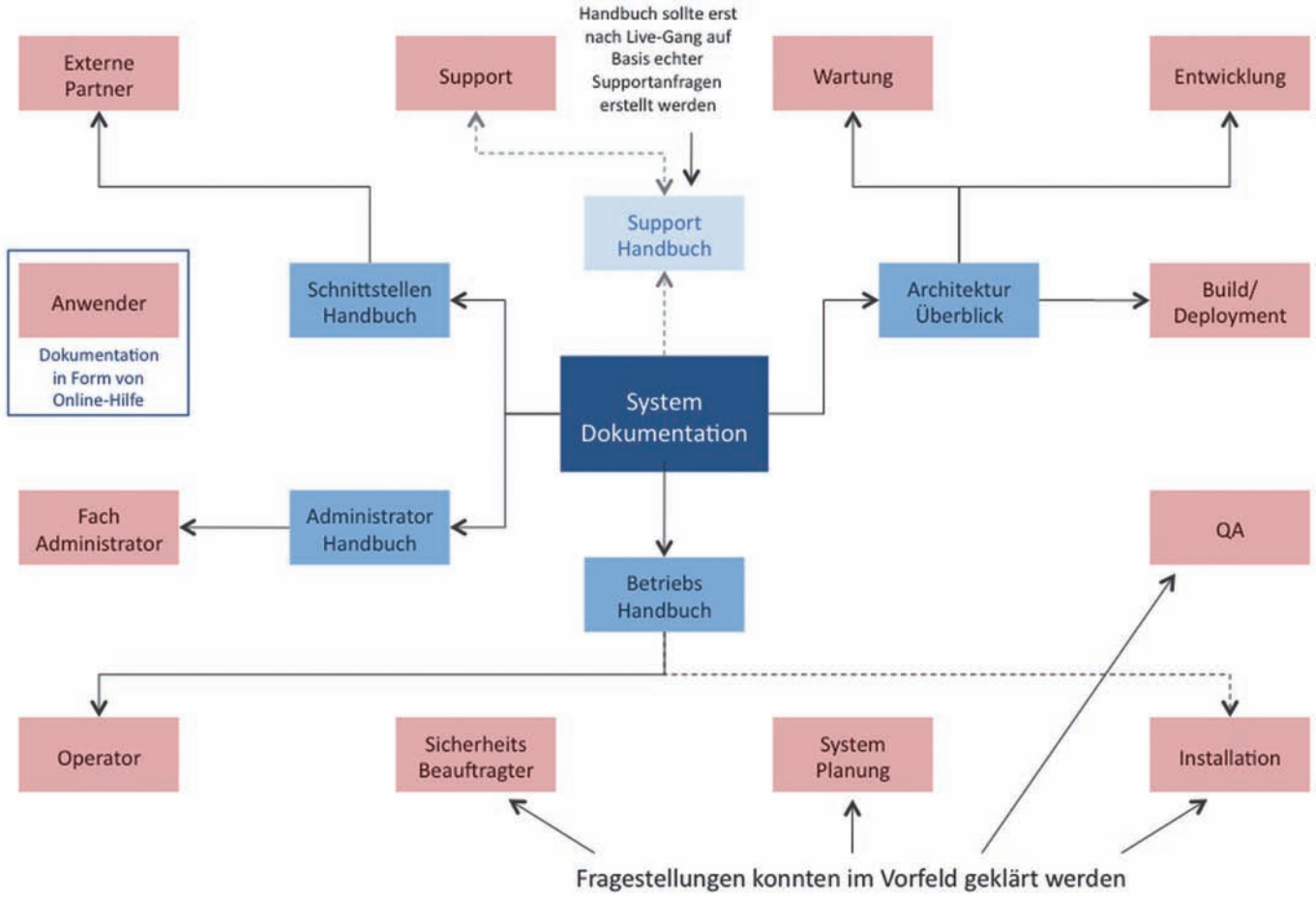

Abb. 4: Schematisches Ergebnis einer stakeholder-basierten Analyse.

tisch erfasst und damit die minimal notwendige Dokumentationsmenge zu einem System definiert. Die stakeholder-basierte Analyse orientiert sich an dem zuvor beschriebenen IEEE-Standard und funktioniert folgendermaßen:

1. Identifiziere alle für die Inbetriebsetzung sowie für Betrieb und Wartung relevanten Stakeholder. (Ich betrachte die Inbetriebsetzung ebenfalls, weil es häufig Dokumente gibt, die notwendig sind, damit das System eine Betriebsfreigabe erhält, z. B. im Sicherheits- oder Datenschutzumfeld.)
2. Identifiziere die Fragestellungen dieser Stakeholder. Dabei sollte nicht jede Einzelfrage erfasst werden, sondern die Themenfelder. Als Faustregel tut es in der Regel eine einstellige Zahl Fragestellungen pro Stakeholder.
3. Leite aus den Fragestellungen Themengebiete für die Dokumentation ab.
4. Bündle die Themengebiete nach Inhalten und Zielgruppen in Dokumenten (Stichwort „Sichtweisen").
5. Halte die Inhalte der Dokumente als Verzeichnisstruktur mit kurzen Erläuterungen je Kapitel fest („In diesem Kapitel steht …").
6. Gib die Dokumente mit den Verzeichnisstrukturen und Inhalten den Stakeholdern zum Review.

Wichtig ist hierbei, dass das es sich um keine Veranstaltung im stillen Kämmerlein handelt, sondern dass diese sehr interaktiv erfolgt. Sprechen Sie mit möglichst vielen Parteien, um die relevanten Stakeholder zu ermitteln. Es ist erstaunlich, wie häufig der Fachbereich oder die IT-Entwicklung Ihres Auftraggebers nicht wissen, wer alles in Betrieb und Wartung des beauftragten Systems involviert ist. Nicht selten werden Sie regelrechte Aha-Erlebnisse bei Ihren Auftraggebern auslösen, wenn Sie die Stakeholder transparent machen.

Raten Sie nicht, welche Fragen die Stakeholder haben könnten. Sprechen Sie mit ihnen! Zum einen werden Sie häufig feststellen, dass die Fragestellungen andere sind, als Sie angenommen haben. Zum anderen lässt sich so häufig eine Differenzierung zwischen wirklich wichtigen Informationen und „Nice to have"-Informationen herausarbeiten und damit viel Aufwand einsparen. Und drittens kann man so manche Frage auch direkt bei der Abstimmung beantworten und muss dafür dann gar nichts schreiben – wiederum gesparter Aufwand. Wichtig ist es auch, die erarbeiteten Dokumenten-Templates den Stakeholdern zum Review zu geben, um frühzeitig Feedback zu erhalten und so unangenehme Überraschungen kurz vor der Inbetriebsetzung zu vermeiden.

Sieht das nach viel Aufwand für ein solches Thema aus? Aus meiner Erfahrung, nein. Natürlich bekommt man die notwendigen Informationen in der Regel nicht auf einem Silbertablett serviert, sondern man muss einen gewissen Klärungs- und Abstimmungsaufwand betreiben, wenn man ein gemeinsames Verständnis herstellen will. Das gleiche gilt für die Anforderungen und alle anderen Informationen. Aber der Aufwand ist trotzdem recht überschaubar: einige Gespräche, der Entwurf

einiger Dokumenten-Templates und noch ein paar Abstimmungen. Das alles lässt sich in der Regel recht zügig erledigen.

Der Nutzen hingegen ist ungemein höher: Zum einen haben Sie Transparenz und ein gemeinsames Verständnis bezüglich der benötigten Systemdokumentation hergestellt – an einer Stelle, die häufig nur von Annahmen und Halbwissen geprägt ist. Das führt auch schon zum zweiten Punkt: Durch das Eliminieren der Unsicherheit bei den benötigten Informationen können Sie den Umfang der geforderten Dokumentation stark reduzieren, weil häufig keiner so genau weiß, welche Informationen benötigt werden, und deshalb sicherheitshalber alles dokumentieren lässt. Stattdessen dokumentieren Sie jetzt nur noch die wirklich benötigten Aspekte, die einen echten Mehrwert für Betrieb und Wartung liefern. Allein die hier möglichen Einsparungen rechtfertigen den Aufwand für die initiale Analyse.

Auch wenn Ihr Auftraggeber Ihnen schon – meist recht umfangreiche – Templates für die Systemdokumentation auf Basis seines Prozessmodells vorgibt, lohnt sich die Analyse in der Regel. So hatte ich die Analyse in einem Projekt durchgeführt und es ergab sich daraus ein Bild, das dem Beispiel in Abbildung 4 recht ähnlich war. Die Fragen einiger Stakeholder konnten direkt geklärt werden, so z. B. die der Systemplanung und des Sicherheitsbeauftragten. Andere hatten gar nicht so hohe Dokumentationsanforderungen, wie initial angenommen. Auf Basis der Analyse konnte die gemäß Prozessmodell eigentlich erforderliche Dokumentation deutlich reduziert werden und aufgrund der geschaffenen Transparenz war das auch für alle Beteiligten nachvollziehbar. Der Zeitaufwand für die Analyse war im Verhältnis vernachlässigbar.

Zusammenfassung

Damit haben Sie zwei einfache, agile Werkzeuge an der Hand, um eine für Sie und Ihr Projekt optimale Systemdokumentation zu schaffen: Dokumentationsvermeidung für die Projektdokumentation und stakeholder-basierte Analyse für die Systemdokumentation. Man kann jetzt bemängeln, dass das in Summe nicht sonderlich spektakulär ist, sondern nur etwas, was man eigentlich sowieso machen sollte, ausgerichtet an ein wenig agilem Gedankengut. Dennoch habe ich immer wieder überrascht feststellen müssen, wie selten es in der Praxis so gemacht wird. Deshalb kann ich Sie nur dazu ermutigen, es in Ihrem nächsten Projekt zu versuchen und Ihre Erfahrungen damit zu sammeln.

Wie in fast allen Fällen gibt es auch für gute Systemdokumentation kein Patentrezept, weshalb ich Ihnen hier auch keine fertige Dokumentationsmenge anbieten konnte. Es kommt immer auf Ihren genauen Projektkontext an, aber allein durch diese einfachen Werkzeuge haben Sie die Möglichkeit, wesentlich näher an Ihre optimale Systemdokumentation heranzukommen. ■

Literatur & Links

[Cun01] W. Cunningham et. al., 2001, Manifesto for Agile Software Development, siehe: http://www.agilemanifesto.org/

[IEEE] IEEE Std. 1471-2000, IEEE Recommended Practice for Architectural Description of Software-Intensive Systems, siehe: http://standards.ieee.org/findstds/standard/1471-2000.html

[Kos03] J. Koskinen, Software Maintenance Cost, 2003, siehe http://www.cs.jyu.fi/~koskinen/smcosts.htm