Eigenschaften optimierter API-Dokumentationen im Entwicklungsprozess sicherer Software

Rolf Huesmann
Hochschule Darmstadt
Darmstadt, Deutschland
rolf.huesmann@h-da.de

Alexander Zeier
Hochschule Darmstadt
Darmstadt, Deutschland
alexander.zeier@h-da.de

Andreas Heinemann
Hochschule Darmstadt
Darmstadt, Deutschland
andreas.heinemann@h-da.de

ZUSAMMENFASSUNG

Softwareentwicklern fällt es nicht leicht, die vom jeweiligen Hersteller herausgegebenen offiziellen Dokumentationen von Sicherheits-APIs zu lesen. Hingegen bevorzugen sie informelle Dokumentationen, wie sie zum Beispiel auf Q&A Plattformen wie Stack Overflow zu finden sind. Allerdings enthalten Codebeispiele solcher Quellen des Internets oft falschen bzw. aus Sicht der IT-Sicherheit unsicheren Code, welcher oft unreflektiert übernommen wird und letztendlich zu unsicheren Applikationen führt.

Diese Arbeit untersucht, welche Eigenschaften eine Dokumentation aus Sicht der Entwickler haben sollte, um sie bei der Erstellung von sicherem Code zu unterstützen. Dazu wurden insgesamt 26 Programmierer in sieben Fokusgruppen über die Vor- und Nachteile beispielhafter, gängiger Dokumentationsarten von APIs befragt. Abschließend wurden die Teilnehmer gebeten, Eigenschaften einer aus ihrer Sicht optimierten API-Dokumentation zu beschreiben.

Herausgestellt hat sich, dass eine optimierte Dokumentation viele Beispiele haben sollte. Sie sollte gut strukturiert, durchsuchbar und bei Google leicht auffindbar sein. Weitere wichtige Funktionen wären: Ein Bereich mit Tutorials, eine klassische Referenz und Videos für den schnellen Überblick. Zu jedem Themenbereich sollte es die Möglichkeit für Fragen, Antworten und Diskussionen geben. Durch ein Bewertungssystem sollten gute Lösungen aus der Diskussion in die Beispiele einfließen können.

Weiter konnte anhand der Antworten der Probanden ein Interaktionsmuster identifiziert werden, welches das Zusammenspiel der einzelnen Teile einer optimierten API-Dokumentation beschreibt.

Auf Basis dieser Erkenntnisse soll in einem nächsten Schritt ein Prototyp entwickelt und seine Wirksamkeit hinsichtlich der Unterstützung im Softwareentwicklungsprozess von sicherheitskritischen Anwendungen evaluiert werden.

CCS CONCEPTS

• **Human-centered computing** → **Empirical studies in HCI**; • **Security and privacy** → *Usability in security and privacy*.

KEYWORDS

API Dokumentation, Benutzbarkeit, Fokusgruppen, Software-Entwicklungsprozess

1 EINLEITUNG UND MOTIVATION

Entwicklern fällt es schwer, kryptografische Algorithmen korrekt zu verwenden (vgl. [13]). Sie wünschen sich mehr Code-Beispiele, welche häufige Anwendungsfälle abdecken. Hinzu kommt, dass Entwickler API Dokumentationen nicht sorgfältig lesen oder nicht verstehen (vgl. [18, 19]). Vielmehr weichen die Entwickler auf alternative Informationsquellen wie Stack Overflow aus (vgl. [1]). Weiter haben Fischer et al. [7] gezeigt, dass ca. 15, 4% der 1, 3 Millionen untersuchten Apps im Google Appstore sicherheitsrelevante Code-Beispiele aus Stack Overflow enthalten. Von diesen Code-Beispielen wurden 97, 9% durch die Autoren als unsicher eingestuft. Jedoch enthalten auch „offizielle“ Informationsquellen die vom Entwickler benötigten Informationen nicht immer oder nicht in dem benötigten bzw. gewünschten Format (vgl. [2]). Mit der Weiterentwicklung des Quanten-Computers werden neue kryptographische Verfahren und zugehörige APIs aufkommen. Aktuell organisiert hierzu das NIST einen Wettbewerb zur Standardisierung[1]. Es ist abzusehen, dass die Quanten-Computer resistenten Verfahren für einen Entwickler weniger zugänglich und die zugehörigen APIs komplexer ausfallen werden. Somit sollte deren API-Dokumentation besser gestaltet werden, damit Entwickler diese leichter lesen und verstehen, um letztendlich Fehler in sicherheitsrelevantem Code zu minimieren.

Ziel der vorliegenden Studie ist es, mithilfe von Fokusgruppen herauszufinden, welche Eigenschaften eine API-Dokumentation mitbringen sollte, um Softwareentwickler

[1] https://csrc.nist.gov/Projects/Post-Quantum-Cryptography. Zuletzt aufgerufen am 14.06.2019.

MuC'19 Workshops, Hamburg, Deutschland

https://doi.org/10.18420/muc2019-ws-302-03

im Entwicklungsprozess sicherer Software besser zu unterstützen.

Zunächst wird im zweiten Abschnitt kurz auf verwandte Arbeiten eingegangen. Anschließend wird das Vorgehen und die zugrundeliegende Methodik (Abschnitt 3) der durchgeführten Studie erläutert. Im Anschluss werden die Ergebnisse vorgestellt (Abschnitt 4) und bewertet (Abschnitt 5). Abschnitt 6 fasst die wesentlichen Erkenntnisse zusammen und gibt einen Ausblick auf die nächsten Schritte.

2 VERWANDTE ARBEITEN

In der Literatur finden sich wenige Arbeiten, die sich auf die Verbesserung der API-Dokumentation von Sicherheitsfunktionen fokussieren.

Mindermann und Wagner [12] haben eine Webseite[2] entwickelt, auf der sichere Codebeispiele veröffentlicht werden. Sie soll den Nachteil, dass API-Dokumentationen eines Herstellers nur schwer zu ändern sind, ausgleichen und Entwicklern eine zusätzliche Quelle für sichere Code-Beispiele bieten.

Dekel und Herbsleb [5] haben in Ihrer Arbeit den Ansatz verfolgt, die wichtigsten Informationen in der Entwicklungsumgebung des Entwicklers hervorzuheben und diese dadurch zur weiteren Recherche anzustoßen. Die beiden Arbeiten betrachten jeweils einen Teilaspekt der API-Dokumentations-Landschaft.

Die vorliegende Studie ist im wesentlichen von der Arbeit von Novick und Ward [14] inspiriert. Dort wird nach ganzheitlichen Gründen gesucht, welche Eigenschaften einer API-Dokumentation bei der Erstellung von sicherem Code unterstützen. Novick und Ward schlagen vor, dass eine Dokumentation einfach zu navigieren, Probleme anhand von Beispielen zu lösen, Beschreibungen vollständig und korrekt sein sollten und das Entwickler verschiedene Niveaus der Dokumentation brauchen. Plattformen wie Stack Overflow sind erst später entstanden und konnten daher von Novick und Ward nicht berücksichtigt werden. Diese Arbeit überprüft die Ergebnisse von Novick und Ward unter Einbeziehung der derzeit genutzten und populären Dokumentationsarten.

Ein komplementärer Ansatz ist es, die API selbst um sicherheitsrelevante Fehlermeldungen zu erweitern. Groski et al. [8] haben eine kryptografische Python API um solche Fehlermeldungen erweitert, sodass beispielsweise ein Entwickler bei der Nutzung eines unsicheren Algorithmus über sichere Alternativen aufgeklärt wird. Da es sich hierbei um einen neuen Ansatz handelt, wurde diese Dokumentationsart jedoch in dieser Arbeit nicht weiter betrachtet.

[2] http://CryptoExamples.com. Zuletzt aufgerufen am 14.06.2019.

3 METHODIK

Aufgrund der sehr offenen Fragestellung werden Fokusgruppen durchgeführt, wie es für explorative Studien dieser Art üblich ist [16]. Die Gruppendynamik soll hierbei eine angeregte Diskussion zwischen den Teilnehmern begünstigen, wodurch die Teilnehmer in der Regel engagierter und ehrlicher antworten [10]. Die Fragestellung kann in das von Dürrenberger und Behringer definierte Themenfeld der „Testverfahren" eingeordnet werden [6]. Eine Grenze der Methode zeigen Przyborski und Riegler [15] auf. Sie schreiben, dass sich das Verfahren nicht zur Bearbeitung von Fragen, bei denen das individuelle Handeln, die individuelle Haltung oder Entscheidungsprozesse der Teilnehmer die zu untersuchenden Komponente darstellen. Dementsprechend können die ersten im folgenden Abschnitt beschriebenen Fragen kritisch gesehen werden. Allerdings bereiten die Fragen auf die letzte Frage nach der „perfekten" Dokumentation vor und lassen den gruppendynamischen Prozess starten, welcher zum Ende hin mit seinen Vorteilen überwiegt.

Studiendesign und Durchführung

Durch ein Brainstorming wurden fünf Dokumentationsarten identifiziert: API Spezifikation des Herstellers, Blog, textuelles Tutorial, Video-Tutorial und Frage-und-Antwort-Plattform (Q&A Plattform). Mailing-Listen und Foren wurden als Q&A Plattform ähnlich definiert und daher nicht extra aufgeführt. Es wurde sich explizit auf Dokumentationen beschränkt, die online verfügbar sind. Die jeweils verwendeten Bespiele sind im Anhang A angegeben.

Die Zielgruppe der Fokusgruppen sind Programmierer, da diese regelmäßig mit Dokumentationen verschiedener Art arbeiten. Diese wurden in Informatik-Vorlesungen einer Hochschule, aus Unternehmen und aus dem persönlichem Umfeld der Autoren eingeladen an der Studie teilzunehmen. Der Einladung sind 26 Probanden gefolgt. Die Teilnahme war freiwillig und mit keinerlei Nachteilen für die Probanden verbunden. An den Fokusgruppe nahmen jeweils zwei bis fünf Teilnehmer teil. Für jede Befragung wurde ein Zeitlimit von ca. einer Stunde festgelegt. Sie wurde nach Zustimmung der Teilnehmer auditiv aufgenommen und in Stichworten mitgeschrieben, damit später eine Zuordnung der Teilnehmer zu ihren Aussagen möglich war. Die Aufzeichnungen wurden transkribiert und anschließend gelöscht, um datenschutzrechtlichen Aspekten gerecht zu werden.

Der Interviewer hat die Diskussion auf Basis der „reflexiven Prinzipien der Initiierung und Leitung von Gruppendiskussionen" nach Przyborski und Riegler geführt (vgl. [15]). Zu Beginn wurden die Teilnehmer begrüßt, organisatorische Einzelheiten besprochen und von jedem in Erfahrung gebracht, wie lange dieser bereits programmiert. Zur Auflockerung sollten die Programmierer berichten, welche Quellen

sie zuletzt genutzt haben und wie zufrieden sie damit waren. Eine der genannte Dokumentationsart wurde anschließend aufgegriffen. Ein Beispiel wurde dazu über einen Beamer an die Wand projiziert und von den Teilnehmern erfragt, welche Vor- und Nachteile die entsprechende Dokumentationsart hat. Dies wurde mit den Dokumentationsarten API Spezifikation, Blog, textuelles Tutorial, Video-Tutorial und einer Q&A-Plattform durchgespielt. Die Beispiele sollen die Teilnehmer zur Diskussion über die jeweiligen Vor- und Nachteile anregen, um später die eigentliche Frage nach einer „perfekten"[3] Dokumentation beantworten zu können. Im Anhang A ist ein grober Ablauf der Fokusgruppen als Leitfaden inklusive der inhaltlich ähnlichen Fragen dargestellt.

Auswertung

Um die Aufnahmen der Fokusgruppen computerunterstützt auswerten zu können, wurden diese unter datenschutzrechtlichen Grundsätzen anonymisiert und wörtlich transkribiert. Es wurde sich für eine sogenannten wörtlichen Transkription entschieden, da inhaltlich-thematische Aspekte der Analyse im Vordergrund stehen (vgl. Reinders at al. [16]) und eine qualitative Inhaltsanalyse durch kategorisieren (auch als „Coding" bezeichnet [9]) angestrebt wird. Da sich diese Arbeit nicht auf bekannte Aspekte konzentriert, konnten vorher keine Kategorien mit begründeten Auswertungsaspekten festgelegt werden. Deswegen wurde sich zur Kategorisierung über die induktive Kategorienbildung nach Mayring [11] entschieden. Zu jeder im vorherigen Abschnitt genannten Dokumentationsart wurde jeweils eine Hauptkategorie mit den zwei Unterkategorien „Vorteile" und „Nachteile" erstellt. Zusätzlich wurde die Hauptkategorie „Optimierte Dokumentation" erstellt. Einer der Autor hat für jede Kategorie durch die induktive Kategorienbildung weitere Kategorien erstellt und diesen Aussagen der Teilnehmer zugeordnet. Mehrfachnennungen der Teilnehmer wurden nicht gezählt. Anschließend wurden die Texte mit den zugeordneten Kategorien von einem zweiten Autor betrachtet und diskutiert, bis Einigung über alle Aussagen bestand. Daher ist keine Aussage zur Intercoder-Reliabilität nach Reinders et al. [16] möglich.

Limitierungen

Allgemein lässt sich die Güte dieser Art der qualitativen Studie schwer beurteilen. Es herrscht allerdings Konsens darüber, dass Fokusgruppen neben explorativen Informationen verallgemeinerbare Trends liefern (vgl. Schulz [17]). Da Fokusgruppen zu teilweise unterschiedlichen Ergebnissen

[3]Es wurde überspitzt nach der perfekten Dokumentation gefragt. In dieser Arbeit wird nach einer optimierten Dokumentation gesucht, da nicht ausgeschlossen werden kann, dass es eine bessere Dokumentation gibt.

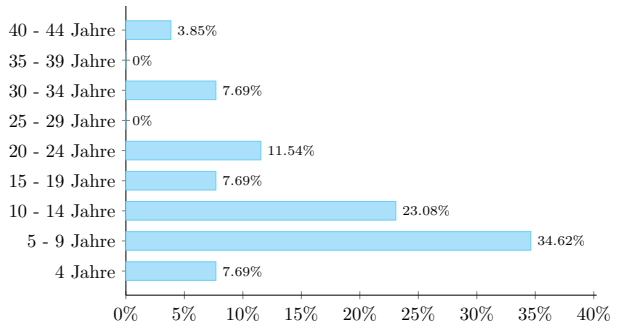


Abbildung 1: Programmiererfahrung der Teilnehmer

führen können, wird das Vorgehen möglichst offen und transparent gestaltet, so dass es nachvollziehbar und kritisierbar bleibt (vgl.[3]).

Reinders at al. [16] führt noch den Punkt der „kommunikativen Validierung" an. Er beschreibt, dass wenn sich die Teilnehmer mit ihren Ansichten, Meinungen oder Haltungen in der vom Forschenden erstellten Interpretation der Ergebnisse wiederfinden, dies ein wichtiger Aspekt der Güte einer Studie sein kann. Dies wird in dieser Studie offen gelassen und bietet einen Punkt für weiterführende Arbeiten.

Somit können die im Rahmen dieser Studie erarbeiteten Ergebnisse bestmöglich als Tendenzen angesehen werden und sollen in weiterführenden Arbeiten weiter untersucht werden.

Die Teilnehmer der Fokusgruppen wurden nicht gezielt nach Merkmalen der Dokumentation sicherheitsrelevanter APIs gefragt, da sich die Konzepte von „normalen" auf solche APIs problemlos übertragen lassen sollten. Eine gute Dokumentation zur Erstellung sicherer Software gewinnt jedoch dadurch weiter an Wichtigkeit, da hier Fehler in der Programmierung weitreichende Folgen haben können. Unserer Erfahrung nach führt das Informatikstudium i.d.R. Studierende nicht ausreichend in sicherheitsrelevante Konzepte wie Kryptografie ein, weshalb insbesondere hier eine gute Unterstützung durch Dokumentationen jeglicher Art notwendig ist.

4 ERGEBNISSE

An den Fokusgruppen haben insgesamt 26 Probanden an sieben Terminen teilgenommen. Ihre durchschnittliche Programmiererfahrung beträgt 13, 92, der Median beträgt 11 Jahre. Die meisten Teilnehmer haben zwischen 5 und 15 Jahren Erfahrung. Die Reichweite ging von Teilnehmern mit 4 bis zu Teilnehmern mit 42 Jahren Programmiererfahrung. In Abbildung 1 ist die Programmiererfahrung detaillierter aufgeschlüsselt.

Abbildung 2 gibt eine Übersicht über die entstandenen Kategorie-Bäume. Innerhalb der Vor- und Nachteile sind

die Unterkategorien nach ihrer Häufigkeit der Zuordnung zu Textpassagen angeordnet. Die Häufigkeit wird dabei in Klammern angegeben. Da die stumme Zustimmung anderer Teilnehmer, beispielsweise durch Gestik oder sogar nur in Gedanken, nicht kodiert werden kann, könnte der tatsächliche Wert hier auch höher liegen. Bei der folgenden Erläuterung der einzelnen Vor- und Nachteile greifen wir daher nur auf vage Begriffe zurück, um eine Tendenz der Häufigkeit darstellen zu können.

API Spezifikation - Vorteile

Referenz. Sehr viele Teilnehmer beschreiben, dass sie die API Spezifikation nutzen, um einen Überblick über eine API zu bekommen. Sie schauen nach Funktionen, deren Parameter oder ob es andere Funktionen gibt, die besser zu ihrem Problem passen. Letzteres setzt voraus, dass sie bereits eine Lösungsstrategie für ihr Problem entwickelt haben.

Richtigkeit. Mehrere Teilnehmer sagten: *„Bei solchen Dokumentationen weiß man, die sind meistens korrekt und es hat sich jemand ein paar mehr Gedanken gemacht"*. Die Teilnehmer gehen außerdem von der Vollständigkeit der Dokumentation aus.

Gedanken des Autors. Einige Teilnehmer haben positiv angemerkt, dass man aus der API Spezifikation die Ideen und Konzepte der Autoren herauslesen könne.

API Spezifikation - Nachteile

Struktur und Aufbau. Die Dokumentation nach den Klassenstrukturen aufzubauen verwirrt viele Teilnehmer, da sie den Pfad zur Klasse oder zur Funktion, die sie suchen, nicht kennen. Besonders, wenn sie sich gerade versuchen in die API einzuarbeiten.

Keine Beispiele. Sehr oft wurde von den Teilnehmern bemängelt, dass es in der API Spezifikation häufig keine Beispiele gibt. Aus Beispielen können sie besser erkennen, wie ein Funktion arbeitet, als mit einer technischen Beschreibung.

Nicht nachvollziehbar. Einige Teilnehmer können mit der Dokumentation wenig anfangen. Unter anderem, weil der Entwickler eventuell ein anders Verständnis für den eigenen Code hat. Für ihn ist klar, wie einzelne Klassen und Funktionen zusammenhängen. Für einen Programmierer, insbesondere wenn dieser neu in der Thematik ist, ist die Dokumentation hingegen oft unverständlich.

Blog - Vorteile

Konkrete Probleme. Als Vorteil an einem Blog wurde in mehreren Gruppen gesagt, dass der Blog zu einem bestimmten Thema ein spezielles Problem behandelt. Wenn dieses zu dem gerade zu lösenden Problem eines Teilnehmers passt, nutzt er diesen Blog-Eintrag.

Informativ. Um sich in ein neues Thema einzuarbeiten oder wenn ein neues Feature aus einer API oder einer Programmiersprache erklärt wird, finden einige Teilnehmer Blogs hilfreich.

Reputation. Ein weiteres Thema, welches bei Blogs angesprochen wurde, ist die Reputation des Autors. Viele Teilnehmer haben beschrieben, dass sie der Hintergrund des jeweiligen Autors interessiert. Ein Autor gewinnt an Vertrauenswürdigkeit, wenn er bereits mehrere Blog-Einträge zu ähnlichen Themen verfasst hat. Auch der berufliche Hintergrund des Autoren spielt dabei eine Rolle. Arbeitet dieser bei einer renommierten Firma oder gilt vielleicht sogar als ein Experte auf dem Gebiet, gewinnen seine Beiträge ebenfalls an Vertrauenswürdigkeit.

Umfassend. Als Vorteil wurde öfter genannt, dass ein Blog-Beitrag ein Thema meistens von Anfang bis Ende beschreibt. Es wird an einem roten Faden das Thema eingeleitet, beschrieben und die Zusammenhänge erklärt.

Blog - Nachteile

Wenig Kommentare. Viele Teilnehmer bemängeln, dass unter den Blog-Einträgen zu wenig bis keine Kommentare von Lesern geschrieben werden. Es könnte sein, dass der Autor in seinem Beitrag einen Fehler veröffentlicht hat oder dass der Beitrag veraltete Funktionen nutzt. Ohne einen entsprechenden Kommentar fallen eventuelle Fehler der Allgemeinheit nicht auf.

Nur ähnliches Problem. Einige Teilnehmer beschreiben das Phänomen, dass der Blog-Eintrag meistens nicht genau das Problem oder Thema beschreibt, welches die Teilnehmer aktuell brauchen. Es tangiert das gesuchte Themengebiet, aber entspricht nicht genau dem Problem, wofür gerade eine Lösung gesucht wird.

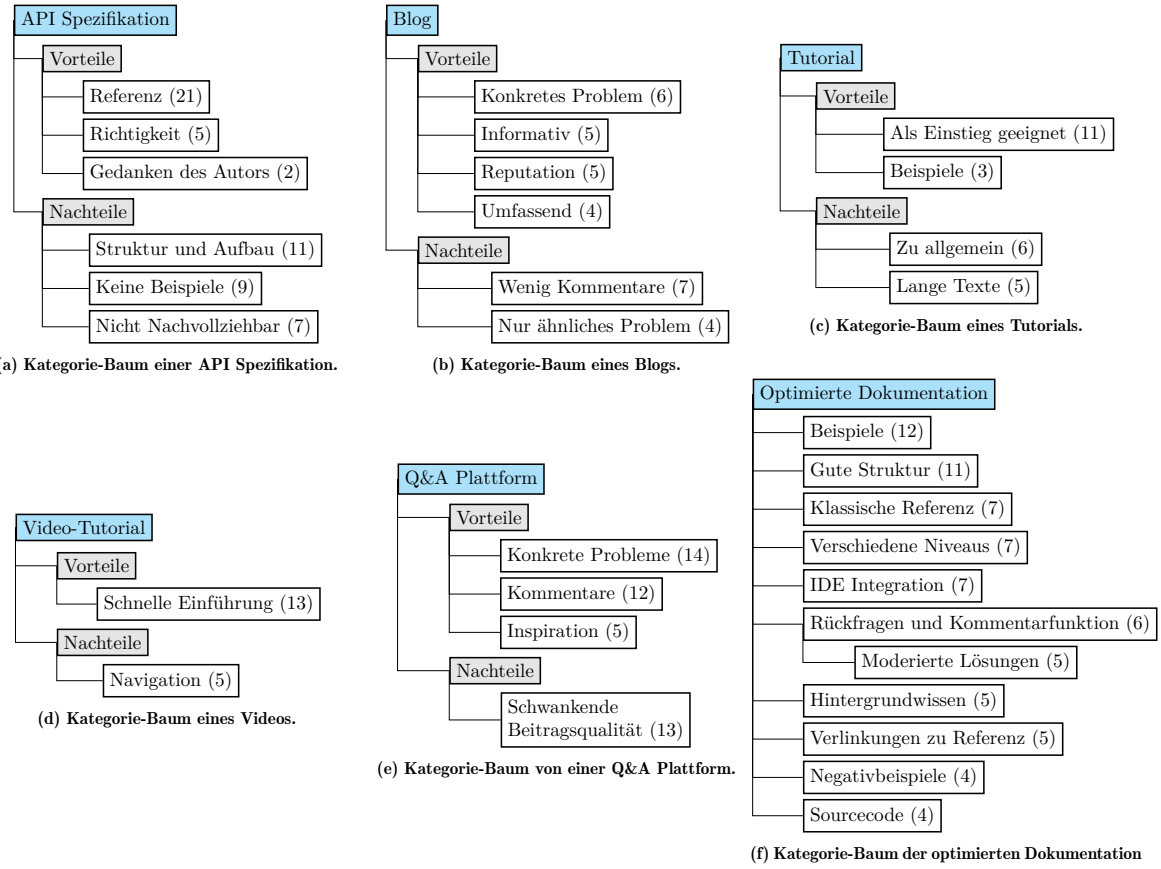
Tutorial - Vorteile

Als Einstieg geeignet. Bei einem textuellen Tutorial waren sich alle Teilnehmer augenscheinlich einig, dass es sich für den Einstieg in ein neues Thema eignet. In anderen Fällen wird jedoch nur selten auf Tutorials zurückgegriffen.

Beispiele. Positiv aufgefallen sind die vielen kleinen Code-Beispiele, welche meistens zwischen den Texten eingepflegt sind. Dazu ein Teilnehmer: *„Das mit dem kurzen Beispiel lässt sich auch ganz gut überfliegen."*

Tutorial - Nachteile

Zu allgemein. Der größte negative Punkt bei Tutorials ist, dass sie keine Beispiele liefern, wie ein konkretes Problem gelöst werden kann. Ein Teilnehmer formuliert das so: *„Man geht halt [als Entwickler] schnell von den Grundlagen weg zu spezifischen Problemen, die man nicht direkt in so was findet."*



(a) Kategorie-Baum einer API Spezifikation.

(b) Kategorie-Baum eines Blogs.

(c) Kategorie-Baum eines Tutorials.

(d) Kategorie-Baum eines Videos.

(e) Kategorie-Baum von einer Q&A Plattform.

(f) Kategorie-Baum der optimierten Dokumentation

Abbildung 2: Die, nach dem in Abschnitt 3 beschriebenen angewandte Verfahren, entstandene Kategorie-Bäume. In den Klammern die jeweilige Häufigkeit der Zuordnung zu einer Textpassage.

Lange Texte. Um schnell Probleme zu lösen sind die Texte vielen Teilnehmern zu lang.

Video-Tutorial - Vorteil

Schnelle Einführung. Viele Teilnehmer haben beschrieben, dass sie Videos nur nutzen, wenn sie etwas neues lernen wollen und um einen groben Überblick oder ein Konzept vermittelt zu bekommen. Andere Teilnehmer sagten, dass sie Video-Tutorials schauen um herauszufinden, ob eine Funktionalität von einer API oder einem Programm das Problem löst, welches der Teilnehmer gerade hat.

Video-Tutorial - Nachteil

Navigation. Viele Teilnehmer haben beschrieben, dass sie bei Videos immer anfangen, in der Zeitleiste hin und her zu springen. Sie versuchen, die für sie wichtigen Punkte zu finden. In Videos können von dem Ersteller Sprungmarken gesetzt werden. Allerdings wird dies häufig nicht genutzt.

Q&A Plattform - Vorteile

Konkrete Probleme. Auf einer Q&A Plattform finden die Teilnehmer oft auf ihr Problem abgestimmte Lösungen, da zuvor bereits andere Entwickler das gleiche Problem hatten und dort bereits nach einer Lösung gefragt haben. Ein Teilnehmer hat dazu erläutert: *„Aber wenn du in deinem Projekt auf einen Sonderfall triffst, dann hast du erst mal ein Problem. Aber das wird dann ja nicht weiter auf den offiziellen Seiten erklärt, sondern da musst du dann zu so Seiten wie Stack Overflow wechseln...“*

Kommentare. Alle Teilnehmer finden die Kommentarfunktion auf Stack Overflow gut. Durch Kommentare können veraltete oder falsche Lösungen korrigiert werden. Zusätzlich werden zu den Problemen öfter mehrere Lösungsmöglichkeiten angeboten. Die Teilnehmer können die für sie beste Lösung heraussuchen.

Inspiration. Einige Teilnehmer berichten, dass sie sich auf Q&A Plattformen Inspirationen holen, um anschließend mit

einem besseren Suchbegriff gezielter zu suchen oder Funktionsaufrufe im Kontext zu sehen.

Q&A Plattform - Nachteil

Schwankende Beitragsqualität. Die Beiträge und Kommentare sind nicht immer vollständig, hilfreich oder richtig. Das betrifft zum Teil auch die durch den Fragesteller als akzeptiert markierte Lösung. Meistens scrollen die Teilnehmer die ganze Seite durch, um eine passende Antwort auszuwählen. Ein Teilnehmer hat erzählt: *„ Da steht auch oft viel Mist und tatsächlich die beste Lösung ist meistens erst weiter unten.“*

Optimierte Dokumentation

Eine verbesserte Dokumentation würde aus Sicht der Probanden die folgenden Eigenschaften aufweisen.

Beispiele. Am häufigsten wurde von allen Teilnehmern gesagt, dass Beispiele nicht fehlen dürfen. Beispiele, die zeigen, wie Funktionen zusammen am besten benutzt werden sollten und wie diese funktionieren. Eine Aussage eines Teilnehmers: *„Beispiele sind mega wichtig. Das ist eigentlich immer das erste wo ich darauf schaue. Wie das überhaupt benutzt wird, was es zurück gibt, wie es funktioniert.“*. Ein anderer Teilnehmer wählt die Worte: *„Gute Beispiele sind eigentlich genau [die], die eben den größeren Zusammenhang zeigen. [...] Und wenn die da sind, braucht man auch nicht unbedingt Google anschmeißen [...] Man möchte wissen, wie ein bestimmtes Problem in seiner Gesamtheit gelöst wird. Wie halt jemand das Problem programmiertechnisch angegangen ist.“*

Gute Struktur. Ein übersichtlicher, leicht zu navigierender Seitenaufbau ist genau so wichtig, wie eine gute Suchfunktion und ein sehr gutes Google-Ranking.

Klassische Referenz. Viele Teilnehmer wünschen sich weiterhin eine klassische Referenz, in der jede Klasse, Methode und Funktion mit allen Parametern beschrieben ist.

Verschiedene Niveaus. Die Dokumentation sollten an verschieden erfahrene Programmierer angepasst sein. Der unerfahrene Einsteiger braucht eine andere Detailtiefe und Herangehensweise als der Programmierer, der ein spezielles Problem hat.

IDE Integration. Die Integration der Dokumentation in die Entwicklungsumgebung des Entwicklers ist vielen wichtig. *„Gute IDE Integration, dass du die IDE gar nicht erst verlassen musst. Mit der Info ob das noch in der Version funktioniert.“*, sagte zum Beispiel ein Teilnehmer dazu.

Rückfragen und Kommentarfunktion. Einigen ist eine Kommentarfunktion oder eine offiziell kommunizierte Stelle, an die Fragen gestellt werden können, wichtig. Ob das jetzt in die Dokumentation integriert ist oder auf einer externen Plattform realisiert wird, scheint nicht so wichtig.

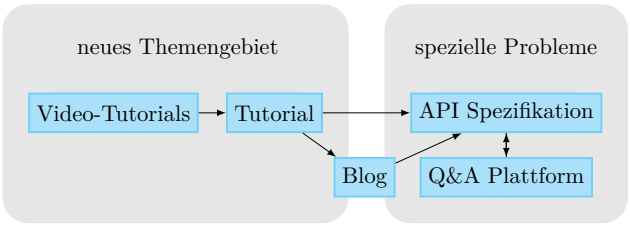


Abbildung 3: Interaktionsmuster zwischen den einzelnen Dokumentationsarten

Moderierte Lösungen. Diese Kategorie ist eine extra Funktionalität von dem Punkt "Rückfragen und Kommentarfunktion". Es geht darum, dass Fragen und Lösungen von dem Dokumentations-Ersteller beantwortet und bewertet werden, dass sehr gute Lösungen oder spezielle Fälle in die Dokumentation wieder einfließen und sie so verbessern.

Hintergrundwissen. Einige Teilnehmer regen an, in Dokumentationen mehr Hintergrundwissen zu vermitteln um dadurch Designentscheidungen nachvollziehen zu können. Ein Teilnehmer dazu: *„Wichtig ist auch das Warum. Warum wollte der Entwickler das so? Wenn man immer etwas Kontext gibt, hat man dann schon weitere Fälle mit abgedeckt.“*

Negativbeispiele. Fallen, in die Programmierer häufig tappen, sollten als Negativbeispiele aufgezeigt werden.

Sourcecode. Ein paar Teilnehmer würden sich freuen, wenn sie als letzte Möglichkeit noch in den Sourcecode schauen können. Ein Teilnehmer sagte dazu: *„ [...] dass man nicht nur die Dokumentation zum Sourcecode hat sondern man kann auch sagen: 'Gut die Dokumentation ist halt gerade blöd. Ich gucke mir jetzt einfach mal an, wie die Funktion aussieht.' Das ist einfach die letzte Dokumentation die man haben kann, dass man einfach den Sourcecode sieht.“*

Verlinkungen zu Referenz. Alle Funktionen, Methoden und Klassennamen in Tutorials, Beschreibungen oder Code-Beispielen sollten in der Dokumentation mit der klassischen Referenz verlinkt sein.

5 BEWERTUNG

In diesem Abschnitt werden die vorgestellten Ergebnisse aus Abschnitt 4 besprochen und daraus ein Interaktionsmuster der verschiedenen Dokumentationsarten untereinander abgeleitet. Die Eigenschaften einer optimierten Dokumentation werden zusammenfassend aufgelistet. Hierbei wird zwischen funktionalen und nicht-funktionalen Eigenschaften unterschieden.

Interaktionsmuster

Es kann ein Interaktionsmuster (Abbildung 3) zwischen den einzelnen Dokumentationsarten festgestellt werden. Wenn ein Entwickler in ein für ihn neues Themengebiet vordringt, helfen ihm zuerst Video-Tutorials, um einen Überblick über die Funktionalitäten und Konzepte zu bekommen. Um sich eingehender mit dem Thema zu befassen, kann zu einem textuellen Tutorial gegriffen werden und die ersten Erfahrungen durch das Programmieren der Beispiele gesammelt werden. Anschließend oder parallel unterstützt die API Spezifikation, ein Gespür für den Umgang, Aufbau und Überblick der einzelnen Funktionen zu bekommen. Ein Blog wird gegebenenfalls bei einem speziellen Setup genutzt. Er kann das Zusammenspiel zwischen verschiedenen Sprachen, Bibliotheken oder Versionen beschreiben. Wenn sich der Entwickler in dem Themenfeld auskennt und auf die ersten speziellen Probleme stößt, hilft ebenfalls die API Spezifikation bzw. eine Referenz, um einen Überblick über die Parameter oder alternative Funktionen zu bekommen, sowie Q&A Plattformen als Anregung zur weiteren Suche oder zum Finden inspirierender Lösungen.

Dieses Interaktionsmuster wurde von einigen Teilnehmern bereits umschrieben. Er würde ihnen bereits helfen, ihre Probleme zu lösen. Sie argumentierten, dass sie wissen, wo sie suchen müssen, um Lösungen in entsprechender Qualität zu finden. Ein Teilnehmer sagte dazu: *„Ist eigentlich mit der Kombi gar nicht mal so schlecht. Bis jetzt finde ich, man sucht sich halt sowieso, je nachdem was man für ein Problem hat, die geeignet Plattform raus."*.

Eigenschaften für eine optimierte Dokumentation

Den Eigenschaften, welche die Teilnehmer in Abschnitt 4 erarbeitet haben, kommt die folgende Aussage von einem Teilnehmer sehr nahe: *„Eine gute Dokumentation [vereint] verschiedene Arten einer Dokumentation [...] Man braucht ja mehreres in verschiedenen Stufen. Man braucht einerseits ein Tutorial für den Einstieg, aber man braucht dann auch die Referenz, wenn man wirklich ans Eingemachte geht."*.

Im folgendem listen wir die Eigenschaften, die eine optimierte API-Dokumentation haben sollte, um einen Entwickler angemessen bei der Erstellung von Code zu unterstützen, auf.

Funktionale Eigenschaften.

- **Rückfragen und Kommentarfunktion:** Jeder Bereich der Dokumentation sollte die Möglichkeit bieten, Fragen zu stellen. Der Dokumentation-Betreiber kann gute Lösungen in die Dokumentation aufnehmen oder falsche Lösungen als Negativbeispiel kennzeichnet.
- **Verlinkung zu Referenz:** Links innerhalb der Dokumentation, welche zur Referenz der Klasse oder Funktion führen.

Nicht-funktionale Eigenschaften.

- **Beispiele:** *„Beispiele sind mega wichtig."*, dieser Satz fiel in jedem Interview. Die Beispiele sollten den Zusammenhang von Funktionen und eventuelle Abhängigkeiten zeigen.
- **Gute Struktur:** Die Dokumentation sollte gut strukturiert, durchsuchbar und bei Google indexiert sein.
- **Klassische Referenz:** Nicht nur aus einer klassischen Referenz bestehen, aber auch eine haben.
- **Verschiedene Niveaus:** Die Dokumentation sollte verschiedene Niveaus haben, für Einsteiger und Fortgeschrittene.
- **IDE Integration:** Eine IDE Integration sollte gewährleistet sein, vielleicht mit einem Link zu Beispielen in der online Dokumentation.
- **Hintergrundwissen:** Die Dokumentation könnte Hintergrundwissen und Konzepte vermitteln, warum Dinge so umgesetzt wurden.
- **Negativbeispiele:** Einige Teilnehmer finden Negativbeispiele hilfreich, um sich vor geläufigen Fehlern zu bewahren.
- **Sourcecode:** Ebenso finden es Teilnehmer hilfreich, wenn sie in den Sourcecode der API schauen können um sich dort zur Not, falls die Dokumentation nicht weiter hilft, informieren zu können.

Die in Abschnitt 2 aufgeführten von Novick und Ward [14] herausgefundenen Eigenschaften lassen sich in dieser Studie wiederfinden.

6 ZUSAMMENFASSUNG UND AUSBLICK

Im Rahmen einer explorativen Studie unter Zuhilfenahme von Fokusgruppen mit insgesamt 26 Teilnehmern wurde die Vor- und Nachteile bestehender Dokumentationsarten und -systeme von Software APIs erarbeitet. Weiter wurde eine Menge von Eigenschaften für eine optimierte API-Dokumentation identifiziert. Darüber hinaus wurde ein Interaktionsmuster vorgestellt, der das Ineinandergreifen der Teilaspekte illustriert.

In einem nächsten Schritt soll der in Abschnitt Limitierungen erläuterte Punkt der „kommunikativen Validierung" von Reinders at al. [16] betrachtet werden. Des Weiteren soll anhand einer prototyptischen Umsetzung auf Basis der gesammelten Erkenntnisse evaluiert werden, ob ein optimiertes API-Dokumentations-System Entwickler dabei ausreichend unterstützt, Code zu erzeugen, der weniger sicherheitskritische Implementierungsfehler enthält. Zusätzlich zur von den Teilnehmern vorgeschlagenen Unterteilung der APIs nach unterschiedlichen Niveaus wird außerdem eine weitere Unterteilung gemäß der von Clarke [4] beschriebenen Programmierertypen erfolgen.

DANKSAGUNG

Wir bedanken uns bei den Gutachtern für ihr Feedback. Dieses Projekt (HA-Projekt-Nr.: 633/18-56) wird im Rahmen der Innovationsförderung Hessen aus Mitteln der LOEWE – Landes-Offensive zur Entwicklung Wissenschaftlich-ökonomischer Exzellenz, Förderlinie 3: KMU-Verbundvorhaben gefördert.

LITERATUR

[1] Y. Acar, M. Backes, S. Fahl, D. Kim, M. L. Mazurek, and C. Stransky. 2016. You Get Where You're Looking for: The Impact of Information Sources on Code Security. In *2016 IEEE Symposium on Security and Privacy (SP)*. 289–305. https://doi.org/10.1109/SP.2016.25

[2] Yasemin Acar, Christian Stransky, Dominik Wermke, Charles Weir, Michelle L. Mazurek, and Sascha Fahl. 2017. Developers Need Support, Too: A Survey of Security Advice for Software Developers. IEEE, 22–26. https://doi.org/10.1109/SecDev.2017.17

[3] Rolf Bürki. 2000. *Klimaänderung und Anpassungsprozesse im Wintertourismus*. Vol. 6. Ostschweizerische Geographische Gesellschaft.

[4] Steven Clarke. 2010. How Usable Are Your APIs? In *Making Software: What Really Works, and Why We Believe It* (1 ed.). O'Reilly, 545–565.

[5] Uri Dekel and James D. Herbsleb. 2009. Improving API Documentation Usability with Knowledge Pushing. In *Proceedings of the 31st International Conference on Software Engineering (ICSE '09)*. IEEE Computer Society, 320–330. https://doi.org/10.1109/ICSE.2009.5070532

[6] Gregor Dürrenberger and Jeannette Behringer. 1999. *Die Fokusgruppe in Theorie und Anwendung*. Akad. für Technikfolgenabschätzung in Baden-Württemberg.

[7] F. Fischer, K. Böttinger, H. Xiao, C. Stransky, Y. Acar, M. Backes, and S. Fahl. 2017. Stack Overflow Considered Harmful? The Impact of Copy and Paste on Android Application Security. In *2017 IEEE Symposium on Security and Privacy (SP)*. 121–136. https://doi.org/10.1109/SP.2017.31

[8] Peter Leo Gorski, Luigi Lo Iacono, Dominik Wermke, Christian Stransky, Sebastian Möller, Yasemin Acar, and Sascha Fahl. 2018. Developers Deserve Security Warnings, Too: On the Effect of Integrated Security Advice on Cryptographic API Misuse. In *Fourteenth Symposium on Usable Privacy and Security (SOUPS 2018)*. USENIX Association, Baltimore, MD, 265–281. https://doi.org/10.14722/eurousec.2017.23015

[9] Jonathan Lazar, Jinjuan Heidi Feng, and Harry Hochheiser. 2017. *Research Methods in Human-Computer Interaction*. Morgan Kaufmann. Google-Books-ID: hbkxDQAAQBAJ.

[10] Beate Littig and Claire Wallace. 1997. *Möglichkeiten und Grenzen von Fokus-Gruppendiskussionen für die sozialwissenschaftliche Forschung*. Vol. 21.

[11] Philipp Mayring. 2000. Qualitative Inhaltsanalyse. In *Forum Qualitative Sozialforschung/Forum: Qualitative Social Research*, Vol. 1.

[12] K. Mindermann and S. Wagner. 2018. Usability and Security Effects of Code Examples on Crypto APIs. In *2018 16th Annual Conference on Privacy, Security and Trust (PST)*. 1–2. https://doi.org/10.1109/PST.2018.8514203

[13] Sarah Nadi, Stefan Krüger, Mira Mezini, and Eric Bodden. 2016. Jumping Through Hoops: Why Do Java Developers Struggle with Cryptography APIs?. In *Proceedings of the 38th International Conference on Software Engineering (ICSE '16)*. ACM, 935–946. https://doi.org/10.1145/2884781.2884790

[14] David G. Novick and Karen Ward. 2006. What Users Say They Want in Documentation. In *Proceedings of the 24th Annual ACM International Conference on Design of Communication (SIGDOC '06)*. ACM, 84–91. https://doi.org/10.1145/1166324.1166346

[15] Aglaja Przyborski and Julia Riegler. 2018. Gruppendiskussion und Fokusgruppe in der psychologischen Forschung. In *Handbuch Qualitative Forschung in der Psychologie*. 1–17. https://doi.org/10.1007/978-3-658-18387-5_34-1

[16] Heinz Reinders, Hartmut Ditton, Burkhard Gniewosz, and Michaela Gläser-Zikuda. 2011. Kapitel III - Methoden der Empirischen Bildungsforschung. In *Empirische Bildungsforschung*. https://doi.org/10.1007/978-3-531-93015-2

[17] Marlen Schulz. 2012. Quick and easy!? Fokusgruppen in der angewandten Sozialwissenschaft. In *Fokusgruppen in der empirischen Sozialwissenschaft: Von der Konzeption bis zur Auswertung*. VS Verlag für Sozialwissenschaften, Wiesbaden, 9–22. https://doi.org/10.1007/978-3-531-19397-7_1

[18] Alexander Zeier. 2018. Benutzbarkeit von zustandsbehafteten, Hash-basierten Signaturverfahren. Masterthesis, Hochschule Darmstadt.

[19] Alexander Zeier, Alexander Wiesmaier, and Andreas Heinemann. 2019. API Usability of Stateful Signature Schemes. Springer International Publishing. To appear in *Advances in Information and Computer Security*.

A ABLAUF DER FOKUSGRUPPE

Der grobe Ablauf der Fokusgruppen im Überblick (Dauer ca. eine Stunde):

(1) Begrüßung
(2) Datenschutzrechtliche Erläuterung
(3) Fragen:
  (a) Seit wie vielen Jahre programmieren Sie schon?
  (b) Welche Quellen haben Sie zur Vorbereitung und Durchführung Ihrer letzten Programmieraufgabe/-lösung genutzt? Hat Ihnen dabei etwas gefehlt oder gestört?
  (c) Screenshots von einem Blog[4] zeigen. Was denken Sie darüber? Was halten Sie davon? Was fehlt Ihnen hier?
  (d) Screenshots von einem Video-Tutorial[5] zeigen. Was denken Sie darüber? Was halten Sie davon? Was fehlt Ihnen hier?
  (e) Screenshots von einer API Spezifikation[6] zeigen. Was denken Sie darüber? Was halten Sie davon? Was fehlt Ihnen hier?
  (f) Screenshots von einem schriftlichen Tutorial[7] zeigen. Was denken Sie darüber? Was halten Sie davon? Was fehlt Ihnen hier?
  (g) Screenshots von Q&A Plattform[8] zeigen. Was denken Sie darüber? Was halten Sie davon? Was fehlt Ihnen hier?
  (h) Wie sollte die perfekte Dokumentation einer sicherheitskritischen API gestaltet sein?
(4) Abschließende Informationen über die Studie
(5) Verabschiedung

[4] Blog: https://java-blog.de/tag/java/. Zuletzt aufgerufen am 14.06.2019.
[5] Video-Tut.: https://www.youtube.com/watch?v=BS9Y7ozXa60. Zuletzt aufgerufen am 14.06.2019.
[6] API Spezifikation: https://docs.oracle.com/javase/7/docs/api/. Zuletzt aufgerufen am 14.06.2019.
[7] Schriftliches Tutorial: https://www.w3schools.com/java/default.asp. Zuletzt aufgerufen am 14.06.2019.
[8] Q&A Plattform: https://stackoverflow.com/questions/10240538/use-string-in-switch-case-in-java. Zuletzt aufgerufen am 14.06.2019.