Dokumentationen in einem agilen IT-Projekt

Vorlesung *Juristisches IT-Projektmanagement*

Ludwig-Maximilians-Universität München

Autor: Johannes Groschopp
Lehrbeauftragter: Dr. Frank Sarre
Wintersemester 2018/2019

Inhaltsverzeichnis

Einleitung 2

Definition von Dokumentation 3

Arten von Dokumentationen in IT-Projekten 3

- **Dokumentation des Projektmanagements** 3
- **Anforderungsdokumentation** 4
  - Leistungsbeschreibung & Lasten- und Pflichtenheft 4
- **System- & Installationsdokumentation** 5
- **Anwenderdokumentation** 5
- **Quellcodedokumentation** 6
- **Dokumentation der Systemarchitektur** 6
- **Datenmodelldokumentation** 6
- **Schnittstellendokumentation** 6
- **Dokumentation des dynamischen Systemverhaltens** 7
- **Weitere Dokumentationsformen** 7

Dokumentation in agilen Projekten 8

- **Agile Prinzipien** 8
- **Dokumentationswerkzeuge bei agilen Projekten** 9
  - Product Backlog & User Stories 9
  - Sprint Backlog 9
  - Inkremente 10
- **Ergänzungen & Fachmeinungen zu Dokumentation in agilen Projekten** 10
  - Anforderungsdokumentation vorteilhaft 11
  - Probleme bei fehlender oder zu geringer Dokumentation 11
  - Vertragliche Leistungsvereinbarung 12

Dokumentationspflicht nach Rechtslage 12

- Rechtsfall aus 2016 14

Zusammenfassung und Fazit 14

Literaturverzeichnis 16

Einleitung

Kauft man sich heutzutage ein Haushaltsgerät im Elektrofachmarkt und möchte es anschließend erstmalig benutzen, so kommt man oftmals nicht ohne eine Anleitung aus, die beschreibt, wie man das Gerät bzw. die Maschine benutzen muss. Teils kompliziertere, nicht von vornherein ersichtliche Schritte müssen durchgeführt werden, um das Gerät letztendlich zum Laufen zu bringen und es sinngemäß benutzen zu können. Auch in der IT sind Benutzerhandbücher oder generell Dokumentationen von ähnlicher Bedeutung. Soll man beispielsweise am Arbeitsplatz eine neu eingeführte Software benutzen, bedarf es oftmals einer Beschreibung, wie man die Software zu benutzen hat. Liegt solch eine Beschreibung nicht vor, kann man als Endnutzer schnell ratlos werden und eine sinnvolle Nutzung der Software ist zunächst einmal nicht gegeben. Wie in (Juhl, 2015) erwähnt, ist es die Gebrauchsanleitung, die den Käufer eines Konsumguts zum Produktnutzen führt. So führt in der Regel kein Weg an einer angemessenen Dokumentation des Produkts vorbei.

Die vorliegende Arbeit beschäftigt sich mit dem Thema Dokumentationen in einem agilen IT-Projekt. Dabei soll anhand ausgewählter Fachliteratur zunächst untersucht werden, welche Arten von Dokumentationen es bei Softwareprojekten gibt. Außerdem wird des Weiteren die Situation bei agilen IT-Projekten unter die Lupe genommen. Wie im Agilen Manifesto[1] deklariert, wird bei der agilen Softwareentwicklung eine funktionierende Software größer geschrieben als eine umfassende Dokumentation. Was das nun für die Dokumentation bei agilen Projekten bedeutet, wird anhand verschiedener Artikel aus der Fachliteratur untersucht. Des Weiteren wird die rechtliche Lage betrachtet. Ist es gesetzmäßig verpflichtend, eine Dokumentation zur Software mitzuliefern?

Die Arbeit ist also wie folgt gegliedert: Als erstes wird eine kurze Definition von Dokumentation auch im Kontext von Software gegeben. Anschließend erfolgt ein Überblick über die gängigen Dokumentationen bei Softwareprojekten. Darauf wird die agile Vorgehensweise bei Softwareprojekten am Beispiel von *Scrum*[2] kurz thematisiert. Unter der Annahme, dass Dokumentation bei agilen IT-Projekten als eher unwichtig erscheint, werden Expertenmeinungen aus der Fachliteratur zum Thema Dokumentation innerhalb agiler Projekte untersucht. Festzuhalten ist, dass in den Meinungen verschiedener Stimmen ein gewisser Konsens herrscht und ein angemessenes Maß an Dokumentation auch bei agilen IT-Projekten absolut notwendig ist. Der Umfang der Dokumentation richtet sich dann nach diversen Parametern wie unter anderem der Art der Leistung, den Adressaten oder der Komplexität eines Projektes. Zu guter Letzt wird zudem ein Blick auf die rechtliche Lage bezüglich Dokumentationen in Softwareprojekten geworfen.

[1] http://agilemanifesto.org

[2] https://www.it-agile.de/wissen/einstieg-und-ueberblick/scrum/

Definition von Dokumentation

Es gibt verschiedene Arten von Dokumentationen, die bei IT-Projekten zur Verwendung kommen und die Planung, das Management, die Entwicklung sowie die Benutzung von Software adressieren und unterstützen.
In (Liesegang, 2015) wird Dokumentation als ein Wissen verkörpernder Gegenstand bezeichnet, welcher in verschiedenen Erscheinungsformen wie textuell, in natürlicher Sprache, in Diagrammen, in Bildern oder in Audio-oder Videoaufzeichnungen erscheinen kann. Durch eine Dokumentation entsteht die Möglichkeit, Wissen an mehrere verschiedene Personen weitergeben zu können. Des Weiteren führt (Liesegang, 2015) aus, dass Dokumentation nicht nur aufgrund von formellen Voraussetzungen angefertigt, sondern dies bedarfsgerecht und zielgerichtet geschehen sollte, sodass der Inhalt und das darin vermittelte Wissen adressatengerecht wiedergegeben werden.
Wie in (Koch, Computer-Vertragsrecht, 2009) erwähnt, ist der Begriff „Dokumentation“ (im Software-Kontext) nicht eindeutig festgelegt. Der Autor beschreibt, dass hier zwischen Benutzerdokumentation und Wartungsdokumentation zu unterscheiden ist.

Arten von Dokumentationen in IT-Projekten

Die in der Fachliteratur genannten Arten von Dokumentationen für IT-Projekte werden nun zum Teil im Folgenden aufgeführt. Es sei angemerkt, dass dies lediglich einem Teil der möglichen Dokumentationen für IT-Projekte entspricht. In Form von zahlreichen Normen und Vertragswerken (wie z.B. DIN-Normen, ISO-Standards oder dem V-Modell XT) wurden wie in (Schreiber-Ehle, 2015) aufgezeigt, Dokumentationsverfahren verbindlich festgelegt. Somit gibt es eine große Landschaft von Dokumentationsverfahren. Grob lassen sich zwei Kategorien von Dokumentationsarten beschreiben. Zum einen gibt es Dokumentationen, die vor oder während eines Softwareherstellungsprozesses erstellt werden und der Herstellung selbst dienen, sowie Dokumentationen, die als Beilage zur fertigen Software übergeben werden. In (Koch, IT-Projektrecht, 2007) werden (zumindest für größere Anwendungen) die folgenden vier Dokumentationstypen grob unterschieden: Anwenderdokumentation, Systemdokumentation, Betriebsdokumentation und Installationsdokumentation.

Dokumentation des Projektmanagements

Eine Form von Dokumentation in einem IT-Projekt stellt die des Projektmanagements dar. Diese wird in (Liesegang, 2015) genannt und ist für diejenigen Personen von Bedeutung, die ein Projekt Leiten und dessen Verantwortung tragen. Solch eine Dokumentation des Projektmanagements beinhaltet laut (Liesegang, 2015) ausgewählte, wesentliche Daten über Konfiguration, Organisation, Mitteleinsatz, Lösungswege, Ablauf und die erreichten Ziele eines Projekts. Unter anderem sind in Norm DIN 69901-2:2009 *Projektmanagement – Projektmanagementsysteme* 14 Prozesse, darunter beispielsweise die Definition von Zielen oder die Erstellung eines Terminplans, als Bestandteil des Mindeststandards definiert. In der Regel muss eine Projektmanagementdokumentation nach Erstellung eines Projekts nicht an den Auftraggeber (AG) übergeben werden, wie in (Liesegang, 2015) beschrieben.

Anforderungsdokumentation

Eine ganz zentrale Rolle bei der Dokumentation in der Softwareerstellung nimmt die Anforderungsdokumentation ein, welche die gewünschte Softwarelösung möglichst vollständig umschreibt. Wie in (Hoppen, 2015) erklärt, führt eine unzureichende Dokumentation der Anforderungen an ein Softwareprojekt zu vielen Problemen. „Wenn eine Anforderung erst während der laufenden Entwicklung erkannt wird, sind die Kosten schnell bis zu 10fach so hoch, als wenn sie Gegenstand der Leistungsbeschreibung gewesen wäre. Wenn die Anforderung erst nach der Einführung des Systems umgesetzt wird, können die Kosten auch schnell das 100fache betragen“ (Hoppen, 2015). Auch aus rechtlicher Sicht ist eine Anforderungsdokumentation nach Hoppen, vereidigter IT-Sachverständiger, sinnvoll, da im Falle einer juristischen Aufarbeitung durch eine vorhandene Anforderungsdokumentation das Feststellen von Sachmängeln leichter fällt und diese somit klarer erkannt werden können. Des Weiteren führt der Autor aus, dass es für die Erstellung einer Anforderungsdokumentation, genauer zum Sammeln der Systemanforderungen, zunächst der Einbeziehung von *Stakeholdern* bedarf, also derjenigen Personengruppen, für die die Software letztendlich hergestellt wird. Wichtig ist nach Hoppen letztendlich, dass eine Anforderungsdokumentation eindeutig, konsistent und nach Möglichkeit widerspruchsfrei sein sollte, was unter anderem auch im IEEE Standard 830 festgehalten ist.

Leistungsbeschreibung & Lasten- und Pflichtenheft

Wie in (Hoppen, 2015) beschrieben gibt es zunächst einmal die Begrifflichkeiten *Lastenheft* und *Pflichtenheft* im Kontext der Anforderungsdefinition. Speziell bei Softwareprojekten die nach klassischen Vorgehensmodellen wie z.B. dem *Wasserfallmodell*[3] finden diese Verwendung. Die Begrifflichkeiten *Lastenheft* und *Pflichtenheft* kommen ursprünglich aus dem Maschinenbau, wie in (Hoppen, 2015) erklärt. Hoppen führt aus, dass ein *Lastenheft* in erster Linie ergebnisrelevant, ein Pflichtenheft dagegen aufwandsrelevant ist. Somit beschreibt ein *Lastenheft*, was die zu erstellende Software können soll, das *Pflichtenheft* hingegen beschreibt, wie die Lösung umgesetzt wird. Dies macht deutlich, dass ein *Lastenheft* dann nur vom Auftraggeber (AG) erstellt werden kann, da dieser ja die Software fordert und deklarieren muss, was er als Ergebnis fordert. Das *Pflichtenheft* hingegen, so Hoppen, sollte unter maßgeblicher Beteiligung des Auftragnehmers (AN) erstellt werden, da dieser in der Regel die Fachkompetenz bezüglich der technischen Umsetzung des Projekts mitbringt. Gegebenenfalls kann das Pflichtenheft auch komplett von Seiten des AN erstellt werden. Dies wird so auch in (Roth & Dorschel, 2008) ausgeführt: „Nach der DIN 69901 basiert das Pflichtenheft auf dem vom Auftraggeber vorgegebenen Lastenheft und wird vom Auftragnehmer erstellt“. Allerdings steht dies nach (Roth & Dorschel, 2008) scheinbar im Widerspruch zur BGH-Rechtsprechung, die die Verantwortung zur Erstellung des Pflichtenhefts primär dem AG zuweist. Generell scheint es bei den Begrifflichkeiten *Pflichtenheft* oder *Lastenheft* keine eindeutige Einigkeit zu geben, da wie in (Ihde, 1999) beschrieben, der Begriff des *Pflichtenheftes* weder in der Rechtsprechung noch in der Literatur einheitlich gebraucht wird, noch von Synonymen wie fachliches Feinkonzept, Leistungsbeschreibung oder *Lastenheft* abgegrenzt wird. Eine Leistungsbeschreibung besteht nach (Hoppen, 2015) aus funktionalen und nicht-funktionalen Anforderungen. Funktionale Anforderungen beschreiben wie in (Hoppen, 2015) erklärt, was die Software leisten soll, ähnlich wie beim *Lastenheft.* Bezüglich nicht-funktionalen Anforderungen wird in (Hoppen,

[3] http://www.infrasoft.at/downloads/Vorgehensmodelle_in_der_Softwareentwicklung.pdf

2015) erklärt, dass sich diese primär auf die Software-Qualitätskriterien beziehen, sogenannte Qualitätsanforderungen. Zu nennen sind hier Kriterien wie Sicherheit, Bedienbarkeit, zukünftige Wartbarkeit. Außerdem werden hier auch die Anforderungen an die zu erstellende Dokumentation beschrieben. Diese definieren, welche Dokumentationen als Teil der geschuldeten Leistung gefordert werden. Wie wichtig solch eine vertragliche Vereinbarung ist, konkretisiert er damit, dass es in vielen Softwareentwicklungs-Projekten gängige Praxis sei, dass die Software irgendwie ans Laufen gebracht werde und außer dem Quellcode so gut wie keine weitergehende Dokumentation übergeben werde. Außerdem sollten Testfälle nach (Hoppen, 2015) direkt aus den Anforderungen ableitbar sein, damit anhand gut definierter Testfälle auch valide überprüft werden kann, ob die Software das leistet, was sie leisten soll.

System- & Installationsdokumentation

Damit eine Software überhaupt erst in Betrieb genommen werden kann, Bedarf es nach Hoppen unbedingt einer Dokumentation der Installation und des Systems. Wichtige Inhalte hebt der Autor indes heraus. *Angaben für das Betriebskonzept* und *Verhalten in besonderen Fällen* sind bei Störfällen von hoher Bedeutung, um das System anschließend wieder in Betrieb nehmen zu können (speziell bei größeren und komplexen Systemen). Außerdem wird noch das *Notfall-Konzept* genannt, welches ebenfalls im Falle eines Ausfalls zum Beispiel zum Tragen kommt. Nach (Koch, IT-Projektrecht, 2007) werden in einer Systemdokumentation technische Angaben zusammengefasst, die für den Betrieb, Wartung und Pflege relevant sind, eine Installationsdokumentation sollte den Status nach Installation einschließlich erfolgter Parametrisierung festhalten.

Anwenderdokumentation

Die Anwender- oder auch Benutzerdokumentation wird in mehreren Artikeln aus der Fachliteratur genannt und nimmt deshalb eine ebenfalls sehr wichtige Rolle innerhalb der zu übergebenden Dokumentationsarten ein. Eine Anwenderdokumentation richtet sich an die Endnutzer des Systems, also in der Regel an die Personen, die i.d.R. die Oberfläche der Software bedienen. Sie ist nach Rechtsprechung eine Mindestanforderung und muss in jedem Fall bei der Übergabe der Software als Dokumentation beiliegen, ist somit auch ohne jegliche Vereinbarung geschuldet (Hoppen, 2015). Bestätigt wird dies mitunter auch in (Beckmann, 1998). Als durch die BGH-Rechtsprechung rechtlich geklärt bezeichnet Beckmann die Pflicht zur Überlassung einer Bedienungsanleitung. Dies sei sogar eine Hauptpflicht des Lieferanten. Zudem, so Beckmann liegt die Beweislast bezüglich einer vollständigen und fehlerfreien Aushändigung der Handbücher beim AN. Viele Webseiten und Softwareprodukte sind heutzutage allerdings schon so selbsterklärend, dass die Notwendigkeit einer Anwenderdokumentation zumindest in Frage gestellt werden kann. Denn tatsächlich ist es so, „[...] dass viele komplexe Anwendungen im Consumer-Bereich (man denke etwa an große Web-Portale wie *Amazon* o.Ä.) heute von breiten Bevölkerungsschichten ohne jedes Benutzerhandbuch intuitiv bedient werden können“. Möchte der Kunde keine Anwenderdokumentation bekommen, sollte dies vor dem Hintergrund der BGH-Rechtsprechung vertraglich auch explizit festgehalten werden. Denn wie in (Stiemerling, 2011) beschrieben kann der Kunde je nach Art von Software eine Auslieferung komplett ohne Handbuch beanspruchen und dass die Software allein durch eine intuitive, selbsterklärende grafische Benutzeroberfläche bedienbar sein sollte.

Quellcodedokumentation

Quellcode kann heutzutage sehr komplex und auf den ersten Blick schwer durchschaubar sein. Gerade bei objektorientierter Programmierung, bei welcher viele Klassen, Interfaces etc. entstehen können, kann es für den Entwickler oft schwierig sein, sich schnell in den Quellcode einzulesen und diesen zu verstehen. Um dem vorzubeugen empfiehlt Hoppen, bereits in der Anforderungsdokumentation qualifizierte Anforderungen an die Quellcode-Kommentierung festzuhalten. Denn wie der dieser ausführt, kommt es nicht selten vor, dass zwar sogenannte Inline-Kommentare verfasst werden, diese aber inhaltlich trivial sind und somit für den Entwickler keinen Mehrwert haben. Abbildung 1 zeigt ein Beispiel für triviale Inline-Kommentare aus (Hruschka, Rupp, & Starke, 2009). Die Tatsache, dass die Variable i um eins erhöht wird, lässt sich aus dem Code bereits lesen und bedarf daher keiner obsoleten Kommentierung.

```
i = i + 1;
// i wird um eins erhöht
```

Abb. 1: Beispiel für trivialen Inline-Kommentar aus (Hruschka, Rupp, & Starke, 2009)

Aus den Quellcode-Kommentaren lässt sich anschließend mit bestimmten Werkzeugen wie zum Beispiel *Javadoc*[4] automatisch eine sinnvolle Dokumentation generieren, wie in (Schreiber-Ehle, 2015) angemerkt. Allerdings ergibt sich aus der Quellcodekommentierung lediglich eine Übersicht über Funktionen, Klassen sowie Hinweise auf Implementierungsdetails, was für komplexe Systeme oftmals nicht ausreicht. Des Weiteren führt die Autorin aus, dass eine technische Dokumentation des Quellcodes grundsätzlich unabhängig zum Prozessmodell ist.

Dokumentation der Systemarchitektur

In (Hoppen, 2015) wird beschrieben, dass es bei heutigen Softwaresystemen außerdem wichtig zu verstehen ist, wie von der Softwareanwendung erfasste Daten verarbeitet werden und welche Zusammenhänge außerdem zwischen einzelnen Funktionsbereichen innerhalb der Software bestehen. Bei sehr komplexen Systemen erscheint es als sehr sinnvoll, die Architektur beispielsweise der Infrastruktur innerhalb einer dafür vorgesehenen Dokumentation festzuhalten.

Datenmodelldokumentation

Viele heutige Softwareanwendungen arbeiten mit einer großen Menge an Daten, betrachtet man einmal zum Beispiel Data Warehouses[5]. In (Hoppen, 2015) wird erläutert, dass für den Entwickler ein gut beschriebenes Datenmodell essentiell sei, da es beispielsweise bei der Auswertung der Datenbestände zum besseren Verständnis beiträgt.

Schnittstellendokumentation

Schnittstellen[6] spiegeln, wie in (Riedl, 2018) erwähnt, Berührungspunkte innerhalb eines Softwaresystems wider und kommen in Softwaresystemen, die beispielsweise in einer objektorientierten Programmiersprache geschrieben wurden, häufig vor. Über den Inhalt von Schnittstellendokumentationen gibt (Nowak, 2002) genauere Auskunft: „*Schnittstellenbeschreibungen* sind Dokumente, welche Anzahl, Typen und erlaubte Reihenfolgen von Daten angeben, die zwischen Softwaremodulen ausgetauscht werden.“

[4] https://www.oracle.com/technetwork/java/javase/documentation/index-jsp-135444.html

[5] https://www.bigdata-insider.de/was-ist-ein-data-warehouse-a-606701/

[6] https://www.it-business.de/was-ist-eine-schnittstelle-a-714831/

Eine Funktionsbeschreibung der Syntax, Semantik und Fehlerbehandlung wird der Schnittstellendokumentation in (Schreiber-Ehle, 2015) zudem zugeschrieben.

Dokumentation des dynamischen Systemverhaltens

Eine weitere in (Schreiber-Ehle, 2015) genannte Dokumentationsart ist die Beschreibung des dynamischen Systemverhaltens, also die Reihenfolge, in der die Anweisungen des Quellcodes durchlaufen werden. Diese ist nach (Schreiber-Ehle, 2015) vor allem bei komplexeren Systemen wichtig, da Systemabläufe dort nicht allein aus dem Quellcode ablesbar sind.

Weitere Dokumentationsformen

Wie in (Schreiber-Ehle, 2015) angemerkt, kann der Umfang einer Dokumentation für ein Softwareprojekt je nach Gebrauch differieren, wobei die Leistungsart maßgeblich für Notwendigkeit von bestimmten Dokumentationsarten ist. Mögliche Dokumentationsformen, die in (Schreiber-Ehle, 2015) erwähnt werden, sind in Abbildung 2 aufgezeigt. Abhängig von der Leistungsart stellt sich dann die Sammlung der notwendigen Dokumentationen zusammen.

It-Leistung	Benutzerhandbuch	Anpassungsanleitung	Schnittstellendokumentation	Anforderungsdokumentation	Entwurfsdokumentation	Sourcecode-Dokumentation	Übersetzungs-und Testanleitung	Betriebsanleitung	API-Dokumentation
Anpassbare Software		x							
Integrierbare Software			x						
Softwareprodukte									x
Escrow/Entwicklerübergabe	x	x	x	x	x	x	x	x	x

Abb. 2: Tabelle aus (Schreiber-Ehle, 2015)*, die abhängig von der Leistungsart die notwendigen Dokumentationsarten aufzeigt*

Schreiber-Ehle merkt in (Schreiber-Ehle, 2015) an: „Im Rahmen einer Softwareanpassung werden nur die Dokumentationen benötigt, die sich auf den anpassbaren Teil der Software beziehen. Im Rahmen einer Integration wird die Software mit einer anderen Software verbunden, so dass beide miteinander kommunizieren. Zu diesem Zweck wird eine Beschreibung der Schnittstelle benötigt. Ein Einbau des Softwareproduktes in eine andere Software durch Programmierung erfordert eine API[7]-Dokumentation. Diese beschreibt die Aspekte der Software, die bei der Einbettung durch die andere Software verwendet werden. Im Falle der Entwicklerübergabe wird die Pflege und Weiterentwicklung an ein anderes Entwicklungsteam übergeben. Dieses Team benötigt umfassende Informationen über das System."

[7] Application Programming Interface- Programmierschnittstelle zur Nutzung einer Software. (vgl. (Schreiber-Ehle, 2015)

Dokumentation in agilen Projekten

Es gibt verschiedene agile Methoden, welche auch in der Literatur genannt sind. Darunter beispielsweise *Scrum*, *eXtreme Programming (XP),* oder auch *Crystal Family*. Diese Methoden werden unter anderem in (Koch, Agile Softwareentwicklung - Dokumentation, Qualitätssicherung und Kundenmitwirkung, 2010) aufgeführt. Hält man sich die Aussagen der Literatur vor Augen, wird deutlich, dass das Thema Dokumentation bei agilen Projekten oft vernachlässigt zu sein scheint. Dies wird unter anderem in (Bednarczyk & Queins, 2013) so bestätigt. Nichtsdestotrotz machen die Autoren klar, dass eine Anwendung von agilen Verfahren keine Abschaffung von jeglicher Dokumentation bedeutet. Im Folgenden werden nun die Prinzipien der agilen Entwicklung, entnommen aus dem Agilen Manifesto[8] kurz erläutert. Anschließend werden Dokumentationspraktiken und daraus resultierende Dokumentationen bei agilen Projekten, welche in der Fachliteratur beschrieben sind, aufgezeigt. Fallbeispiele erfolgen entlang der agilen Methode *Scrum*, zu welcher in der Literatur umfassende Informationen zu finden sind.

Agile Prinzipien

Den Aussagen des Agilen Manifestos[8] ist zu entnehmen, dass *Individuen und Interaktionen* höher als *Prozesse und Werkzeuge, Funktionierende Software* höher als *umfassende Dokumentation, Zusammenarbeit mit dem Kunden* höher als *Vertragsverhandlung* und *Reagieren auf Veränderung* höher als das *Befolgen eines Plans* bewertet werden. Daraus lässt sich zunächst die Vermutung herleiten, dass alles bürokratischen Angelegenheiten, die im Zuge eines Softwareherstellungsauftrags zwischen AG und AN existieren, als unwichtig angesehen werden und daher auch vernachlässigt werden können. Wie in (Liesegang, 2015) erwähnt, ist dem auch oft in der Praxis so und es werden Dokumentationen in Softwareprojekten gerne vernachlässigt. Jedoch wird im Agilen Manifesto[8] auch gesagt, dass, obwohl die Werte auf der linken Seite höher eingeschätzt werden, auch die Werte auf der rechten Seite als wichtig bewertet werden. Wie bereits weiter oben erwähnt und auch in der Fachliteratur bestätigt, rechtfertigt die Anwendung agiler Methoden also auf keinen Fall die Vernachlässigung, geschweige denn den Verzicht auf Dokumentation.

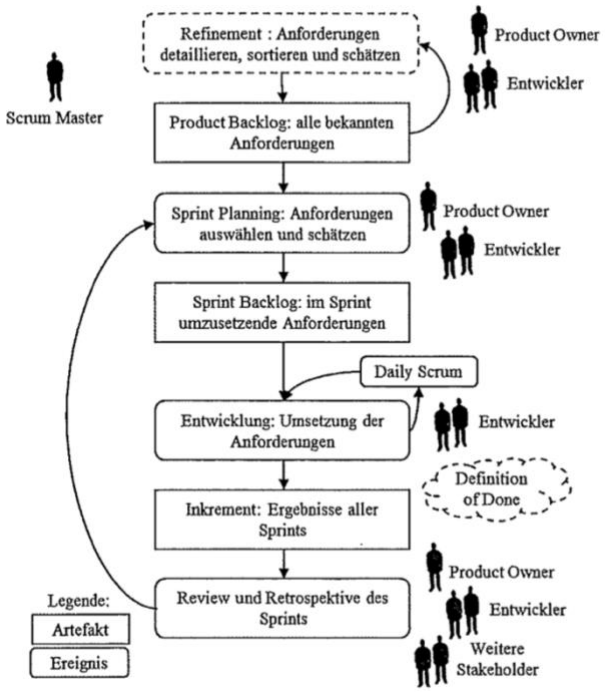

Abb. 3: Struktur und Ablauf eines Scrum-Projekts aus (Liesegang, 2015).

[8] http://agilemanifesto.org/

Dokumentationswerkzeuge bei agilen Projekten

Die Werkzeuge aus der agilen Entwicklung zur Erstellung von Dokumentation hinsichtlich der Bereitstellung gewisser Dokumentationen, werden nun anhand der agilen Methode *Scrum* nähergelegt. Wie in (Liesegang, 2015) erwähnt, ist *Scrum* die populärste agile Methode und verkörpert im Sinne des Projektmanagements einen Mindestumfang, welcher entsprechend der Rahmenbedingungen eines konkreten Projektes zu ergänzen ist. Weiter wird in (Liesegang, 2015) ausgeführt, dass *Scrum* ein Rahmenwerk ist, welches im Wesentlichen aus Teams und deren Rollen (Product Owner, Entwickler und Scrum Master), Ereignissen (Sprint, Sprint Planning, Daily Scrum, Sprint Review, Sprint Retroperspective), den Artefakten (Product Backlog, Sprint Backlog, Inkrement) und Regeln besteht. Um eine angemessene Dokumentation in *Scrum*-Projekten zu erzielen, kann diese nach (Herzog, Holzmüller, & Schoepe, 2010) durch Aufwertung und Aufarbeitung vorhandener Artefakte des agilen Vorgehens erreicht werden. Werkzeuge, welche das Scrum-Framework bereitstellt, sind in (Hoppen, 2015) kurz beschrieben. So dienen beispielsweise *Backlogs* als Themenspeicher für Anforderungen, *Visions* und *Goals* als Anforderungen an das Gesamtsystem, *Epics* als Beschreibung noch grober Anforderungen und *User Stories* als Beschreibungsmittel für funktionale Anforderungen. Ebenfalls erwähnt sind *Test Cases,* welche bei agiler Entwicklung ein wichtiges Mittel zur Konkretisierung von Anforderungsdetails sind und welche wiederum z.B. aus funktionalen Anforderungen abgeleitet werden können. Abbildung 3 zeigt einen typischen Projektablauf mit dessen Komponenten. Ein Augenmerk sei hier auf die Artefakte gelegt, welche für die Dokumentation wichtig sind.

Product Backlog & User Stories

Wie Abbildung 3 zu entnehmen ist, beinhaltet das erste Artefakt, der Product Backlog, alle bekannten Anforderungen an die Software. Die Autoren in (Herzog, Holzmüller, & Schoepe, 2010) erklären, dass das Product Backlog sogenannte User Stories enthält und als Darstellung eines Fachkonzepts dient. Beim *XP* beispielsweise werden nach (Koch, Agile Softwareentwicklung - Dokumentation, Qualitätssicherung und Kundenmitwirkung, 2010) die Anforderungen in Form von User Stories möglichst vom Kunden selbst auf Story Cards geschrieben. Nach (Bednarczyk & Queins, 2013) sieht dies bei *Scrum* ähnlich aus. Die Autoren erwähnen, dass bei *Scrum* Anforderungen in Form von User Stories dokumentiert und in der Theorie auf Kärtchen geschrieben werden. Diese User Stories können dann wohl im Product Backlog hinterlegt werden. Auch sog. Goals werden nach (Liesegang, 2015) im Product Backlog festgehalten. Die Sammlung an Anforderungen ist am Ende dann wie in (Koch, Agile Softwareentwicklung - Dokumentation, Qualitätssicherung und Kundenmitwirkung, 2010) angemerkt gewissermaßen ein „lebendes“ Pflichtenheft.

Sprint Backlog

Sind die Anforderungen im Product Backlog definiert, können innerhalb von Sprints einzelne Teilmodule entwickelt werden. Wie in (Koch, Agile Softwareentwicklung - Dokumentation, Qualitätssicherung und Kundenmitwirkung, 2010) definiert, können aus dem Product Backlog die für den im nächsten Entwicklungsschritt abzuarbeitenden Anforderungen im Sprint Backlog gesammelt werden (siehe auch Abb. 3).

Inkremente

Wie Abbildung 3 zeigt, enthält ein Inkrement die Ergebnisse aller Sprints und konsolidiert diese somit zum aktuellen Ist-Stand der Entwicklung. Somit lässt sich aus dem Inkrement lesen, welche der im Product Backlog enthaltenen Anforderungen bereits erfüllt sind. In (Hruschka, Rupp, & Starke, 2009) ist dies ebenso aufgeführt, dort wird gesagt, dass Inkremente in kurzen Iterationen entwickelt werden, die für sich gesehen schon geschäftlich Wert bringend genutzt werden können. Der Product Owner, so ist es u.a. in (Liesegang, 2015) beschrieben, legt im Rahmen des Sprint Reviews fest, welche Einträge des Product Backlogs den Zustand „done" (entsprechend der Definition of Done[9]) haben.

Ergänzungen & Fachmeinungen zu Dokumentation in agilen Projekten

Anhand der oben genannten Artefakte aus *Scrum* wird deutlich, dass Dokumentationen, die innerhalb Softwareprojekten vorzufinden sind, welche nach klassischen Methoden durchgeführt werden, sich auch gewissermaßen bei agilen Methoden wiederfinden. Einige Dokumentationen finden sich bei agilen Projekten jedoch nicht explizit wieder. Nach (Liesegang, 2015) verpflichtet *Scrum* beispielsweise implizit zu einer Projektmanagementdokumentation, entsprechend der Transparenz-Definition. Allerdings finden sich hier gewisse Projektmanagementprozesse, wie in (Liesegang, 2015) erklärt, im *Scrum Guide* nicht wieder, darunter z.B. eine Bewertung der Machbarkeit eines Projekts. Einige Bestandteile einer Projektmanagementdokumentation nach der DIN 69901-2, wie u.a. die Erstellung eines Ressourcen- oder Terminplans, geschehen bei *Scrum* nach (Liesegang, 2015) jedoch anhand Schätzungen, welche auf den Einträgen aus dem Product Backlog basieren. Bezüglich der Erstellung eines Projektstrukturplans gilt: „Das Product Backlog und das Sprint Backlog verkörpern eine hierarchische Zerlegung des Liefer- und Leistungsumfangs und stellen somit einen einfachen Projektstrukturplan dar" (Liesegang, 2015). Außerdem erklärt Liesegang, dass bei *Scrum* beispielsweise zu dokumentieren ist, wie die Definition of Done[9] strukturiert sein soll, wie diese zu interpretieren ist, ob und in welcher Form ein explizites Risikomanagement im Projekt betrieben werden soll sowie mit welcher Methode im Projekt geschätzt wird.

Eine Anforderungsdokumentation erfolgt bei *Scrum* gewissermaßen im *Product Backlog*, in welchem Anforderungen an die Software (z.B. in Form von User Stories) festgehalten sind. Allerdings stellen User Stories häufig nur punktuelle Beschreibungen eines gewünschten Systemverhaltens ohne Anspruch auf Vollständigkeit dar, wie in (Hoppen, 2015) erwähnt und sind somit als alleiniges Mittel zur Definition von Anforderungen bei komplexeren Projekten nicht ausreichend. Zudem wird diesbezüglich in (Koch, Agile Softwareentwicklung - Dokumentation, Qualitätssicherung und Kundenmitwirkung, 2010) angemerkt, dass bei agiler Entwicklung die Erstellung eines Pflichtenhefts eher durch eine gemeinsame Abstimmung über die Ausgestaltung der Software zwischen AG und AN geschehen soll.

Prinzipiell scheint hinsichtlich Dokumentation bei agilen Projekten der in (Hruschka, Rupp, & Starke, 2009) definierte Grundsatz, nämlich so wenig wie möglich und so viel wie nötig zu dokumentieren, als sinnvoll. Zudem merken die Autoren an, dass eine Dokumentation möglichst frei von Redundanz sein sollte. Oder auch wie in (Schreiber-Ehle, 2015) ausgedrückt: „Letztlich ist für eine gute Dokumentation nicht die Menge der Dokumente wichtig, sondern dass man für den Zweck angemessen die richtige Dokumentation anfertigt".

[9] https://www.scrum-academy.de/product-owner/wissen/definition-of-done-simpel-und-doch-komplex/

Anforderungsdokumentation vorteilhaft

Wie in (Hoppen, 2015) angemerkt, sind Backlogs oftmals nur ein Sammelsorium inkonsistenter und unvollständiger Anforderungen und reichen somit alleine nicht für eine gute Anforderungsdokumentation aus. Die Autoren in (Bednarczyk & Queins, 2013) nehmen ebenfalls Bezug auf den Umfang einer Anforderungsdokumentation. Fachliches Wissen der Entwickler, die Komplexität des Systems, fachliche Abhängigkeiten sowie gesetzliche und vertragliche Vorgaben bestimmen nach den Autoren den Umfang und die Detaillierung einer Anforderungsdokumentation. Des Weiteren führen diese aus, dass das Anfertigen einer zusätzlich erstellten Anforderungsdokumentation, welche ergänzend (z.B. im Vorfeld) zu den „Standard"-Dokumenten aus der *Scrum*-Entwicklung, also den Backlogs, angefertigt wurde, vorteilhaft sein kann. Unter anderem wird dadurch eine stabilere Architektur, eine fundierte Basis für das Testen und eine Entlastung der Entwickler begünstigt. Entwickler haben so auch die Möglichkeit, Anforderungen an die aktuell zu entwickelnden Module innerhalb eines Sprints im Kontext der Anforderungen an das Gesamtsystem zu betrachten. Zudem können nach (Bednarczyk & Queins, 2013) durch eine vorab vorhandene Anforderungsdokumentation im Vorfeld auch konkrete Testfälle abgeleitet werden. Dies hilft den Entwicklern, da diese dann ihre entwickelten Teilmodule nicht nur gegen sich selbst bzw. gegen deren Anforderungen gewissermaßen isoliert, sondern eben gegen Anforderungen auf allgemeinerer Ebene testen können. Auch in (Hoppen, 2015) wird das Anfertigen einer umfassenden Anforderungsdokumentation bei agilen Projekten befürwortet. Wie der Autor erwähnt, wird bei agilen Projekten zwar vordergründig auf Pflichtenhefte verzichtet, jedoch entstehen diese parallel zur Entwicklung. Die bereits weiter oben in dieser Arbeit beschriebenen Punkte zur Anforderungsdokumentation, welche in (Hoppen, 2015) erklärt sind, sind auch auf agile Projekte abzubilden: „Insofern gelten die [...] genannten Kriterien für Leistungsbeschreibungen durchaus auch in agilen Projekten" (Hoppen, 2015). Ergänzend hierzu führt Hoppen aus, dass die (bereits oben) aufgezeigten Anforderungen an die zu erstellende Software-Herstellungsdokumentationen und Software-Projektdokumentationen unabhängig vom gewählten Vorgehensmodell zu sehen sind.

Probleme bei fehlender oder zu geringer Dokumentation

Dass das Thema Dokumentation bei agilen Projekten oft zu kurz kommt, wird u.a. in (Bednarczyk & Queins, 2013) verdeutlicht. Die Autoren führen aus, dass in der Praxis solch eine Vernachlässigung nicht selten damit begründet wird, dass laut dem agilen Manifest funktionierende Software höher zu bewerten sei als eine umfassende Dokumentation. Dies birgt jedoch einige Risiken in sich. Denn wie die Autoren verdeutlichen, fehlt bei nicht vorhandener Dokumentation später jegliche Basis für die Wartung und Weiterentwicklung der Software. Fehlt nach (Bednarczyk & Queins, 2013) beispielsweise eine Dokumentation von Funktionalitäten und Schnittstellen, ist im Falle einer Weiterentwicklung der Software ein Reverse-Engineering erforderlich, was im schlimmsten Falle sogar Jahre oder Jahrzehnte in Anspruch nehmen kann. Nach (Stiemerling, 2011) muss im Falle einer kompletten Softwareüberlassung die Dokumentation grundsätzlich so gestaltet sein, dass ein neuer Entwickler ohne weitere Rückfragen eine neue Version der Software und aller zugehörigen Dokumente produzieren kann. Dies erfordert dann den höchsten Grad an zu überlassender Dokumentation samt Quellcodedokumentation, einer Anleitung zur Übersetzung und zum automatisierten Testen des Quellcodes im ablauffähigen Programm sowie allen Anforderungs- und Entwurfsdokumentationen.

Eine unzureichende Dokumentation in Softwareprojekten kann nach (Schreiber-Ehle, 2015) zu Mängeln führen, die einen großen wirtschaftlichen Schaden verursachen, ja gar je nach Einsatzgebiet auch menschliches Leben bedroht sein kann.

Vertragliche Leistungsvereinbarung

Um einen Interessenkonflikt zwischen AG und AN also im Vorhinein zu vermeiden, empfiehlt es sich, Leistungen, die innerhalb eines Softwareprojektes gefordert sind, vertraglich festzuhalten. Wie (Schreiber-Ehle, 2015) ausführt, sollte festgelegt werden, welche Dokumentation von welcher Qualität wann abgegeben werden muss, da eine nachvertragliche Dokumentationserstellung in vielen Fällen genauso aufwendig sein kann wie eine Neuerstellung der Software. Des Weiteren sollte darauf geachtet werden, dass die Leistungsbeschreibung möglichst detailliert und unmissverständlich gehalten wird. Denn je unbestimmter ein Leistungsbild vertragstechnisch gestaltet bleibt, wie in (Koch, Agile Softwareentwicklung - Dokumentation, Qualitätssicherung und Kundenmitwirkung, 2010) erwähnt, desto schwerer wird es zum einen für den AG, die Kontrolle über die Leistungserfüllung durch den AN zu haben. In (Hoppen, 2015) ist außerdem ausgeführt, dass vor dem Abschluss eines Vertrages vorab eine umfassende Anforderungsspezifikation vorliegen sollte und erst dann der Vertrag samt Pflichtenheft, oder auch dem AN-seitigen Umsetzungskonzept, sowie der Rolle des Pflichtenhefts (Werk- vs. Dienstleistung) festgelegt werden sollte. Vor dem Hintergrund beispielsweise, dass wie in (Stiemerling, 2011) erwähnt die Zusammenarbeit zwischen AG und AN innerhalb eines agilen Projektes in erster Linie auf gegenseitigem Vertrauen basiert, sollte im Projektvertrag zumindest eine Dokumentation des Quellcodes sowie der Schnittstellen während des Projektes vereinbart werden, da dies eine spätere Einarbeitung neuer Entwickler erleichtert. Wie Stiemerling anmerkt, lässt sich bezüglich Quellcode ein gewisses Niveau von Quellcodedokumentation auch in einem agilen Projekt umsetzen. Diese Qualität der Quellcodedokumentation kann auch als Anforderung vertraglich vereinbart werden.

Dokumentationspflicht nach Rechtslage

Ein Software-Lieferant bzw. AN schuldet, wie es auch mehrheitlich der Literatur, unter anderem (Beckmann, 1998) zu entnehmen ist, dem Kunden bzw. AG, sofern nicht explizit vertraglich anders vereinbart, grundsätzlich ein Handbuch zur Anwendung (Bedienungsanleitung) der Software. Dies wird dort als Hauptpflicht des AN beschrieben, dieser muss allerdings lediglich die Dokumentation in Form von schriftlichen Unterlagen überlassen. Dem amtlichen Leitsatz des zugehörigen BGH-Urteils aus dem Jahre 1993[10], auf das Beckmann in seinem Artikel verweist, ist Folgendes zu entnehmen: „Eine Werkleistung, die die Herstellung von Software zum Gegenstand hat, ist nicht vollendet und damit nicht vollständig erbracht, solange die Aushändigung des dazu gehörenden Handbuches an den Besteller noch aussteht“ (BGH, 14.07.1993 - VIII ZR 147/92). Unmissverständlich ist hierbei oben bereits Genanntes. Einer Software ist also in jedem Fall (falls vertraglich nicht anders vereinbart) eine Anwenderdokumentation/Benutzerhandbuch mitzuliefern. Wie in (Koch, IT-Projektrecht, 2007) erwähnt, hat der AN eine Hauptleistungspflicht teilweise nichterfüllt, fehlt bei der Übergabe die geschuldete Dokumentation (insbesondere Anwenderdokumentation/Bedienerhandbuch). Außerdem, so ist es (Koch, Agile

[10] https://www.jurion.de/urteile/bgh/1993-07-14/viii-zr-147_92/

Softwareentwicklung - Dokumentation, Qualitätssicherung und Kundenmitwirkung, 2010) zu entnehmen, schuldet ein AN dem AG, sofern die entwickelte Software dem AG komplett übergeben wird, dieser also anschließend auch für die Pflege und Wartung zuständig ist, zusätzlich eine Entwicklungsdokumentation. Diese ist, so nach (Koch, IT-Projektrecht, 2007) in diesem Fall auch ohne besondere Vereinbarung geschuldet. In (Koch, Computer-Vertragsrecht, 2009) bezeichnet der Autor die Dokumentationserstellung als Teil der Verpflichtung zur Softwareerstellung und der AN ist bezüglich der Lieferung einer Dokumentation vorleistungspflichtig (im Werkvertragsrecht). Außerdem führt der Autor weiter aus, dass nach DIN 66 230 Software begrifflich aus „Programm“ und „Dokumentation“ besteht.

Bei agilen Methoden ist dies im Prinzip genauso. Nur, weil agile Prinzipien Dokumentation keiner schreiben als klassische Methoden, ist dies kein Freifahrtschein für Softwarehersteller, auf Dokumentation zu verzichten. Verdeutlicht wird das auch in (Koch, Agile Softwareentwicklung - Dokumentation, Qualitätssicherung und Kundenmitwirkung, 2010). Koch schreibt dazu, dass ein AN dem AG bei Verzicht auf bestimmte Dokumentationen ausdrücklich klarmachen muss, dass die Dokumentationserstellung nicht zur gewählten Leistungsform gehört. Hierfür liegt dann die Beweislast beim AN. Das heißt, ohne eine vertragliche Regelung wird der AN im Falle einer rechtlichen Auseinandersetzung sicherlich schnell in Schwierigkeiten geraten. Wie in (Koch, Agile Softwareentwicklung - Dokumentation, Qualitätssicherung und Kundenmitwirkung, 2010) weiter ausgeführt, ist eine vertragliche Vereinbarung eines Verzichts auf Dokumentation zwischen AG und AN denkbar. Aus Sicht des AN ist es hier wichtig, genauestens zu konkretisieren, auf welche Dokumentationen verzichtet wird. Denn im Zweifel trägt nach Koch der AN das Risiko, sich nicht voll von seiner Dokumentationspflicht entlasten zu können. Ohnehin scheint, wie in (Koch, Agile Softwareentwicklung - Dokumentation, Qualitätssicherung und Kundenmitwirkung, 2010) angemerkt, eine vertragliche Regelung auf Verzicht von Dokumentationen (weil dadurch z.B. ein schnelleres Entwickeln angeboten werden kann) nicht vollkommen risikofrei für einen AN zu sein. Denn nach § 307 Abs. 2 Nr. 2 BGB[11] kann im Zweifel bei einem Verzicht auf Dokumentation - trotz vertraglicher Regelung innerhalb der Leistungsbeschreibung - auch eine Benachteiligung des Kunden festgestellt werden, da wesentliche Rechte oder Pflichten, die sich aus der Natur des Vertrags ergeben, so eingeschränkt sind, dass die Erreichung des Vertragszwecks gefährdet ist. Des Weiteren wird ein vollständiger Verzicht auf Dokumentation laut Koch auch von energischen *XP*-Anhängern als nicht sachgemäß angesehen, betrachten diese bei Entwicklung und Test einer Komponente die Dokumentation als doch „selbstverständlich“ verpflichtend. Koch erklärt außerdem, dass ein vollständiger Verzicht auf Dokumentation das Übermaßverbot[12] verletzen würde. Geboten bei der Reduzierung von Dokumentation ist demnach eine gewisse Verhältnismäßigkeit. Dies bestätigt auch ein in (Hruschka, Rupp, & Starke, 2009) formulierter Grundsatz zur Dokumentation in agilen IT-Projekten: „So wenig Dokumentation wie möglich, aber so viel wie nötig“. Rechtliche Folgen für den Fall, dass diese der übergebenen Software nicht beiliegt, sind in (Beckmann, 1998) genannt: „Wird dem Benutzer eine völlig unvollständige, unverständliche, unübersichtliche oder fremdsprachliche Programmdokumentation geliefert, ist von Nichterfüllung auszugehen, wenn die Fehler so schwerwiegend sind, daß sie die Gesamtlieferung völlig unbrauchbar machen, also eine untaugliche Leistungsbewirkung

[11] https://www.gesetze-im-internet.de/bgb/__307.html

[12] http://www.rechtslexikon.net/d/übermassverbot/übermassverbot.htm

vorliegt, die trotz der Übergabe der Dokumentation die Rechte aus § 326 BGB wegen Nichterfüllung eröffnet.“ Die Beweislast für die Vollständigkeit der Dokumentation liegt wie in (Beckmann, 1998) des Weiteren erwähnt, nach wie vor beim AN. Wie in (Liesegang, 2015) angemerkt gibt es bei *Scrum* aufgrund der Definition zur Transparenz aus dem *Scrum Guide* durchaus eine erwähnenswerte Pflicht zur Dokumentation.

Rechtsfall aus 2016

Das LG Wiesbaden hatte 2016 einen Rechtsstreit zu klären, bei dem ein AN vom AG eine Zahlung der von ihm getätigten Teilleistung nach Abbruch des Software-Projekts, welches nach dem agilen Vorgehensmodell *Scrum* durchgeführt wurde, einforderte. Wie in (Software, 2017) im Auszug aus dem Urteil erwähnt, hat das Landgericht die Klage des AN im Urteil vom 30.11.2016 abgelehnt. Als Begründung für den gescheiterten Vergeltungsanspruch des AN, so in (Software, 2017) beschrieben, sah das LG nicht die Tatsache, dass das Projekt vorzeitig abgebrochen wurde und somit keine vollständige Leistung vom AN erbracht werden konnte. Grund für die Ablehnung der Klage war eine nicht hinreichende Dokumentation seitens des AN, welche die erbrachten Teilleistungen unbrauchbar für den AG machte. Die unzureichende Dokumentation, wie in (Software, 2017) weiter ausgeführt, wurde durch einen IT-Sachverständigen festgestellt. Zwar lag eine vom AN erstellte Java-Doc Quellcodedokumentation vor, dieser bemängelte jedoch eine fehlende Dokumentation der Systemarchitektur und Systemkomponenten. Nichtsdestotrotz wurde vom OLG Frankfurt am 17.08.2017 das Urteil abgeändert. Das OLG hat in einem neuen Prozess entschieden, den AG zu einer Zahlung von 155.854,30 € nebst Zinsen in Höhe von 8 Prozentpunkten über dem Basiszinssatz an den AN verurteilt.

Ersichtlich wird anhand des eben genannten Rechtsfalls, dass softwarerelevante Themen gerichtlich sehr differenziert betrachtet werden können. Das Vorgehensmodell, aber auch Dokumentation spielen hierbei eine wichtige Rolle. So kann eine fehlende Dokumentation einer Partei negativ zu Lasten gelegt werden, andererseits stellt sich hier die Frage der Fälligkeit. Denn wie in (Schneider, 2017) beschrieben, war für das OLG Frankfurt nicht ersichtlich, dass eine Dokumentation der Systemarchitektur bereits zum Zeitpunkt des Projektabbruchs geschuldet war. Wie Schneider weiter ausführt, ist eine solche erst sinnvoll, wenn die letztendlich verwendeten Komponenten des Systems feststünden.

Zusammenfassung und Fazit

Alles in allem lässt sich festhalten, dass sich zum Thema Dokumentation bei Software-Projekten vielerlei Arten von Dokumentationen in der Literatur finden lassen und das Thema, wie in (Stiemerling, 2011) erklärt, ein weites Feld ist. Ein gewisser Konsens in der Fachliteratur herrscht über den Zweck der Dokumentation, die mit der übergebenen Software mitgeliefert wird. Nämlich, dass durch eine mitgelieferte Dokumentation der vertraglich vereinbarte Nutzungszweck, wie u.a. in (Stiemerling, 2011) erwähnt, ermöglicht wird. Dies entspricht auch der BGH-Rechtsprechung[13] aus dem Jahre 2003.
Sofern vertraglich nicht anders vereinbart, schuldet der AN dem AG grundsätzlich ein Handbuch zur Anwendung der Software. Wenn der AN die Software komplett übergibt und auch die Pflege der Software in Zukunft der AG übernimmt, ist auch eine

[13] https://www.jurion.de/urteile/bgh/2003-12-16/x-zr-129_01/

Entwicklungsdokumentation geschuldet. Grundsätzlich sollten geschuldete Dokumentation vertraglich festgehalten werden. Wie auch in (Stiemerling, 2011) erwähnt, ergeben sich in einem typischen IT-Vertrag die Dokumentationspflichten aus der Summe der einzelnen Leistungen, welche vorab festgelegt werden. Der AG muss die Software mithilfe der Dokumentation zur Ausübung des bestimmungsgemäßen Gebrauchs nutzen können. Entsprechend verschiedener Interessenlagen der Adressaten (z.B. künftiger Architekt, Entwickler usw.) sind nach (Schreiber-Ehle, 2015) zudem die Dokumentationen zu erstellen. Bezüglich Umfang und Detailgrad einer Dokumentation sollte eine gewisse Verhältnismäßigkeit eingehalten werden.

Bei agilen Methoden ergibt sich durch die Verwendung entsprechender methodenspezifischer Werkzeuge wie z.B. bei *Scrum* den *Product Backlogs, Sprint Backlogs* oder *Inkrementen* die Möglichkeit, Dokumentation zu erstellen. Beim *XP* liegt der Fokus noch deutlich stärker auf der Entwicklung. Wie in (Koch, Agile Softwareentwicklung - Dokumentation, Qualitätssicherung und Kundenmitwirkung, 2010) erwähnt, kann hier der Kunde nur Quellcodedokumentation und Unit Tests als Dokumentation erwarten. Eine Anforderungsdokumentation ist auch bei agilen Methoden meist unabdingbar für eine funktionierende Projektdurchführung. User Stories sind in der agilen Entwicklung ein bewährtes Mittel zur Projektsteuerung, reichen aber nach (Bednarczyk & Queins, 2013) als alleinige Dokumentationsform nicht. Anforderungen möglichst früh in einer umfassenden Dokumentation hat in der Regel somit Vorteile.

Anhand des in dieser Arbeit aufgezeigten Rechtsstreits, welcher vor dem LG Wiesbaden und OLG Frankfurt geführt wurde, zeigt sich, wie unterschiedlich das Fehlen von Dokumentation im Zusammenhang mit dem Vorgehensmodell (hier *Scrum*) bewertet werden kann. Wie in (Schneider, 2017) angemerkt, wurde durch das Urteil vom OLG Frankfurt jedoch nicht die Fälligkeit einer Dokumentation der Systemarchitektur mitentschieden. Auch hier wird deutlich, wie wichtig es ist, zwischen AG und AN den Umfang und Zeitpunkt der zu liefernden Dokumentation vertraglich festzuhalten, um möglichen Rechtsstreits aus dem Weg zu gehen.

Literaturverzeichnis

Beckmann, H. (1998). EDV-Anwenderdokumentation. *Computer und Recht*, S. 519-523.

Bednarczyk, M., & Queins, S. (10 2013). Dokumentation in agilen Projekten – so geht's. *ProjektMagazin*.

Herzog, J., Holzmüller, G., & Schoepe, W. (2010). Agile Softwareentwicklung im Kontext unternehmensweiter IT-Prozesse. *OBJEKTspektrum*, S. 6-7.

Hoppen, P. (15. 11 2015). Software-Anforderungsdokumentation. *Computer und Recht*, S. 747-760.

Hruschka, P., Rupp, C., & Starke, G. (2009). *Agility kompakt.* Heidelberg: Spektrum Akademischer Verlag.

Ihde, R. (1999). Das Pflichtenheft beim Softwareerstellungsvertrag. *Juris*, S. 409-141.

Juhl, D. (2015). *Technische Dokumentation.* Berlin: Springer Berlin Heidelberg.

Koch, F. (2007). *IT-Projektrecht.* Berlin: Springer-Verlag (Berlin Heidelberg New York).

Koch, F. (2009). *Computer-Vertragsrecht.* Planegg, München: Haufe-Lexware.

Koch, F. (2010). Agile Softwareentwicklung - Dokumentation, Qualitätssicherung und Kundenmitwirkung. *ITRB*, S. 114-119.

Liesegang, W. (15. 08 2015). Projektmanagement und die zugehörige Dokumentation. *Computer und Recht*, S. 541-556.

Nowak, H. (12 2002). *Infrasoft.* Abgerufen am 12. 01 2019 von Infrasoft: http://www.infrasoft.at/downloads/Dokumentation_in_der_Softwareentwicklung.pdf

Riedl, S. (14. 01 2018). *IT-Business*. Abgerufen am 12. 01 2019 von IT-Business: https://www.it-business.de/was-ist-eine-schnittstelle-a-714831/

Roth, B., & Dorschel, J. (2008). Das Pflichtenheft in der IT-Vertragsgestaltung. *ITRB*, S. 189-191.

Schneider, J. (12. 09 2017). *CRonline*. Abgerufen am 20. 01 2019 von CRonline: https://www.cr-online.de/blog/2017/09/12/neue-dimensionen-des-softwareentwicklungsvertrages/

Schreiber-Ehle, S. (7 2015). Dokumentation in Softwareerstellungsverträgen. *CReport*, S. 469-481.

Software, O. F.-V. (04. 10 2017). *Dr. Damm & Partner* . Abgerufen am 16. 01 2019 von Dr. Damm & Partner : http://www.damm-it-recht.de/olg-frankfurt-a-m-zur-abnahme-von-agil-nach-dem-scrum-verfahren-entwickelter-software/

Stiemerling, O. (2011). Dokumentation von Software. *ITRB*, S. 286-290.