Docs for developers \*\*\*\*

An engineer’s field guide to technical writing

By Jared Bhatti, Zachary Sarah Corleissen, Jen Lambourne,David Nunes, Heidi Waterhouse

**“A great guide for beginners to overcome all the hurdles and succeed with well written and well organised documentation.”**

Unlike many other publications in the Technical Writing field, this book is written for developers who face the necessity of creating their own documentation, rather than for writers. The aim of the authors was to equip developers with a comprehensive guide to producing successful and effective documentation.

This relatively small volume covers several subjects, leading the aspiring writer through all steps of documentation creation, from audience analysis to maintenance and removal of obsolete content. Each of these topics is a big subject on its own. The authors opted to cover the essentials, leaving the details behind. There are extra references to some useful resources at the end of the book.

To illustrate the process and the pitfalls that developers who are undertaking writing tasks encounter, the authors include a story of a fictional software company named Corg.ly, that develops a creative toolkit for translating dog barks into English.

At the end of the development, the company faces the need to produce the required user documentation for their product, and each chapter shows through daily situations how the writing process should be tackled.

## 

## What is between the book covers

The book follows the writing process step-by-step in its working order, from planning and performing user research to content maintenance. The first part of the book deals with the writing and publishing aspects, taking us through each stage of the process. The second part introduces the practicalities of gathering feedback, establishing quality metrics, and maintaining the content.

Throughout the book, the authors show lots of sympathy for developers. They seek to convince the reader that they are not alone in having problems with writing. It's not because of their lack of ability, but because writing is hard. Feeling stuck during the writing process is common among writers and the authors offer several ways to overcome this problem.

Indeed, writing is not what developers do on a daily basis, and there are significant differences between coding and writing.

Facing a blank page can be daunting even for experienced writers, but the book claims that a good amount of planning is key, hence writing is part of a long process.

## Getting into the details

In the first chapter, readers are introduced to audience analysis. The authors insist that collecting some (or even any) data about the user is better than nothing, as understanding their needs can make all the difference in deciding on the purpose of the documentation and on what it should include.

There are a couple of sources that can give you some insight, such as ticket data, user interviews and surveys. Putting together user personas, user stories and user journey maps helps to organise the information that was gathered.

## Planning and drafting documentation

At the next stage (after developers have learned more about their audience) the authors start discussing the outlining and planning of documentation by sharing examples of the most common types of content . They also argue that analysing the audience and planning the content carefully can help to alleviate writer’s block – a helpful tip for developers who might see writing as a vague creative process.

The most common types of content described in the book are:

* Code comments
* README files
* Getting started guides
* Conceptual documentation
* Procedural documentation
* How-to-guides
* Tutorials
* Reference documentation
* Glossary
* Change documentation
* Troubleshooting documentation

They also share simple templates for each type of content (see Figures 1 and 2).

Text, letter

Description automatically generated

Figure 1 README file template

Text, letter

Description automatically generated

Figure 2 Conceptual guide

Additionally, at the end of the book the reader will find a comprehensive list of templates for diverse types of content, so developers can select the most suitable outline (see Figure 3). The authors strongly advise using templates as they allow writers to create documents which are consistent in style and structure.

Table

Description automatically generated

Figure 3 Content outline

What I found particularly convenient in this book is that with any new task the authors provide a checklist to ensure writers follow each step and don’t get lost (see Figure 4).

Table

Description automatically generated

Figure 4 Checklist for planning documentation

When covering the drafting step, which is known as one of the most difficult stages of a writing project, the book mentions a few important points to bear in mind such as using an inverted pyramid structure, breaking up big pieces of text, and seeking clarity and conciseness.  In my opinion, as coding and writing are creative processes that are different in nature, I appreciated what the authors were trying to achieve by bringing logic and method into the drafting process - this section might appeal more to developers who often see writing as a spontaneous work, totally dependent on inspiration.

## Editing

In this section again, the authors advise logical to avoid confusion. They draw parallels with coding by comparing editing (for writing) to testing (for coding).

They also stress the importance of peer review. I found the *plussing technique* quite interesting as it entails a constructive way to provide a critique - the idea is to only criticise if you can give a practical suggestion for improvement at the same time.

## Visual elements and code samples

After having discussed the purely textual aspects of the document, the authors embark on code samples and visuals.

Unlike writing, dealing with code samples lies within the developers' comfort zone.

Facing a relentless stream of new technologies, applications, and tools to take on board, developers know all too well how important code samples and what difference they can make to assist a better comprehension of new material.

The book claims that code samples should be clearly explained, concise and simple. For example, if the sample is executable code, then it should work and be trustworthy. In summary, code samples should demonstrate exactly what they are expected to do, and any additional functionality should be avoided. Language choice, code presentation and testing are all important aspects of document design.

Visual materials, such as screenshots, diagrams and video, are another powerful tool to improve users' comprehension. As with code samples, developers should be familiar with visuals as they tend to deal with charts and diagrams in their working lives.

Accessibility issues are also mentioned as it’s important to keep the consistency in shapes and colours in diagrams and aim for clarity at all times.

Diagram

Description automatically generated

Figure 5 Example of flowchart

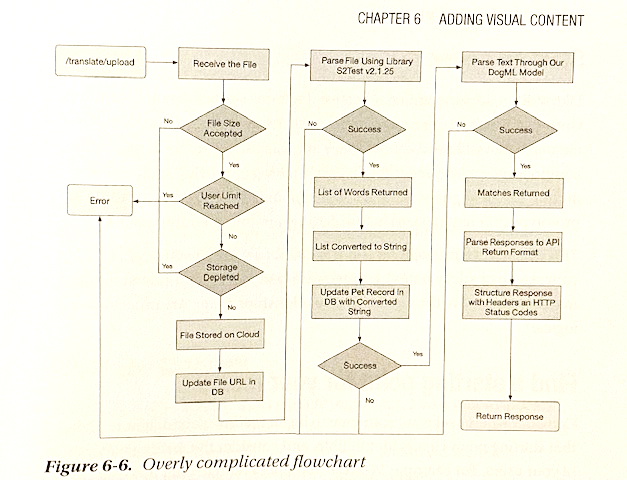


Figure 6 Overcomplicated flowchart

After the writing and visual materials have been finished, it is time to publish the documentation – there’s a section dedicated to finalising and approving documentation, choosing the publishing medium, and planning the publishing timeline.

## Improving and maintaining documentation

Gathering feedback and evaluating quality are aspects that are often overlooked. The authors explain how important it’s to maintain ongoing communication with the user in order to build up trust.

Several methods of feedback collection are examined, including collecting user feedback directly through documentation pages, using support tickets, performing sentiment analysis, running surveys and user councils.

Once the feedback is collected, it needs to be integrated into the documentation. The integration process that the authors describe in the book mirrors that of resolving incidents in the software industry. It starts with triaging. If the issue cannot be resolved, the documentation team requests more details from the user.

The authors emphasise that documentation is a way of connecting with the users; they advise to stay in touch with them, follow-up on submitted incidents and thank them for their feedback.

Measuring success

The quality assessment procedures are divided into two categories: functional and structural. The functional quality assessment looks at whether the documentation meets its goal by being accessible, purposeful, accurate, findable, and complete.

In its turn, the structural quality helps to decide if the document is written and structured well by being clear, concise, and consistent.

Analytics tools and terms, such as page views and bounce rates, can be used to evaluate the important aspects of content quality (see Figure 7).

Table

Description automatically generated

Figure 7 Questions about quality and corresponding metrics

## Information Architecture (IA)

Labelling and organising content are the best way to ensure users will find the information they are looking for in an easy manner.

This chapter covers sequential, hierarchical, and web-like types of content structure, followed by main site navigation principles, including information categorisation, landing pages, and navigation cues.

The practical aspects of organising content are raised, from auditing to outlining new information, and migrating the content to a new structure.  When speaking about outlining new information, the main advice is to refer to users’ expectations. In deciding upon the content organisation, the reader will be briefly introduced to the card-sorting method to categorise information. Then the authors show how to test the new information organisation through the users’ feedback.

## Documentation Maintenance

It is critical to update the documentation regularly, so that it always stays relevant to the latest releases. This is important for maintaining the trust and good relationships with the users. In a few short sections, the authors manage to distil the best practices for maintenance and deprecating documentation.

Maintenance includes planning, assigning personnel and allocating responsibility for documents, plus coordinating documentation accordingly along with new product with releases, checking links and grammar- some of these tasks can be automated.

Rewarding documentation maintenance efforts is highly recommended as recognition motivates professionals and employees.

In the end, the authors describe the process of announcing the documentation obsolete (deprecation) when corresponding software becomes obsolete. They point out the importance of retaining the obsolete content and flagging it as such to warn the user. Only when the documentation is not needed or not helpful at all, it should be removed.

## Final comments

My first impression was that when aiming to simplify the presentation of material, the authors went a little too far. For example, whilst the story of Corg.ly would look appealing to any dog lover, it seemes to be oversimplified.  However, after giving it another thought, I came to appreciate the book much more.

The book was conceived as a practical guide to developers writing their documentation, in other words, a "How to" guide to writing - and it achieves this purpose.

The authors managed to squeeze a good number of practices that are presented in a clear, concise, logical and consistent way throughout 200 pages, but overall I felt that the reader could benefit from more examples and resources.

Having said that, I liked the overall style of the book: clear, cheerful, and sympathetic. - bearing in mind that the community of developers is far from being uniform, the authors did a very good job of accommodating this highly diverse crowd.

I would highly recommend this book as a reference to anybody who is looking to create documentation - a great guide for beginners to overcome all the hurdles and succeed with well written and well organised documentation.

Reference:

# Docs for Developers: An Engineer’s Field Guide to Technical Writing

<https://origin-smile.amazon.co.uk/Docs-Developers-Engineers-Technical-Writing/dp/1484272161> (accessed 21 April 2022)

### About the Authors

Jared Bhatti  
Jared is a Staff Technical Writer at Alphabet, and the co-founder of Google’s Cloud documentation team. He’s worked for the past 14 years documenting an array of projects at Alphabet, including Kubernetes, App Engine, Adsense, Google’s data centers, and Google’s environmental sustainability efforts. He currently leads technical documentation at Waymo and mentors several junior writers in the industry.  
  
Zachary Sarah Corleissen  
Zach began this book as the Lead Technical Writer for the Linux Foundation and ended it as Stripe’s first Staff Technical Writer. Zach served as co-chair for Kubernetes documentation from 2017 until 2021, and has worked on developer docs previously at GitHub, Rackspace, and several startups. They enjoy speaking at conferences and love to mentor writers and speakers of all abilities and backgrounds.  
  
Heidi Waterhouse  
Heidi spent a couple decades at Microsoft, Dell Software, and many start-ups learning to communicate with and for developers. She currently works as a principal developer advocate at LaunchDarkly, but was reassured to find that technical communication is universal across all roles.   
  
David Nunez  
David heads up the technical writing organization at Stripe, where he founded the internal documentation team and wrote for Increment magazine. Before Stripe, he founded and led the technical writing organization at Uber and held a documentation leadership role at Salesforce. Having led teams that have written about cloud, homegrown infrastructure, self-driving trucks, and economic infrastructure, he’s studied the many ways that technical documentation can shape the user experience. David also acts as an advisor for several start-ups in the knowledge platform space.  
  
Jen Lambourne  
Jen leads the technical writing and knowledge management discipline at Monzo Bank. Before her foray into fintech, she led a community of documentarians across the UK government as Head of Technical Writing at the Government Digital Service (GDS). Having moved from government to finance, she recognizes she’s drawn to creating inclusive and user-centred content in traditionally unfriendly industries. She likes using developer tools to manage docs, demystifying the writing process for engineers, mentoring junior writers, and presenting her adventures in documentation at conferences.