Foundations of Practice: An Ethnographic Exploration of Software Documentation and its Cultural Implications for an Agile Startup

NICOLE TIETZ-SOKOLSKAYA, Principal Engineer, Remesh
SUZANNE WALSH, Research Consultant, Remesh

This case study offers an in-depth analysis of the foundational role of documentation for software quality, efficiency, and organizational success. It includes an inside look how a start-up overcame inefficiencies and scaling challenges by implementing a sociotechnical documentation practice, and 10 recommendations for implementation or assessment based the case study data and a wider analysis of research and expertise. Time pressures of agile workflows that prioritize writing code, coupled with rapidly scaling software and startups, create an environment where documentation often takes a back seat. But its value to organizations and engineering teams as a source of knowledge and a practice of knowledge sharing becomes clear when critical inefficiencies and even chaos to prompt an overhaul of norms and behaviors. The experience at Remesh demonstrates the value of an ethnographic approach to gain a holistic understanding of engineering and work practices and to implement positive change.

Transforming Culture: Remesh's Journey to Comprehensive Software Documentation Practices

Remesh, a SaaS research platform, has undergone a significant cultural transformation through the adoption of comprehensive software documentation practices since the pandemic. This shift has not only enhanced operational efficiency but also fostered collaboration, inclusivity, and technical excellence within the organization. This case study delves into the personal and organizational values that have emerged from interviews with Remesh engineers, underscoring the profound impact that thorough documentation has had on the company's culture. By emphasizing formality, collaboration, effective onboarding, tailored documentation, automation, and a cultural recognition of its importance, Remesh's journey towards better documentation practices highlights a strategic approach to fostering a supportive and efficient work environment.

Remesh was founded in 2014 by Andrew Konya, current Chief Science Officer. Konya realized during his PhD program the significant gap in effectively representing the collective voice of people. The initial concept was to develop a platform that could facilitate many-to-one communication, enabling large groups to engage in meaningful dialogue and represent collective responses. The company began in earnest when it joined the 2014 FlashStarts accelerator program in Cleveland, Ohio, which provided the necessary resources to build a prototype. The company secured further growth and funding through its participation in the 2015 Techstars Barclays accelerator program, where Gary Ellis, current CEO, joined the organization as a co-founder.

The platform is designed to facilitate real-time qualitative research and insights gathering, leveraging artificial intelligence and machine learning to engage with large groups of participants simultaneously. This enables organizations to gain deeper and more actionable insights from their target audience. Key features include real-time and asynchronous engagement, AI-driven analysis, scalability, versatility and a user-friendly interface. Overall, Remesh offers a powerful solution for market researchers and organizations looking to conduct in-depth qualitative research at scale, offering the ability to gather rich, real-time insights from large groups of participants with the speed of AI.

Building Structured Documentation Practices

Prior to 2019, Remesh's engineering team consisted of three members, Engineers A, B, and C, each operating within their specialized areas. The lack of formal documentation was a significant challenge, particularly after Engineer B rewrote the entire backend without any accompanying documentation. This led to a situation where Engineers A and B were indispensable for any changes or understanding of the system, as knowledge transfer relied solely on direct interactions. The absence of documented current and future system workings left Engineer C and others dependent on informal, inefficient knowledge-sharing methods, creating a bottleneck and hindering productivity.

Recognizing the need for inclusivity and organizational efficiency, Engineer A championed a guiding ethos: all team members should have access to system knowledge, both current and future. This initiative was driven by the realization that undocumented knowledge was unsustainable, especially as the team grew and individuals took time off or left the company.

As Remesh's team expanded and transitioned to remote work during the pandemic, the necessity for formal documentation became apparent. Engineer A spearheaded efforts to enhance documentation practices, emphasizing their importance due to the challenges posed by remote work. By early 2021, after securing operational support and experiencing the drawbacks of inadequate documentation, the team adopted more structured processes, including design reviews and the use of collaborative tools like Google Drive and Notion. This shift ensured that knowledge was not only documented but also easily accessible and shareable among all team members.

In the early stages, Remesh's documentation was sparse and informal, leading to inefficiencies and repeated efforts. Essential information was often kept in tools like Slack, complicating the process of referencing or explaining code behavior.

> I do maintain a lot of documentation in Slack, but it isn't in the official place where it should be. I remember when we first started, we were told to add anything we found wrong or unclear, but as a newcomer, that was quite challenging.
>
> —Engineer E1

Outdated commands and incomplete records further hindered newcomers' ability to navigate the system and understand past decisions.

The implementation of structured documentation processes marked a turning point. Engineers began systematically documenting design reviews and testing protocols using tools like Notion and TestRail, making documentation accessible to all team members and providing clear and comprehensive guidelines. The value of comprehensive "how-to" guides and detailed pull request descriptions became more accepted, which facilitated knowledge transfer and aided in code reviews. By focusing on both the "why" and the "how" behind decisions, documentation ensures that even non-technical team members can understand and contribute to basic operations, maintaining clarity and supporting effective collaboration across departments.

> I chose to use Notion for documentation, considering that the audience was not engineers. My approach included two components – I focused first on explaining why I made certain decisions and how things worked, providing links for future operations.
>
> —Engineer E2

Remesh's adoption of formal design review processes further solidified their documentation practices. Engineers systematically documented high-level design decisions, architectural choices, and technical approaches, fostering consistency and clarity in problem-solving. The transition from co-located to geographically distributed teams during the global pandemic necessitated new documentation processes to maintain efficiency. Collaborative tools like Google Drive and Notion became integral, allowing for real-time feedback and iteration. Additionally, GitHub is used for documentation to align with coding processes, demonstrating Remesh's commitment to robust documentation practices that support growth and operational efficiency.

The team's journey involved experimenting with various documentation processes and iterating based on feedback and observed pain points. The current documentation practices are well-regarded within the team, with most members satisfied with the outcomes.

> At our company, documentation is highly valued. We strive to make our code readable and our documentation clear, reflecting our care for each other's work. This practice facilitates easier replication and modification of work, fostering a supportive and efficient work environment.
>
> —Engineer E3

The engineering team and leadership recognize the importance of a robust documentation process to support the company's growth and operational efficiency, reflecting a well-evolved practice that meets the organization's needs.

Software Documentation: A Scan of the Literature

The goal of software engineering is to produce working software that solves real-world problems. Many activities are required to achieve this goal, with writing code being the most obvious. However, supporting activities like creating automated tests and writing documentation are crucial for project success by decreasing the rate of bugs, increasing confidence in planning, and lowering the probability of schedule delays. Software documentation plays a critical role in the software development lifecycle, serving as a key medium for communication, maintenance, and knowledge sharing.

Documentation can be broadly classified into technical and non-technical categories. Technical documentation includes requirement specifications, architectural design documents, source code comments, test documents, and process descriptions, all of which aim to help software engineers comprehend and effectively perform their tasks (Garousi et al., 2015). However, determining the appropriate amount and depth of documentation remains a significant challenge. Many development teams either produce too much or too little documentation, leading to incomplete, inconsistent, or outdated documents (Garousi et al., 2015). This issue is compounded by the perception that documentation is an expensive and often undervalued activity (Garousi et al., 2015). Aghajani et al. (2019) highlights the significant gap between the intended and actual use of documentation, suggesting the necessity for more integrated tools that reduce the overhead associated with documentation tasks. They emphasize the importance of embedding these tools within the development environment to streamline the process and ensure that documentation stays up-to-date and useful. Building on this, Aghajani et al. (2020) reveals that developers generally undervalue documentation, viewing it as a secondary task due to the effort required for its maintenance.

Chomal (2015) argues that clear and comprehensive documentation can mitigate risks and enhance project outcomes by providing clear guidelines and reference points. However, Chomal also notes the common tendency for documentation to become outdated quickly, especially in agile environments where iterative development cycles are prevalent. This presents a significant challenge for maintaining the relevance and accuracy of documentation over time. Das et al. (2007) emphasizes the importance of documentation in the maintenance phase of software development. They argue that thorough documentation can significantly ease maintenance by providing detailed insights into the system's design and implementation. Regular updates to documentation are crucial to ensure it remains useful and relevant, highlighting the ongoing effort required to keep documentation aligned with the current state of the software.

De Souza and colleagues (2005) explore the types of documentation essential for effective software maintenance. They argue that concise and accurate documentation is crucial for facilitating maintenance activities and point out that involving all stakeholders in the documentation process can enhance its quality and relevance. This inclusive approach ensures that documentation addresses the needs of various users and maintains its utility throughout the software's lifecycle. Importantly, agile workflows, created in response to the shortcomings of documentation-driven workflows (Tordup Heeagar, 2012), do not easily incorporate documentation practices as the goal of agile methods is to increase the quality of software through flexibility – and the 'proof' of quality is software that works as intended. Document-driven methods, such as Waterfall, have a similar aim but increase quality through the use of stringent documentation and a lengthy design process. Standards, such as ISO, and maturity models, such as CMMI, are both document-driven and leverage defined documentation as the 'proof' of quality. Neither method is perfect, and both come with strengths and weaknesses. Agile is noted for its innovation and change-embracing flow but can come off the wheels in highly complex environments. Documentation-driven methods are excellent for highly stable projects but can stifle change or innovation (Tordup Heeagar, 2012).

Rai et al. (2022) emphasize the critical role of different types of technical documentation, including system documentation, design documents, source code documentation, and test reports. These documents are crucial for software maintenance and reuse, yet they are often neglected due to time and monetary pressures (Rai et al., 2022; McBurney et al., 2018). High-quality documentation provides numerous benefits, such as better comprehension, faster bug repair, and quicker onboarding of new employees (McBurney et al., 2018). Historically, software testing was done manually, which was expensive, slow, and error-prone (DeSouza et al., 2005). Over time, the industry has shifted towards automated testing, making it ubiquitous and essential for code delivery (DeSouza et al., 2005). In contrast, creating and using documentation is fundamental yet remains in a more nascent state. Software engineers and stakeholders struggle with creating, maintaining, and using documentation, which plays multiple roles in the process of creating and delivering software. Programmers must prioritize their documentation efforts, focusing on the most critical areas of code (McBurney et al., 2018).

Documentation plays a crucial role in knowledge sharing and organizational learning, as highlighted by several studies. Cyr and Choo (2010) emphasize the importance of documentation in facilitating knowledge transfer within organizations, particularly in distributed teams. Their research finds that effective documentation practices can bridge the gap between tacit and explicit knowledge, making information more accessible and actionable for team members. Building on this, Fannoun and Kerins (2019) assess the impact of documentation on organizational learning in software engineering. They argue that comprehensive documentation captures the collective knowledge of the development team, thereby enhancing organizational learning.

Heger et al. (2022) focus on the perceptions and needs of machine learning practitioners regarding data documentation. Their study highlights the necessity of clear and detailed documentation to support the reproducibility and transparency of machine learning models. They identify several challenges, including the complexity of documenting dynamic datasets and the need for standardized documentation practices. Shilton et al. (2013) emphasize the importance of understanding the sociotechnical dimensions of values for design research, particularly in the context of documentation practices. They argue that documentation is influenced by both technical and social factors, and recognizing these dimensions is crucial for creating documentation that aligns with the values and needs of the organization. This perspective underscores the need for documentation practices that are not only technically sound but also socially informed, ensuring they meet the diverse requirements of all stakeholders involved.

The literature on software documentation underscores its pivotal role in enhancing communication, maintenance, and knowledge sharing within the software development lifecycle. Despite its critical importance, documentation practices face numerous challenges, including keeping information current, determining the appropriate level of detail, and integrating seamlessly with developers' workflows. Effective documentation can mitigate risks, improve project outcomes, and facilitate maintenance by providing clear guidelines and reference points. Furthermore, documentation's role in organizational learning and knowledge sharing is emphasized, with comprehensive documentation capturing the collective knowledge of development teams and enhancing learning processes.

Exploring Documentation Practices at Remesh through Ethnographic Methods: Insights from Participant-Observation and Interviews

Ethnographic methods, rooted in anthropology, sociology, and social psychology, offer valuable insights into understanding documentation practices within organizations. These methods involve immersive, qualitative research techniques that capture the complex interactions between people and systems. Shilton et al. (2013) argue that values in design research are not merely attributes of people or systems but emerge from the interplay between them. This perspective is particularly relevant for studying software documentation, where the values and practices of developers, users, and other stakeholders influence the creation and maintenance of documentation. Ethnographic methods can uncover the "accidental" nature of values embedded in documentation practices, as observed in the case study of documentation development at Remesh. These methods can also reveal the sociotechnical dimensions of documentation, providing a holistic understanding of how documentation practices evolve and impact software development processes.

In order to understand the impact of bringing a documentation strategy to the Remesh engineering team, the authors employed the foundational ethnographic methods of participant-observation, interviewing, and archival review. As the principal engineer for the organization, Nicole Tietz-Sokolskaya developed the documentation protocol initially adopted by the engineering division in 2020 and has greatly influenced the ways in which the protocol has morphed and expanded over the last four years. This deep knowledge has afforded a close look into the workings of documentation, including the challenges engineers have faced. Along with this expert participant-observation knowledge, the authors interviewed eight engineers currently with the organization, who also experienced the transition to a fully remote environment. These engineers were involved in implementing the protocol at varying levels, some with extensive oversight in creating how and when documentation would be used by individual teams, others with more a 'reviewer' role.

The authors developed an interview question set and followed semi-structured interviewing techniques, allowing for the freedom to pull on unexpected conversational threads that emerged. The interviews were recorded using Zoom and Google Meet software and transcribed using Otter.ai or available embedded transcription. The transcripts were checked for accuracy and uploaded into Taguette and Atlas.ti for coding. The authors were interested in experimenting with open-source software as well as traditionally available coding software in the analysis. The authors coded specifically for the themes of *collaboration, learning,* and *knowledge sharing*, but otherwise identified emergent codes upon multiple readings of each transcript. While the Remesh platform was considered as a data collection and analysis option, the small dataset (n=9) did not lend itself to optimizing the platform, which performs best with at least 15–20 participants.

Findings

Documentation at Remesh is a socio-technical practice that embodies the engineer's values. This relationship is evident in how documentation is used to facilitate communication and collaboration among team members. For example, efforts to ensure that documentation speaks the language of both market researchers and engineers highlight the importance placed on inclusivity and clarity. The iterative process of creating, reviewing, and refining documentation further underscores a value system that prioritizes thoroughness and continuous improvement.

Value: Collaboration and Inclusivity

A notable cultural shift has been in the collaborative nature of documentation efforts, particularly through the design review process, which involves stakeholders and team members in documenting detailed plans and understanding the rationale behind decisions.

> During a time when our team lacked a clear approach to tackling certain tasks, we decided to implement a design review process that had been successful at another company. This involved talking to stakeholders, like our manager, to understand the details, weighing the pros and cons, and exploring alternative approaches. Once we agreed on a solid plan, we documented the design review.
>
> —Engineer E3

This approach ensures thorough vetting and consensus before development, fostering a supportive and efficient work environment. Distributing responsibilities, such as translation tasks, among team members avoids a single point of failure and promotes effective knowledge sharing.

> To avoid a single point of failure, we distributed the responsibility among all teams, creating translation liaisons who were trained and gradually took over the process. This collaboration ensured the knowledge was shared and documented effectively.
>
> —Engineer E4

This inclusivity aids new hires in integrating quickly, reducing dependency on individual knowledge, and promoting a more collaborative culture. Engineers draft documents and seek feedback from peers and managers, ensuring multiple perspectives are considered, thereby improving the quality and accuracy of documentation. Design reviews provide structured feedback, allowing engineers to refine ideas and solutions based on collective input, enhancing the technical soundness of solutions and fostering a culture of collaboration and knowledge sharing.

Value: Learning and Development

Onboarding at Remesh is facilitated through common training and documentation practices that convey the behaviors and values of the engineering team. These practices are crucial in ensuring new team members grasp the significance of documentation and how it embodies the organization's commitment to transparency, thoroughness, and collaboration.

> When I first joined, onboarding was facilitated by small bug tickets assigned to new engineers. This helped us get familiar with the codebase through practical tasks. [The new] system architecture documentation was particularly helpful in understanding how different services relate to each other, which is essential for new hires.
>
> —Engineer E6

Engineers have developed comprehensive guides to help new hires understand the codebase, tools, and processes, enabling them to quickly become productive and minimizing the reliance on informal knowledge transfer. This effort resulted in the evolution of documentation into living documents, regularly updated and maintained by the team to ensure they stayed relevant and reflected the current state of the codebase and processes. But not everything is documented…

> In engineering, we consider technical debt when deciding what to document. If something is straightforward and easily understood, it might not need extensive documentation. However, complex processes or code that requires significant effort to understand are thoroughly documented to save time and effort in the future.
>
> —Engineer E9

The importance of documentation was further recognized through past experiences where the absence of thorough documentation led to significant issues. Consequently, the company culture evolved to regard documentation as a vital component of the engineering process. This cultural shift ensured that documentation was no longer viewed as an afterthought but as an integral part of the development lifecycle.

Value: Continuous Improvement and Adaptability

The interviews revealed a commitment to continuous improvement and adaptability in documentation practices at Remesh, evolving from manual efforts to fully automated processes.

> We improved our processes, creating centralized, comprehensive documentation to aid new hires and ensure continuity despite team changes. Proper documentation prevents knowledge loss when key team members leave and facilitates smoother onboarding for new members.
>
> —Engineer E5

Initially, documentation was manual and time-consuming, with engineers meticulously documenting their test cases and processes to ensure thorough testing and quality assurance. Over time, tools like JIRA and TestRail became central to managing tasks and test cases, ensuring that documentation remained current and relevant. Further, detailed design reviews provided high-level insights into technical decisions and helped mitigate issues during development.

> I believe our current balance of documentation at Remesh is effective. We have detailed design reviews for new features, high-level system architecture documentation, and practical guides for common issues. This approach ensures that documentation remains useful and relevant without becoming outdated or overwhelming.
>
> —Engineer E6

This iterative process of documentation and feedback ensured the team could adapt to changes efficiently, streamlining efforts and ensuring consistency and accuracy.

Value: Responsibility and Accountability

At Remesh, responsibility and accountability have become core values, with engineers taking personal responsibility for documenting their work. Creating comprehensive design documents for complex projects has ensured that new team members can easily understand the system. This practice not only validates design decisions but also provides a reference for future work, ensuring accountability and reducing the risk of knowledge loss.

> When I joined, there was little to no documentation, making it difficult to understand the codebase. Over time, I advocated for better documentation practices to ensure knowledge was shared and accessible, particularly as the team grew and went remote. This shift culminated in the adoption of a robust design review process that significantly improved our documentation quality and inclusivity.
>
> —Engineer E7

The design review process plays a crucial role in ensuring high-quality documentation and effective communication within the engineering team. It involves multiple iterations and feedback loops, helping to identify potential issues early and incorporate diverse perspectives. Design review documents serve as a key reference for engineers, outlining important design decisions and the rationale behind them. The audience for documentation is carefully considered to ensure the content is relevant and useful. For internal engineering teams, documentation is more technical and detailed, whereas for external stakeholders, it is more high-level and explanatory.

> My approach to writing documentation is guided by understanding the audience and their needs. For internal engineering documents, I might cut certain corners, while external documents for clients require more careful crafting. I start by brain-dumping my thoughts and then organizing them into a coherent structure, often using outlines and diagrams to clarify complex concepts.
>
> —Engineer E7

This audience-centric approach helps ensure that documentation serves its intended purpose effectively, whether for internal alignment or external communication.

Value: Technical Excellence and Precision

Technical excellence and precision in documentation are highly emphasized. Engineers use visual Markdown editors and detailed design reviews to ensure that documentation accurately represents the system's design and functionality.

> A recent design review I conducted focused on importing non-Remesh data into our platform for the first time. This involved translating messages into Remesh message types and figuring out how to store both the data and metadata about the upload process. The design review aimed to organize

> these details and communicate the approach to the team. Additionally, I wrote a research-oriented review to parse another software's' undocumented import format, detailing how it translates to our needs and the properties we would use.
>
> —Engineer E8

This meticulous approach fosters a culture of technical excellence, making it easier to replicate and modify work. Documentation is tailored based on purpose, with high-level overviews for general understanding and detailed technical documents for developers. For complex features or architectural changes, comprehensive design documents capture the rationale, decisions, and implementation details, preserving knowledge and providing a reference for future work.

Remesh balances the need for comprehensive documentation with the agile methodology's emphasis on working software, particularly in projects where technical precision and usability are crucial. Documentation requirements vary along a spectrum influenced by industry standards and regulatory demands, and Remesh finds itself in the middle of this spectrum. Writing documentation is recognized as a distinct skill set requiring both technical and communication abilities.

> Early in my career, I learned the importance of documentation the hard way, through experiences where the lack of documentation led to significant issues. For example, at a former employer, a critical processing script was accidentally deleted, and without documentation, it took weeks to restore functionality.
>
> —Engineer E9

The engineering team prioritizes technical excellence and precision by stipulating documentation as a core competency in performance reviews. It is not possible to be promoted without developing strong documentation skills, highlighting the organization's commitment to high-quality documentation as an integral part of engineering excellence.

Recommendations

Based on the insights from data analysis, the following recommendations are suggested for any start up or organization wrestling with software documentation:

Start with Clear, High-Level Documentation

Begin with high-level documentation that provides an overview of systems and processes. This helps new team members get up to speed quickly and understand the context before diving into specifics. Use collaborative documentation tools like Notion or Google Drive for easy iteration and feedback.

Use Design Review as a Standard Practice

Implement a design review process to document the technical approach for new features, including design decisions, alternatives considered, and potential edge cases. Ensure that design reviews are collaborative, involving feedback from team members and relevant stakeholders.

Tailor Documentation to the Audience

Customize the level of detail based on the audience, with high-level overviews for non-technical stakeholders and detailed technical documentation for engineers. Include both the "why" and "how" in technical documentation to provide context and ensure comprehensibility for readers with different expertise levels.

Maintain and Update Documentation Regularly

Establish a process for regularly reviewing and updating documentation to keep it current, especially for onboarding and design reviews. Encourage team members to update documentation as part of their workflow, ensuring it evolves with the product and processes.

Document Test Plans and Processes

Create detailed test plans and cases to guide QA processes and document feature testing. Tools like TestRail can help organize and maintain these test cases. Ensure test documentation is accessible to both developers and QA personnel, facilitating collaboration and shared understanding.

Formalize Onboarding and Knowledge Sharing

Develop comprehensive onboarding documentation covering environment setup, key systems, and processes to help new team members integrate quickly.

Create shared documentation spaces (e.g., Notion, Confluence) for team members to contribute to and access knowledge easily.

Leverage Post-Mortems and Retrospectives

Conduct post-mortems and retrospectives to document lessons learned from projects and feature implementations, informing future work. Include addendums in design reviews to document significant changes made during implementation.

Encourage Collaborative Documentation Practices

Foster a culture of collaboration in documentation by involving multiple team members in the creation and review process. Use documentation as a tool for mentoring and professional development, especially for junior engineers. Additionally, involve team members at all levels in designing documentation processes to ensure they meet actual needs and are effective. Use feedback mechanisms like 1:1 conversations, small group meetings, and collaborative platforms to gather input from all levels.

Prioritize Critical and Complex Areas for Documentation

Focus detailed documentation efforts on critical and complex codebase areas likely to cause issues or require significant effort to understand. Use documentation to mitigate technical debt, ensuring complex processes and systems are well-documented.

Eliminate Ineffective Documentation and Address Friction Points

Trust your team and eliminate documentation that isn't being done if it doesn't serve its intended purpose, suggesting a need for different solutions. Focus documentation efforts on resolving friction points between teams or in the development process to ensure it serves a clear purpose. Regularly assess and refine documentation processes based on team feedback and observed effectiveness.

Implementing one, some, or all of these recommendations will help startups and other organizations establish robust and effective documentation practices.

About the Authors

Nicole Tietz-Sokolskaya is a software engineer with expertise in performance and software architecture. She has over 10 years of experience at software start-ups and consulting. Currently, she is Principal Software Engineer at Remesh, a SaaS AI-enabled research platform, where she works on platform stability and performance, security, processes, and provides top-level leadership.

Suzanne Walsh is an anthropologist with expertise in business and health research. With over 25 years of experience in academia and consulting, she currently works at Remesh, a SaaS AI-enabled research platform, as a Research Consultant. Suzanne helps clients puzzle through sticky organizational research problems, and leverages machine learning, NLP, and generative AI to understand qualitative data at scale. She has published in high-impact journals and authors thought leadership for Remesh.

References Cited

Aghajani, Emad, Csaba Nagy, Olga Lucero Vega-Márquez, et al. "Software Documentation Issues Unveiled." *Proceedings of the 41st International Conference on Software Engineering*, IEEE Press, 2019, pp. 1199–210. *ACM Digital Library*, https://doi.org/10.1109/ICSE.2019.00122

Aghajani, Emad, Csaba Nagy, Mario Linares-Vásquez, et al. "Software Documentation: The Practitioners' Perspective." *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, Association for Computing Machinery, 2020, pp. 590–601. *ACM Digital Library*, https://doi.org/10.1145/3377811.3380405

Andreeva, Tatiana, and Aino Kianto. "Knowledge Processes, Knowledge-intensity and Innovation: A Moderated Mediation Analysis." *Journal of Knowledge Management*, edited by Gregorio Martín-de Castro, vol. 15, no. 6, Oct. 2011, pp. 1016–34. https://doi.org/10.1108/13673271111179343

Asatiani, Aleksandre, et al. "Sociotechnical Envelopment of Artificial Intelligence: An Approach to Organizational Deployment of Inscrutable Artificial Intelligence Systems." *Journal of the Association for Information Systems*, vol. 22, no. 2, 2021, pp. 325–52. https://doi.org/10.17705/1jais.00664

Chen, Ricky, et al. "Probing Documentation Practices: Reflecting on Students' Conceptions, Values, and Experiences with Documentation in Creative Inquiry." *Proceedings of the 13th Conference on Creativity and Cognition*, Association for Computing Machinery, 2021, pp. 1–14. *ACM Digital Library*, https://doi.org/10.1145/3450741.3465391

Chomal, Vikas S., and Jatinderkumar R. Saini. *Software Project Documentation – An Essence of Software Development. International Journal of Advanced Networking and Applications*, vol. 6, no. 06, May/June 2015, pp. 2563–2572.

Clarke, Steven. “What Is an End User Software Engineer?” In *End-User Software Engineering. Dagstuhl Seminar Proceedings*, vol. 7081, p. 1, Schloss Dagstuhl – Leibniz-Zentrum für Informatik (2007) https://doi.org/10.4230/DagSemProc.07081.26

Cyr, Sylvio, and Chun Wei Choo. “The Individual and Social Dynamics of Knowledge Sharing: An Exploratory Study.” *Journal of Documentation*, vol. 66, no. 6, Oct. 2010, pp. 824–46. *DOI.org (Crossref)*, https://doi.org/10.1108/00220411011087832

Das, Sumita, et al. “Understanding Documentation Value in Software Maintenance.” *Proceedings of the 2007 Symposium on Computer Human Interaction for the Management of Information Technology*, Association for Computing Machinery, 2007, pp. 2-es. *ACM Digital Library*, https://doi.org/10.1145/1234772.1234790

De Souza, Sergio Cozzetti B., et al. “A Study of the Documentation Essential to Software Maintenance.” *Proceedings of the 23rd Annual International Conference on Design of Communication: Documenting & Designing for Pervasive Information*, ACM, 2005, pp. 68–75. *DOI.org (Crossref)*, https://doi.org/10.1145/1085313.1085331

Ding, Wei, et al. “Knowledge-Based Approaches in Software Documentation: A Systematic Literature Review.” *Information and Software Technology*, vol. 56, no. 6, June 2014, pp. 545–67. *DOI.org (Crossref)*, https://doi.org/10.1016/j.infsof.2014.01.008

Fannoun, Sufian, and John Kerins. “Towards Organisational Learning Enhancement: Assessing Software Engineering Practice.” *The Learning Organization*, vol. 26, no. 1, Jan. 2019, pp. 44–59. *DOI.org (Crossref)*, https://doi.org/10.1108/TLO-09-2018-0149

Garousi, Golara, et al. “Usage and Usefulness of Technical Software Documentation: An Industrial Case Study.” *Information and Software Technology*, vol. 57, Jan. 2015, pp. 664–82. *DOI.org (Crossref)*, https://doi.org/10.1016/j.infsof.2014.08.003

Heger, Amy K., et al. “Understanding Machine Learning Practitioners’ Data Documentation Perceptions, Needs, Challenges, and Desiderata.” *Proceedings of the ACM on Human-Computer Interaction*, vol. 6, no. CSCW2, Nov. 2022, p. 340:1-340:29. *ACM Digital Library*, https://doi.org/10.1145/3555760

Herrmann, Thomas, et al. “Sociotechnical Walkthrough: A Means for Knowledge Integration.” *The Learning Organization*, edited by Alex Ramirez, vol. 14, no. 5, July 2007, pp. 450–64. *DOI.org (Crossref)*, https://doi.org/10.1108/09696470710762664

Hicks, Catherine. *It’s Like Coding in the Dark: The Need for Learning Cultures within Coding Teams*. Catharsis Consulting, white paper, 2022. https://www.catharsisinsight.com/_files/ugd/fce7f8_2a41aa82670f4f08a3e403d196bcc341.pdf

Huang, Shihong, and Scott Tilley. "Towards a Documentation Maturity Model." *Proceedings of the 21st Annual International Conference on Documentation*, Association for Computing Machinery, 2003, pp. 93–99. *ACM Digital Library*, https://doi.org/10.1145/944868.944888

Ifenthaler, Dirk, et al., editors. *Digital Transformation of Learning Organizations*. Springer International Publishing, 2021. https://doi.org/10.1007/978-3-030-55878-9

Kajko-Mattsson, Mira. "A Survey of Documentation Practice within Corrective Maintenance." *Empirical Software Engineering*, vol. 10, no. 1, Jan. 2005, pp. 31–55. *ACM Digital Library*, https://doi.org/10.1023/B:LIDA.0000048322.42751.ca

Königstorfer, Florian, and Stefan Thalmann. "Software Documentation Is Not Enough! Requirements for the Documentation of AI." *Digital Policy, Regulation and Governance*, vol. 23, no. 5, Nov. 2021, pp. 475–88. https://doi.org/10.1108/DPRG-03-2021-0047

Lethbridge, T. C., et al. "How Software Engineers Use Documentation: The State of the Practice." *IEEE Software*, vol. 20, no. 6, Nov. 2003, pp. 35–39. https://doi.org/10.1109/MS.2003.1241364

McBurney, Paul W., et al. "Towards Prioritizing Documentation Effort." *IEEE Transactions on Software Engineering*, vol. 44, no. 9, Sept. 2018, pp. 897–913. https://doi.org/10.1109/TSE.2017.2716950

Meng, Michael, Stephanie Steinhardt, et al. "How Developers Use API Documentation: An Observation Study." *Communication Design Quarterly*, vol. 7, no. 2, Aug. 2019, pp. 40–49. *ACM Digital Library*, https://doi.org/10.1145/3358931.3358937

Meng, Michael, Stephanie M. Steinhardt, et al. "Optimizing API Documentation: Some Guidelines and Effects." *Proceedings of the 38th ACM International Conference on Design of Communication*, Association for Computing Machinery, 2020, pp. 1–11. *ACM Digital Library*, https://doi.org/10.1145/3380851.3416759

Mohamed, Shakir, et al. "Decolonial AI: Decolonial Theory as Sociotechnical Foresight in Artificial Intelligence." *Philosophy & Technology*, vol. 33, no. 4, Dec. 2020, pp. 659–84. https://doi.org/10.1007/s13347-020-00405-8

Nakayama, Makoto, et al. "Agility and System Documentation in Large-Scale Enterprise System Projects: A Knowledge Management Perspective." *Procedia Computer Science*, vol. 181, 2021, pp. 386–93. https://doi.org/10.1016/j.procs.2021.01.181

Nassif, Mathieu, and Martin P. Robillard. "A Field Study of Developer Documentation Format." *Extended Abstracts of the 2023 CHI Conference on Human Factors in Computing Systems*, Association for Computing Machinery, 2023, pp. 1–7. *ACM Digital Library*, https://doi.org/10.1145/3544549.3585767

Newman, Christian D., et al. "Automatically Generating Natural Language Documentation for Methods." *2018 IEEE Third International Workshop on Dynamic Software Documentation (DySDoc3)*, 2018, pp. 1–2. *IEEE Xplore*, https://doi.org/10.1109/DySDoc3.2018.00007

Pasuksmit, Jirat, et al. "Towards Just-Enough Documentation for Agile Effort Estimation: What Information Should Be Documented?" *International Conference on Software Maintenance and Evolution* (ICSME), 2021, arXiv:2107.02420

Raglianti, Marco. "Topology of the Documentation Landscape." *Proceedings of the ACM/IEEE 44th International Conference on Software Engineering: Companion Proceedings*, Association for Computing Machinery, 2022, pp. 297–99. *ACM Digital Library*, https://doi.org/10.1145/3510454.3517068

Rai, Sawan, et al. "A Review on Source Code Documentation." *ACM Transactions on Intelligent Systems and Technology*, vol. 13, no. 5, Oct. 2022, pp. 1–44. https://doi.org/10.1145/3519312

Renzl, B. "Trust in Management and Knowledge Sharing: The Mediating Effects of Fear and Knowledge Documentation." *Omega*, vol. 36, no. 2, Apr. 2008, pp. 206–20. https://doi.org/10.1016/j.omega.2006.06.005

Robillard, Martin, et al. "On-Demand Developer Documentation." *2017 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, Shanghai, China, 2017, pp. 479–483. https://doi.org/10.1109/ICSME.2017.17

Shilton, Katie, et al. "Charting Sociotechnical Dimensions of Values for Design Research." *The Information Society*, vol. 29, no. 5, Oct. 2013, pp. 259–71. https://doi.org/10.1080/01972243.2013.825357

Tamburri, Damian A. "Software Architecture Social Debt: Managing the Incommunicability Factor." *IEEE Transactions on Computational Social Systems*, vol. 6, no. 1, Feb. 2019, pp. 20–37. https://doi.org/10.1109/TCSS.2018.2886433

Wagenaar, Gerard, et al. "Working Software over Comprehensive Documentation – Rationales of Agile Teams for Artefacts Usage." *Journal of Software Engineering Research and Development*, vol. 6, Dec. 2018. https://doi.org/10.1186/s40411-018-0051-7

Waheed, Sara, et al. "Improving Knowledge Sharing in Distributed Software Development." *International Journal of Advanced Computer Science and Applications*, vol. 10, no. 6, 2019. https://doi.org/10.14569/IJACSA.2019.0100656

Zhi, Junji, et al. "Cost, Benefits and Quality of Software Development Documentation: A Systematic Mapping." *Journal of Systems and Software*, vol. 99, Jan. 2015, pp. 175–98. https://doi.org/10.1016/j.jss.2014.09.042