Documentation Goals and Techniques

FOR KDE AND OPEN SOURCE

Thiago Masato Costa Sueto

Documentation for Humans

About Me

- Studied Brazilian Portuguese and German Language and Literature
- Translator by profession
- Technical Writer for KDE
- Furry, but that doesn’t matter
- **Jack of all trades:**
- User Support on r/kde → Wikis → Promo → r/kde mod → Websites → Bug Triage → Development → Documentation

What is a Technical Writer?

- Also called a Documentarian
- Also called a Technical Communicator
- “Technical Writers, first and foremost, are **testers** and **researchers**. [Their] job is to know what people want to achieve and precisely how to achieve it. Communicating that knowledge is the last step of the process and it shouldn’t take 10% of [their] time.” [1]

What a Technical Writer **does**

- Improves existing documentation
- Translates technical knowledge for its users
- Formats content to be accessible
- Creates new content (text, images, UIs)
- Makes it easy for readers to fulfill their tasks

What **steps** to take

- Plan
- Structure
- Write
- Review
- Publish

The ultimate **goal** of a Technical Writer

- Allow the user to ***complete their tasks*** in a seamless, ***accessible*** way

When does a Technical Writer **fail**?

- When the ***user cannot understand*** the documentation
- When the ***user gives up*** on reading the documentation
- When the user cannot ***scan*** the documentation for ***what they seek***
- When the documentation ***lies***
- When the documentation ***is broken***
- ***When the user cannot achieve a task***

The **four types** of documentation used in KDE

- ***Wikis***
- ***Tutorials***
- ***Application Manuals***
- ***API Documentation***

Wikis

- ***Volatile***
- ***Easy access***
- ***Easy onboarding***
- ***No review process***
- ***Few guidelines***
- ***MediaWiki formatting***

Tutorials

- ***Relatively static***
- ***Requires gitlab/invent account***
- ***Easy onboarding***
- ***Has a review process***
- ***Some guidelines***
- ***Markdown formatting***

Application Manuals

- ***Quite static***
- ***Requires gitlab/invent account or email***
- ***Difficult onboarding***
- ***Can have a review process***
- ***Some guidelines***
- ***Docbook XML formatting***

API Documentation

- ***Quite static***
- ***Requires gitlab/invent account***
- ***Difficult onboarding***
- ***Has a review process***
- ***Several guidelines***
- ***Doxygen formatting***

The most important things

- ***Audience***
- ***Navigation***
- ***Accessibility***
- ***Formatting***
- ***Language***
- ***Information Disclosure***
- ***Levels of Edit***

- ***Documentation is part of your product/software***

Audience

- ***Different from Persona***
- ***User, Admin, Developer?***
- ***Level of experience?***
- ***Minimum expectations?***
- ***Product use cases?***
- ***The tasks they want to accomplish?***

- ***The audience is everything***
- ***Always think like your audience***

Navigation

- ***“One-stop information lookup and retrieval”[2]***
- ***Headings, sections, titles, pages***
- ***Links, links everywhere***
- ***Cross-references***
- ***Keywords***

- ***The audience only needs to leave the website for third party information***

Accessibility

- ***Is it readable? (language and viewing)***
- ***Is it linkable? (linking to relevant parts)***
- ***Is it exposed? (flat vs deep hierarchy)***
- ***Does the audience know it exists?***

Formatting

- ***Abbreviations***
- ***Use of bold/italics***
- ***Capitalization***
- ***Parallel Constructions/Lists***
- ***Highlights***
- ***Monospaced text***

Language

- ***Style Guides***
- ***Formal Grammar***
- ***Audience-based Language***
- ***Contextual Language***
- ***Consistency***
- ***Typos/Spelling***
- ***Clarity***
- ***Accuracy***

Information Disclosure

- ***Is it a single story or multiple stories?***
- ***How many new elements are introduced per paragraph/section?***
- ***Will the audience understand it only with what you wrote?***
- ***Is the text too dense for your audience?***
- ***Have you introduced your audience to unfamiliar terms?***

Levels of Edit

- ***“Categorical scheme for editing text”***
- ***Better than what I summarized here!***
- ***1976 stuff***

- ***Think of it as: the possible ways you can contribute to documentation***

- ***Do only one or a few at a time!***
- ***Don’t overwhelm yourself!***

Accurate Descriptions > Buzzwords

- ***Lesson from KDE Promo***
- ***Don’t buzzword or attempt to convince, use merit and actual benefits***
- ***You audience wants clarity and accuracy to accomplish a goal***

Knowing it exists > How to use it

- ***Your audience is not dumb***
- ***If they know about a thing, they can search for it (think: keywords)***
- ***If they know how to use it, but not what it is or why to use it, it is useless***

Never document the future

- ***Golden* rule**
- ***Never make promises, document only what is currently there****
- ***Documentation is not the place for announcing new features***

The Curse of Knowledge

- ***“As experts, it is easy to forget that novices don’t know what you already know.”[3]***

No overlap or duplication*

- ***If possible, link to existing explanations***
- ***If not possible, make short summaries***
- ***Create content such that it can be linked later by someone else***
- ****Duplication is fine if it helps to clarify***

Topics Vs Procedures

Procedures

- ***Step-by-step instructions: How?***
- ***Action focused***

Topics

- ***Answers to specific questions: Who? What? When? Where? Why?***
- ***Explain-y is fine***

How to address problems

- ***There are global problems and local problems***
- ***Prioritize and fix global problems***
- ***If a problem occurs frequently, propose a guideline***
- ***Global problems are addressed once***
- ***Local problems are addressed individually***

The review process

- ***No red ink!*** ***(lesson learned)***
- ***Guidelines, style guides are not strict rules***
- ***You should be guided by reality and practicality***
- ***Clarify which changes are optional***

Future contributors

- ***Document your lessons learned***
- ***Prioritize onboarding***
- ***You are not the owner of the docs***
- ***Technical debt is also a thing in docs: future-proof***

No time to explain :(

- ***Agile***
- ***Managerial aspects***
- ***Reader feedback***
- ***Measuring quality***
- ***Additional tech***
- ***Formatting itself***

Extras

Resources to learn more

- ***Modern Technical Writing*: super short read, super introductory, worth how inexpensive it is (4 bucks)**
- ***(Dys)functional Documentation*: explains levels of edit, focus on standardization/guidelines, highly detailed on technical writing techniques and recommendations/best practices, must read**

Extras

Resources to learn more

- ***The Product is Docs*: mentions Agile, extremely comprehensive and the go-to recommendation, a tad too corporate focused**
- ***Technical Writing Process*: focuses on the managerial aspect of technical writing and contact with other team members, large corporate focus**

Extras

Resources to learn more

- ***Docs For Developers*: high level details of all aspects of technical writing, focus on API docs**
- ***Write The Docs*: a global community with many resources to learn about documentation, including an extensive guide and book recommendations**
- ***Daniel Beck's blog*: a blog I found to approach interesting docs topics**

Extras

Resources to learn more

- ***The Elements of Style (4th edition)*: a foundational book on good writing**
- ***Chicago Manual of Style (17th edition)*: the de-facto style guide on formal English grammar, expensive but definitely worth it even if you import it**

Thanks for your time!