Don’t Let Documentation Be a Buzzkill

Victoria Palmiotto and Kristin Ito

victoria.palmiotto@mindbodyonline.com and kristin.ito@mindbodyonline.com

Abstract

Whether you love it, or love to avoid it, documentation is necessary in the worlds of software development and quality assurance. Crafting content for both internal and external audiences to record the work you’ve done is essential for your organization and end users to fully realize the quality of your product. Oftentimes, a non-technical task such as writing is approached as a tedious chore; details may be overlooked as you go through the motions.

By blending some basic writing fundamentals into your documentation process, you can:

- Create quality documentation that reflects the quality of your software
- Ensure your documentation is valuable, understandable, and easy-to-navigate for people outside of your team
- Streamline documentation processes
- Develop documentation that’s usable and consistent
- Promote accountability and collaboration among your team

Biographies

Victoria is a content editor on MINDBODY’s Product Development Team, where she works at the intersection of technical and nontechnical teams. To translate tech-talk into valuable end-user content, she relies on the powers of collaboration and concision.

Kristin is a content editor at MINDBODY, where she writes about new features in their apps and products. She’s interested in style guides, diversifying content types, and developing processes to maintain content quality. Kristin was previously an editor at Expedia.

1 Introduction

We get it. Documentation takes time that you just don't have. Plus, there's an inherent conflict between the agile environment and documentation, and it's natural to wonder: Is agile documentation an oxymoron?

We're here to argue that documentation in an agile environment is not only possible, but can work really well when it's agile, too. Quality documentation has several characteristics—like its ability to change along with the software—that we'll go over in the following pages and that will help you create documentation that works within an agile context and reflects the quality of your software. After all, "your documentation is an advertisement for the quality of your code" (Raymond, 2000), and we'll show you how you can best advertise the value of your product through quality documentation.

2 What we talk about when we talk about documentation

Because there are several types of software documentation, and because people think of different things when referring to documentation (Kiss, 2011), it will be useful to lay out the types of documentation we are *not* talking about. We're not talking about technical documentation of source code or of algorithms, interfaces, and APIs. Nor are we referring to any type of requirements documentation, which is goal-oriented and created at an early stage in development; or to architecture design documentation, which also tends to be inceptive. We're not talking about testing or maintenance documents either, which do play a role in software quality, but are not the focus of this paper. All of this documentation is what Ian Sommerville defines as process documentation, or a record of the ideas and plans put forth along with the steps taken to implement them (Sommerville, 2001).

What we *are* talking about is product documentation, and more specifically under the umbrella of product documentation, what is referred to as the description document or functional description document. While this is sometimes conflated with or even considered a type of user documentation—which is often generated from the description document—we are talking about the *raw materials*, the written communication that is needed to understand how to use your software.

Product documentation serves different purposes for different audiences. While danger lurks in a one-size-fits-all approach, there is potential for efficiency and usefulness with proper documentation planning and creation. Quality software documentation can act as:

- **A single source of truth**
  This is especially important in fast-moving environments that are susceptible to tribal knowledge traps. In iterative agile development, multiple sprints with small releases may be required to produce a full product or feature. Software changes and improves as product and business objectives evolve. Documentation serves to clearly define where the product stands.

- **A vehicle for knowledge transfer and training**
  On some development teams, team members and roles change as fast as the code does. Up-to-date documentation is a quick and reliable way to get everyone on the same page at any stage of a project. New hires can familiarize themselves with the software, and team-swappers can align themselves with the goals and objectives of their new assignment.

- **Success criteria to define project completion**
  In a scrum/sprint environment, projects are constant, and the end of one sprint can often bleed into the start of another. Clear lines are sometimes not drawn to delineate when a project is complete. Imagine running a half marathon, and instead of a line to cross at the end of the race, there's a general finish area that spans a half mile or so. When do you stop running? Documentation can act as your finish line. Once you've submitted your documentation for the sprint, you can consider your project complete and gear up for the next sprint.

- **Support material for users**
  Despite how intuitive your software may be, users new and old will likely need help at some point—whether it be starting up, adjusting to a new feature, or troubleshooting on the fly. Your documentation is the basis (or sometimes the prevailing source) of how-to and what-to-do-when support material. In describing the software's functions, you're explaining how it works and how to use it, which is helpful for anyone who did not build it. What's more, if the feature you're working on overlaps with another team's feature sets, this explanation of how it works and how to use it holds internal value.

- **The announcement about what you've made and why it's great**
  With the fast-paced nature of software development, it's easy to become transfixed on the goal of project completion—but what happens once the code is perfected and the tests are complete? While finishing projects and producing new products and features is satisfying within your organization, these accomplishments hold little value if no one knows about or uses what you've created. Imagine your finger-painted childhood masterpieces never getting hung on the fridge—it's almost heartbreaking. Documentation tells people what you've made. Whether it's modified and delivered straight to end users, or funneled through a content or marketing team to package and deliver, documentation is the starting point for getting the word out about your work.

3 What is quality documentation?

The argument can be made that quality is subjective; standards may vary from one company to the next, and from one person to the next. So it's no surprise that quite a few studies have been conducted in an effort to more narrowly define quality, to peg down what it really means in terms of software documentation. For example, in *The Value of Software Documentation Quality*, the authors' survey found that "the most important quality attributes with regard to documentation quality are accuracy, clarity, consistency, readability, structuredness, and understandability" (Dautovic, Plosch, Saft, 2014). Taking these attributes into account, as well as what we've garnered from our own experience, we've compiled a list of what we believe defines quality documentation.

4 Quality documentation considers your audience

Quality software requires and deserves quality documentation, and one way to ensure this is to always consider your audience. Who you are writing for determines what needs to be included and with what level of detail. Audience will also determine the structure of your documentation, as well as language, style, formality, and other writing elements.

Considering your audience means thinking about your content. Andrew Forward found through a survey on software documentation, that "content is the most important factor. All document tasks (creation, maintenance, verification, validation) should always keep the target audience in mind" (Forward, 2002). Thinking about what your audience needs to know, what they don't need to know, and what order they'll want to read it in, will allow you to cut unnecessary content, structure it with purpose, and create documentation that is more efficient and effective.

Considering your audience also means thinking about *why* you're creating documentation. As a developer or software quality tester, you might be relatively removed from the end user in your quest to test or in your compulsory focus on the deliverables. Documentation can provide an impetus to get team members thinking about the purpose of documentation. *Who will be reading this, and why?*

Another important question to consider is: *What do I want them to understand?* If the point of product documentation is to communicate how to use your software, then the real value here is *knowledge transfer*. In other words, your job is to create documentation that your audience can understand and learn from. Furthermore, considering your audience means not making assumptions about the level of

knowledge your readers have regarding the software. Knowing your audience can prevent this from happening.

> *"I've never met a human being who would want to read 17,000 pages of documentation, and if there was, I'd kill him to get him out of the gene pool."*
> *Joseph Costello*

Listed below are a few common audiences. You may need to consider some or all of them, depending on the size and scope of your team.

- **Internal - Developers/QA**
  Much of the documentation shared between development teams will likely be process documentation; however, for this, as well as for any product documentation, you should consider that while your audience may understand the technical aspects of the project, they may be unfamiliar with other aspects. You'll need to address the distinctive elements of your particular environment, the history and status of the project, or other specifications unique to your team. For this audience, technical language and simplification will be lesser considerations, while context and regard for future developers will be larger concerns.

- **Internal - Technical support, IT**
  This audience has a strong level of technical knowledge and experience with the software; however, you'll need to consider that their knowledge is more support-focused and is not necessarily up-to-date. The main question you'll want to think about is *What will they need to know in order to assist in troubleshooting issues?* In terms of content, consider that your readers will be dealing with different settings and exceptions—this information should be included, but in a way that they can find quickly while communicating with the user. If you're documenting a change or a new feature in your software, you'll want to be specific about what has changed. Providing a concise before-and-after summarization, for example, can help this audience quickly find the answers they need to support users.

- **Internal - Editors, writers, marketing, other non-technical teams**
  For the non-technical audiences that will be reading your documentation, the most important part of the equation will be the why. Putting the why up front will allow readers to decide how much of your documentation they need for their purposes. Additionally, the further your readers are from software creation, the more important your documentation becomes: "Documentation becomes a better option for you the greater the distance, either physical or temporal, between the individuals who are communicating" (Ambler, 2001-14). For example, if a writer is working with your documentation to create a user's guide, then your documentation is likely the only source they have in terms of functionality. Without the ability to test the software themselves, how else are they supposed to know how your software works? You'll need to think about what the writer will need to know so that they can best explain it to the users.

- **Internal - Stakeholders**
  Stakeholder participation in the agile process is a contentious topic itself, but in terms of documentation, you can make sure that even if a stakeholder takes a quick look at it, they'll be able to understand the value of the project.

- **External - Users**
  If you're working alone or your team does not have the bandwidth for editors or writers, you may be responsible for creating user documentation, such as manuals or support articles. In this case, you'll need to consider your end user at all times, and make no assumptions in terms of technical knowledge. This doesn't mean "dumbing down" your content; it just means you need to more carefully consider what the user needs to know, and what is the most clear and concise way to inform them.

Audience is a consideration that should become intrinsic. Once the creator figures out who they are writing for, and why, they should keep their audience, or audiences, in mind during the entire process.

5 Quality documentation is a part of the project

Oftentimes, documentation is wrongly treated as an independent supplement to a project, rather than an integrated component. It can be pushed to the back burner, forgotten about until the last minute, and then given a haphazard effort, which yields ineffective and inadequate documentation. To avoid this, make sure "documentation becomes part of the development process, not a separate activity" (De Lancey, 2012). Thus, at your kick-off meeting or project starting point, make sure documentation is acknowledged.

As you're establishing goals, defining requirements, and divvying tasks, include documentation as part of each. "Treat documentation like any other requirement: it should be estimated, prioritized, and put on your work item stack along with all other work items that you must address" (Ambler, 2001-14). As with any work item, documentation requires time, attention, and accountability.

- **Time**
  As you're refining a project plan and estimating efforts, be sure to include time allowances for documentation.

"This is how you do it: You sit down at the keyboard and you put one word after another until it's done. It's that easy, and that hard."
Neil Gaiman

- **Attention**
  At the beginning of your sprint, decide who will contribute to the documentation and what their contributions will be. Distribute assignments and establish checkpoints. "Documentation should take on a collaborative nature. It should not be written by one person to perfection, and then shared with others. It should instead be shared often during draft to gain input" (Moreira, 2013).

- **Accountability**
  Who has the final approval of the documentation? What other teams or team members will the documentation be distributed to? Who will be the point person for any questions that arise? Where will the documentation live? The answers to these questions should be added to your project requirements checklist, and should be defined when the project starts.

6 Quality documentation is quality

Hold the quality of your documentation to the same standards as the quality of your software; it should be easy-to-use, consistent, aesthetically pleasing, straightforward, error-free, and reliable. Think of the final document as your product and the content as your code.

6.1 Easy-to-use

Your documentation will be viewed by a number of audiences (which you've already carefully considered). As such, you'll want to make sure that no matter who is reading, they're able to navigate the document with ease and extract the information that's valuable to them as efficiently as possible.

- **What's in a name?**
  The title of your document and the file name should be valuable to a reader. They should know what they're getting into before they double-click. Develop a consistent naming convention that identifies the file as product documentation, and identifies the specific product. Use this convention from sprint to sprint and across all development teams.

- **Create a functional first page.**
  Every page of your document should serve a purpose and provide value, including the first page. One way to eliminate unnecessary space between the reader and the information they need is to avoid redundant, title-only cover pages. Instead, approach the first page as an executive summary. Answer some questions for the reader right off the bat by addressing:
    - What the new product or feature is and a synopsis of what it does
    - Who worked on it, and who the point of contact is
    - Project start and release dates
    - The "why" and reasons behind the project
    - Where you can find this product or feature (e.g., if it's a new feature within the software, provide a menu path)
    - The dates the documentation was created and updated

- **Headers, footers, and page numbers, oh my!**
  Make sure everyone is on the same page—literally—when contributing to or using your documentation. In your header, include the name of the project to provide an easy way to identify what the document is, whether someone starts on page one or dives in somewhere in the middle. In your footer, include page numbers, which help tremendously with individual navigation and team collaboration.

- **Always include a table of contents.**
  Different people will access the document for different reasons. Whether it's three or 30 pages long, nobody wants to scroll through each page to find the specific section they need. "Finding useful content in documentation can be so challenging that people might not try to do so" (Forward, A; Lethbridge, T; Singer, J. 2003). Break up your work into clear sections, and map those sections with a simple table of contents.

> *"Never use tricky or irrelevant headlines. People read too fast to figure out what you are trying to say."*
> *David Ogilvy*

- **Establish a heading hierarchy for your sections and subsections.**
  Google Docs and Microsoft Word make this easy with their default style options. With a style set, you'll also be able to whip up and update that table of contents with a few clicks. Make the differences between headings and subheadings distinct.

  Assume that not everyone will start on page one, so make sure your headers are descriptive enough to allow someone to pick up from anywhere. If you have multiple sections with the same subsections (e.g., a "Screenshots" subsection), name each subsection with reference to its parent section (e.g., "Screenshots: Home Page").

6.2 Consistent and aesthetically pleasing

Readers of your documentation should have a consistent experience. Thus, within an organization, product documentation should be consistent from page to page, sprint to sprint, and team to team. More so, documentation should be regarded as a product itself; the final file should be polished and presentable. Templates and style guides are effective tools in achieving uniform, refined documents.

- **Use a template.**
  Develop a template and utilize it for every piece of documentation. A good template will:
  - Define and order the always-necessary sections and subsections
  - Have an established style set
  - Have customizable headers and footers
  - Be distributed for use by anyone who creates documentation
  - Reduce creation time of new documentation

- **Exercise font awareness.**
  With hundreds of fonts to choose from, keep the following in mind: Your font should not change from one paragraph or page to the next. If you use different fonts to distinguish between headers, sub-headers, and body text, limit yourself to no more than three fonts per document.

  Choose something tried and true. Font should not distract readers or make any implications about your software.
  - Arial, Calibri, Times New Roman, and Cambria are all safe bets.
  - Fonts like Comic Sans, Mistral, or Freestyle Script will detract from the readability and credibility of your documentation.

- **Strive for scannability: white space, short paragraphs, bullets, and bold.**
  Avoid walls of text, and try to keep paragraphs to a five-sentence maximum. The white space between paragraphs and around text and images helps readers to scan and move through your document more quickly. Bulleted lists are an easy way to organize information and increase scannability.

  Use **bold** strategically and sparingly. Bolding is an easy way to draw attention to important details. Be wary of ALL CAPS, *italics*, and underlining. These text modifications can sometimes make content harder to read or create confusion; use them only when emphasis is necessary. Keep in mind, too, that many will assume an underlined word or phrase is a hyperlink.

- **Show and tell with images and screenshots.**
  Images and screenshots allow readers to see what you're talking about, and visually familiarize themselves with the product or feature.

"Of all of our inventions for mass communication, pictures still speak the most universally understood language."
Walt Disney

Find a balance: Your documentation should not be a series of screenshots, nor completely void of them. However many you include, make sure your graphic elements are useful. "Screenshots should include some sort of annotation. Adding an arrow, a circle, or number sequences can make end user documentation completely dummy proof, and save end users from having to figure out what to do" (DeVore, 2014).

Snagit, Skitch, and Snipping Tool are all easy-to-use screen capture tools that make grabbing and incorporating screenshots a breeze.

- **Develop a style guide.**
  Is it 9:00 a.m. or 9 AM? When is it okay to use **bold**? Are headings Title Case, or Sentence case? What about the Oxford comma? Should page names be in quotations? These considerations may seem insignificant, but if they are never addressed and defined, your content style can change from page to page or author to author, which lends to sloppiness and confusion. "Like design style guides, which lay out guidelines for colors, fonts, and other design elements, content or

editorial style guides are a set of guidelines for making decisions about spelling, grammar, structure, and style" (Griffis, 2013).

Don't have a style guide? Resources like the *Yahoo! Style Guide*, the *Chicago Manual of Style*, and the *Associated Press Stylebook* are comprehensive and reliable starting points. As you create documentation, build an in-house style guide to set standards for words, phrases, and nuances that are unique to your company's products, voice, and language.

6.3 Straightforward

Your documentation should be concise and easy to understand. Just as a two-click workflow is favorable over a six-click in your software, shorter sentences and smaller words are favorable in documentation.

- **Be concise.**

> *"The most valuable of all talents is that of never using two words when one will do."*
> *Thomas Jefferson*

- **Keep it simple.**
  Avoid jargon or internal-only words and phrases. Even if it's not, imagine that your documentation needs to be read and understood by someone outside of your organization. Spell out any acronyms or initialisms the first time they are referenced.

- **Provide ample background information and context.**
  Don't assume the reader knows everything that you know—they don't. If you're explaining what's new, be sure to acknowledge what's old. This consideration will lean largely on your audience identification.

6.4 Error-free and reliable

Call in a second set of eyes to check for writing quality and content accuracy. Refine your documentation as you would refine your software. "Write a little bit, show it to someone, get feedback, act on that feedback, and then iterate" (Ambler, 2001-14). When you've completed the documentation, ask a colleague for a copy edit. A typo is a surefire route to discredit your work and create confusion.

> *"Put it before them briefly so they will read it, clearly so they will appreciate it, picturesquely so they will remember it, and above all, accurately so they will be guided by its light."*
> *Joseph Pulitzer*

- **Check for accuracy against your teammates and the software itself.**
  Make sure the buttons, links, and pages you write about match what's in the software.

- **Include any known future plans.**
  This can be especially helpful for content and marketing teams to map their support and announcement plans for your products. For example, you might be releasing an update to your software that allows clients to purchase products with credit cards on their phones. Next month, you're expanding that capability to include gift card and PayPal payments. Instead of bugging end users with two too-similar product update notifications, your communication team(s) may opt for one combined announcement.

- **Work collaboratively and cross-departmentally.**
  If you're working on a feature that overlaps with another team's features, be sure to consult someone from that team.

- **Identify a subject matter expert for each project.**
  This point person should field all questions or requests for more information regarding the product.

7 Quality documentation is shared

Once you've created quality documentation, don't let this resource go to waste. Too often, documentation remains with the owner or in a folder on someone's desktop, rendering all of the creator's hard work futile. Quality documentation should be shared, so that your company and your users can benefit from learning about the software. Here are a few things to consider when sharing documentation:

- **Housing your documentation**
  Accessibility is vital. Whether a project manager, content editor, or stakeholder needs to review your documentation, it needs to live in a place that is easy to get to. Depending on what your company uses for other records and written communication, you might consider a wiki, Sharepoint, or any number of platforms available. The important thing is that everyone who might need to access the documentation is able to, and that those who aren't meant to access it, can't. You'll want to consider rights-based access to your software documentation and should be in conversation with your security team. In addition, a designated person or team should be in charge of uploading and updating the documentation, and should let others know when this documentation is updated and ready to be accessed.

- **Open communication and collaboration**
  Not only should the documentation be shared so that other content can be created from it, but the *process* of sharing information should be open and collaborative. For example, on many teams the software quality testers are part of development teams and are the ones responsible for creating QA documentation. Rather than just handing off this documentation to the technical writers or whoever is going to create the user documentation, the software testers and the writers should have an ongoing dialogue. Whether this is in the form of in-person meetings for Q&A, emailing back and forth, or using the comments feature on a Google or Word doc, this conversation between testers and writers ensures that the user documentation is accurate in terms of functionality: "The software quality practitioner must take an active role in the review and evaluation of the user documentation. If the software cannot be used properly, it matters little if it is a quality product" (Horsch, 2003).

"The single biggest problem in communication is the illusion that it has taken place."
George Bernard Shaw

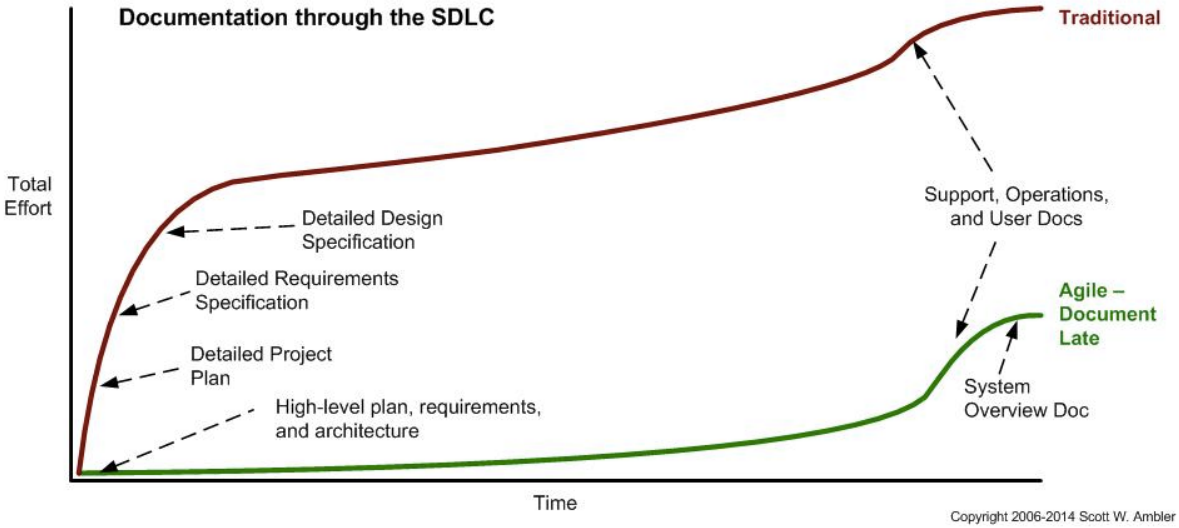
- **Osmotic communication**
  Completed documentation needs to be shared, but your documentation can also be improved by having those who communicate with your external audience, or the end users, nearby during the software grooming and creation processes. For example, most companies have writers or marketing team members in different buildings for practical reasons. They often have to be on the phone or have jobs that would be distracting for developers or testers. However, if these communicators can be in the same room as developers, at least some of the time, the information that they receive has more context, and they are more likely to understand the reasons behind the changes in your products. This has been referred to as osmotic communication, the idea that physically sitting in a room with developers has a positive effect on the transfer of information:

"When people are working close together, both physically and temporally, there exists an opportunity for what Cockburn [Alistair] calls osmotic communication: indirect information transfer through overhearing conversations or simply noticing things happening around you" (Ambler, 2001-14). This way, not only will any user documentation be more accurate and well-informed, but your own documentation can improve from being around people whose job it is to consider the needs of the user.

8 Quality documentation is dynamic

How many times have you thought, "I'll just leave the documentation until after our sprint is over, and then I won't have to change it as things change in the software"? It's true—you don't want to have to double up on your workload, especially at the beginning of the sprint when software plans are constantly changing. To avoid this, here are a couple of points to keep in mind:

- **"Document late"**
  We don't really like this term, because it implies that you should squeeze your documentation in at the end of the sprint, which can result in rushed, lower quality work. However, "document late" just means that you should document at a point in the sprint when you have the core specifications set and are past the stage of abstract ideas and hypotheticals. In other words, rather than the traditional method that begins with a surge of detailed documentation, it's more efficient to "document stable concepts, not speculative ideas" (Ambler, 2001-14).

- 

- **Keep it "alive"**
  "The best documents are written iterative, not all at once" (Ambler, 2001-14). Once you've started documenting, it's important to be vigilant about recording changes. Whether this means taking an extra minute after your morning scrum meeting to jot down a change, or noting any updates at the end of each day, "If the documentation is not maintained in lock step with the code, it quickly becomes inaccurate" (Johnson, 2013).

In the future, we'll all likely be using documentation software that automatically updates when you update code. Craeg Strong, technical lead for the FBI, discusses this possibility: "If you had, for example, documentation that's generated from source code or maybe some reports that get automatically generated, it would be great if by refactoring the design and refactoring the software, you're simultaneously updating the documentation" (Philipp-Edmonds, 2014). Currently, there is a lot of research on ontologies as a base for documentation, and there are even authoring tools in development, such as I-

Doc, which automate the process of generating software documentation, and accommodate agile maintenance (Johnson, 2013).

But until these tools are readily available, as a creator of documentation, you're responsible for keeping it "alive."

9 Quality documentation is fun!

No, but really. Writing documentation is creative in a different way than development or testing, and it allows you to take a step back and think about what you've been working so hard on. It's also an opportunity to get to know other team members and learn what they do with your documentation. Through understanding what they need to do their jobs, empathy is gained, and that's always a good thing.

It's also exciting to know that your documentation can be used in a myriad of different ways. Your documentation can evolve into online tutorials, blog posts, and other public-facing content that you can take ownership of. While it's validating to know that you've created quality software, it's just as validating to know that you've enabled people to learn how to use it.

References

Ambler, Scott. 2001-2014. Best Practices for Agile/Lean Documentation. http://www.agilemodeling.com/essays/agileDocumentationBestPractices.htm

Basaraner, Cagdas. 10 Software Documentation Best Practices. http://java.dzone.com/articles/10-software-documentation-best

Dautovic, A; Plosch, R; and Saft, M. 2014. The Value of Software Documentation Quality. Quality Software (QSIC), 2014 14th International Conference on Quality Software.

De Lancey, Tom. 2012. Emergent Documentation: One way that Agile is very different from Waterfall. https://www.linkedin.com/grp/post/43421-218531537

DeVore, Jonathan. 2014. 10 Examples of Great End User Documentation. http://blog.screensteps.com/10-examples-of-great-end-user-documentation

Forward, Andrew. 2002. Software Documentation – Building and Maintaining Artefacts of Communication. https://www.site.uottawa.ca/~tcl/gradtheses/aforward/aforward_thesis.html#_Toc26336969

Forward, A; Lethbridge, T; Singer, J. 2003. How software engineers use documentation: The state of the practice. http://www.researchgate.net/publication/3247967_How_software_engineers_use_documentation_The_state_of_the_practice

Griffis, Gigi. 2013. How to Make Style Guides That People Will Use. https://blog.gathercontent.com/how-to-make-a-style-guide-that-people-will-actually-use

Horch, John W. 2003. *Practical Guide to Software Quality Management*, pp. 220.

Johnson, W. Lewis. Dynamic (Re)Generation of Software Documentation. http://www.researchgate.net/publication/2817866_Dynamic_(Re)Generation_of_Software_Documentation

Kiss, Fabian. 2011. Documentation in the agile software development process. http://www.slideshare.net/headrevision/documentation-in-the-agile-software-development-process

Moreira, Mario. 2013. Right-sizing Documentation in an Agile World. http://cmforagile.blogspot.nl/2013/06/right-sizing-documentation-in-agile.html

Philipp-Edmonds, Cameron. Best Practices for Lean Documentation: An Interview with Craeg Strong, http://www.stickyminds.com/interview/best-practices-lean-documentation-interview-craeg-strong

Raymond, Eric Steven. 2000. Software Release Practice HOWTO. http://en.tldp.org/HOWTO/Software-Release-Practice-HOWTO/documentation.html

Sommerville, Ian. 2001. Software documentation. http://www.literateprogramming.com/documentation.pdf