Documentation

An Introduction to Documentation for Computer Scientists

In-Class Activity: Discuss Etter's Book Modern Technical Writing

Ask this question of each of your group members?:

- What was your favourite line from the book?
- What did you think was helpful about the book?
- What did you find difficult or confusing about the book?

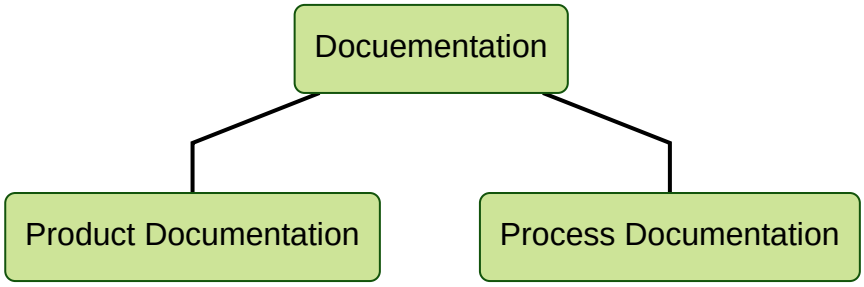
As a group, come up with an answer to this question:

- What is the purpose of the book? In other words, what is the book about?

Types of Documentation

Overview

Documentation can be broken into two major groups: **Product Documentation** and **Process Documentation**.



Process Documentation

Process Documentation refers to all documentation generated during the process of developing the product.

Process Documentation will (hopefully) be out-dated by the time the development is complete.

- Might be required as evidence in the event of a dispute or significant malfunction.

Tools: Process Documentation is often written in word processors (e.g.; Word), email clients, communication tools (eg: Slack), and wikis.

7 Examples of Process Documentation

> Always consider Audience and Purpose

1. Plans
2. Estimates
3. Schedules
4. Reports - formal
5. Reports - informal (often in email)
6. Working Papers - record of ongoing issues and technical problems
7. Standards - including coding and UX (user experience) standards

In-Class Exercise: Find Examples of Process Documentation

Here is an example of Process Documentation that describes the iOS human interface guidelines:
https://developer.apple.com/design/human-interface-guidelines/ios/overview/themes/

- As a group, find at least one other example of Process Documentation.

Ideas: Your sample documentation could be from the web. Or if you have a work experience in the field, you might describe (or show, if you think it does not violate confidentiality) process documentation from the workplace.

Product Documentation

Product Documentation describes and supports a specific product or suite of products.

Product Documentation is relevant for each step of the product lifestyle.

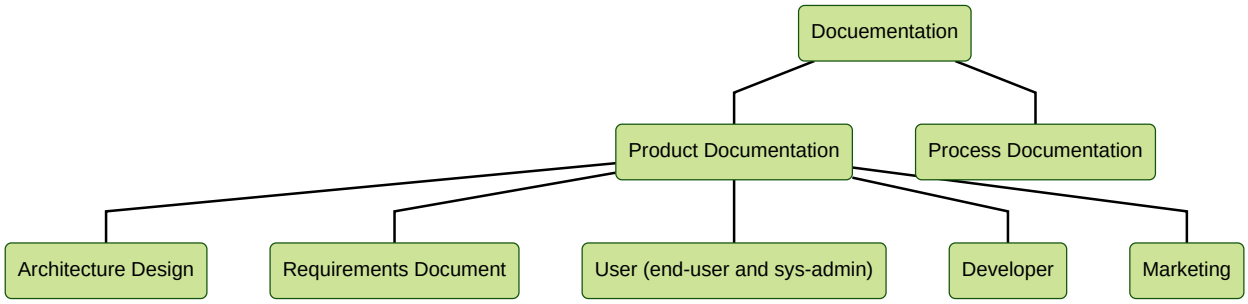
- It begins with design specifications, which may include use cases.
- It includes developer documentation such as technical requirements, specifications and comments.
- It also includes user guides and marketing materials.

Product Documentation is accessed, updated and referenced throughout the entire life-cycle of the product.

Product Documentation: Five categories

1. Architecture Design
2. Requirements Document
3. User (end-user and sys-admin)
4. Developer
5. Marketing

Diagram



Software Architecture Document

> Always consider Audience and Purpose

Written by software engineers and project managers

- To learn more about writing requirements, consider taking Project Management.

Contains diagrams to represent architecture

- e.g.: United Modeling Language (UML) diagrams

Contains use cases

Example of software architecture document:

- http://www.ecs.csun.edu/~rlingard/COMP684/ExampleSoftArch.pdf
- Note: Details of the above document will not be tested.

Requirements Document

> Always consider Audience and Purpose

Written by software engineers and project managers.

To learn more about writing requirements, consider taking Project Management. You could also read this *optional* resource: http://ddi-dev.com/blog/programming/how-to-write-software-requirement-specification-for-your-project/

Often uses large project manager software solutions and/or project tracking software.

Example of requirements doc:

- http://www.cse.msu.edu/~cse435/Projects/F09/PMR-droid/web/Resources/SRS_Final.pdf
- Note: This example is provided as a general reference. You are not expected to know the sections of this document, nor to be able to write this document.

End-User Documentation

> Always consider Audience and Purpose

In the last several years there has been a shift from locating user documentation outside the software interface (e.g.: user manual) to within the user interface (e.g.: point-of-use). In other words, user documentation is now considered to be part of the User Experience (UX).

UX principles also attempt to avoid the need for help documentation, even within the interface.

Examples of End-User Documentation

Examples of help within the user interface:

- Tool tips
- Error messages
- In-app help portals, including chats and texting

Additional user documentation is found online:

- FAQs
- Forums
- Online help

End-User Documentation for System Administrations

Documentation related to installation and support of a software products for System Administers is usually considered End-User Documentation.

Sys Admin Documentation is frequently found on a website or wiki (and not within the product, like other End-User documentation.)

Tools used in End-User Documentation

HATS (Help Authoring Tools) are a category of software designed for writing and distributing end-user and developer documentation.

- MadCap Software's Flare
- Adobe's Tech Comm Suite (including FrameMaker and RoboHelp)

HATs often have support for in-app (also called inline) help within the software product.

Developer Documentation

> Always consider Audience and Purpose

Developer documentation has recently changed: Instead of being an end-product, it is now integrated into the development process.

This trend is sometimes attributed to a trend toward agile development, where documentation is part of the process and does not have specific start and end points. Rather the process - and the documentation - acknowledges the product and documentation must evolve together. Both development and documentation are collaborative. There is no single source of knowledge.

Developer Documentation: Traditional Approach

Robust software tools have been used to facilitate and create technical documents. They often require a significant investment of learning and money.

HATs (Help Authoring Tools) are a category of software designed for writing and distributing end-user and developer documentation.

Two leading HATs are:

- MadCap Software's Flare
- Adobe's Tech Comm Suite (including RoboHelp)

Developer Documentation: New Approach

Increasingly, Markdown, Wikis, Static Site Generators, Forums and In-App communication are used for Developer Documentation.

This is the trend described in Etter's book *Modern Technical Writing*.

Writing in Markdown and then generating documents using a Static Site Generator (like Jekyll) provide a more dynamic and versatile approach to documentation.

This approach is also referred to as *Docs-As-Code*, since the same tools used to write code are also used to write docs.

Marketing Documentation

> Always consider Audience and Purpose

Marketing Documentation is often available through public websites.

- Example: https://scanbot.io/

For software, Marketing Documentation often describe real customers who have benefited from the product. These accounts are often called *Case Studies*, *Customer Stories*, or *Showcases*.

- Example: https://scanbot.io/en/sdk.html?campaign=INTER_Brand_ScanbotSDK&gclid=Cj0KCQjwuZDtBRDvARIsAPXFx3AvH7gprabaV1f1dyAvAyJGlsCJRrEVni_inlwsh0yj-dISQSt7idMaAs5BEALw_wcB
- Example: https://www.adobe.com/ca/products/one-adobe-solution-for-technical-content/customershowcase.html

In-Class Activity: The Two Solitudes of Tech Comm

Contrast these two (current) websites on documentation tools:

- https://www.process.st/software-documentation/
- https://academy.whatfix.com/technical-writing-tools/

Which one advocates for a more traditional approach?

Which one describes the tools from Etter's book?

What types of documentation would be supported by which set of tools?

Which set of tools would you prefer to use? Why?

Modern Technical Writing

Modern Technical Writing Components

Use a *Static Site Generating Tool* to nicely format information in a static website.

Consult with a designer. Use scripting to convert file types and/or transfer batch Markdown files to Jekyll.

Use a *Distributed Version Control Tool* to facilitate multiple authors, including developers and users.

Write/edit documentation in a markup language.

Modern Technical Writing Example

Use *Jekyll* to nicely format information in a static website.

Consult with a designer. Use scripting to convert file types and/or transfer batch Markdown files to Jekyll.

Use a *Github* to facilitate multiple authors, including developers and users.

Write/edit documentation using *Markdown*.

Trends and Tools in Documentation

Single-Source to Collaborative Writing

Previously, technical writers were hired to learn a tool, attend meetings, interview developers, and then write this knowledge in the documentation.

HATs are often considered "Single-Source Authoring Tools" since they required an expert set of knowledge about the HAT tool itself.

The process and tools described in Etter's book can be described as "Collaborative Web-Based Authoring Technologies" since they allow many people involved in the product development to contribute to the documentation on an on-going basis.

Different Tools for Different Approaches

Single-Source tools:

- No longer PDFs generated from word-processing and desk-top publishing tools (including HATS).
- XML expertise has been replaced with static site generators
- DITA is being replaced with simple Markdown files

Collaborative tools:

- Markdown
- Static Site Generators, such as Jekyll
- Open API (for APIs)
- GitHub

GitHub as a new tool in Documentation

Tools like GitHub facilitate a move from Single Source Authoring Tools (HATs) to Collaborative Web-Based Authoring Technologies (all the tools listed in Etter's book).

GitHub allows for content to be generated in simple Markdown files, formatted with GitHub's integrated Jekyll templates, and hosted on GitHub Pages.

GitHub allows many different users to add and update content, always tracking each contribution.

Read-The-Docs for Hosting Documentation

Read-The-Docs is like GitHub specifically designed for documentation. Much of it is free.

Read-The-Docs usually works with reStructuredText (RST) and Jekyll, instead of Markdown and Jekyll.

Resources on Modern Technical Writing

Tom Johnson's Blog: https://idratherbewriting.com/

Write-The-Docs: https://www.writethedocs.org/

- Including annual conference in Portland

In-Class Exercise: Amazon Fire TV Case Study

Begin by checking out the link below and asking yourself what tools might go into writing, managing and publishing this documentation:

- https://developer.amazon.com/docs/fire-tv/device-specifications.html

Next, find out how it was created by reading a post from Tom Johnson's blog on how he created the above site.

- https://idratherbewriting.com/2018/08/31/handling-multiple-versions-of-content/
- What tools does Johnson use? List two tools he uses.
- Why does Johnson use these tools? Give one reason.

> Should I understand Jekyll after reading this post? Of course not! This case study is a chance to view the final product of the tools we're learning about. We won't get this far in this course, but the goals of this section of the course are to introduce you to 1) a new paradigm of tech com, 2) tools in the field and 3) resources to get you started on the correct path.

In-Class Activity: Work together to host a Markdown file on GitHub, format it with a Jekyll template, and host it on GitHub Pages.

Essentially, use the support of your group to make sure you can do the tasks that you are required to describe in README/instruction set for Assignment 2.

Take notes while you are doing these steps so you will remember what is tricky or easy to mix up. These notes will be helpful when writing your instruction set.

Note: Next week we will learn about requirements for READMEs and instruction sets, and have a demo of this activity.