THE HIDDEN COST OF DOCUMENTATION SILOS: HOW FRAGMENTED INFORMATION DRAINS YOUR ENGINEERING BUDGET

Your development team is slowed down, your customers are frustrated, new team members take a long time to onboard, and your support team is constantly banging down the door. All of these problems have the same root cause: poor documentation. This white paper shows you how to build a unified, layered source of truth that connects silos, reduces support costs, accelerates onboarding, and unlocks better communication and higher efficiency.

The Financial Cost of Fragmentation

Low-quality documentation hurts all of the stakeholders in your project. Customers and potential customers can't work out how to use the features they need, so they switch to competing products. Or they open a case with your customer support team, increasing their workload.

The support team themselves are wasting lots of time tracking down root causes for customer cases and dredging multiple sources for the information they need. Records are inconsistent,

and out of date. The customers, support staff, and developers use different terminology, so even when they find what they need, it's hard to use.

Developers trawl through source code comments and Slack chat logs to try to figure out how internal systems work. Changes take longer to review, while reviewers try to work out how the code integrates with the rest of the system. Deployments fail, because key information about what the software needs isn't available to the ops team: an updated service needs a secret that isn't in the vault, or the application tries to read configuration from a file when it's supplied in environment variables.

Meanwhile, your solution to the slowing pace of development was to hire more people, but onboarding is slow, too. There's no unified guide to the architecture to help new people orient, the information they find isn't necessarily current, and so they spend a lot of time asking their senior colleagues for whiteboard walkthroughs and help getting their development environments set up; time that takes away from working with those colleagues on adding business value.

These problems quickly add up. Developers spend less than one hour per day writing software, wasting a total of £10.4B across the UK economy as they perform other administrative tasks, including searching for information.[1] Researchers at Carnegie-Mellon University found that even when coding, developers spend nearly 30% of their time searching, navigating, and reading documentation.[2]

The Problem: Information Silos and the Slack of Babel

The organisation arrived at this point through the best of intentions. Somebody noticed that the support team needed information that isn't in the user manual, so they set up a wiki. Initially, the team diligently added relevant information to the wiki as new cases came up, but as the software evolved and the existing pages started to drift out of date, they learned not to trust it so much.

A senior developer successfully advocated for writing API documentation in source comments, but the comments focus on the details of the API methods. They don't connect the code to user goals, or to deployment considerations.

Meanwhile, the group that writes the user guides for customers is over in the marketing department, and nobody in engineering even knows what systems they're using, let alone how they keep them up to date.

Seeing the lack of unified information repositories, the organisation turned to Slack, connecting everyone together and letting them collaboratively solve problems. Now, searching chat history becomes another chore everybody has to go through to find relevant information, and because each group initially worked from their own, siloed sources, they've all created their own languages and terminology. The chat channels are a Slack of Babel: everybody's talking, but they can't understand each other.

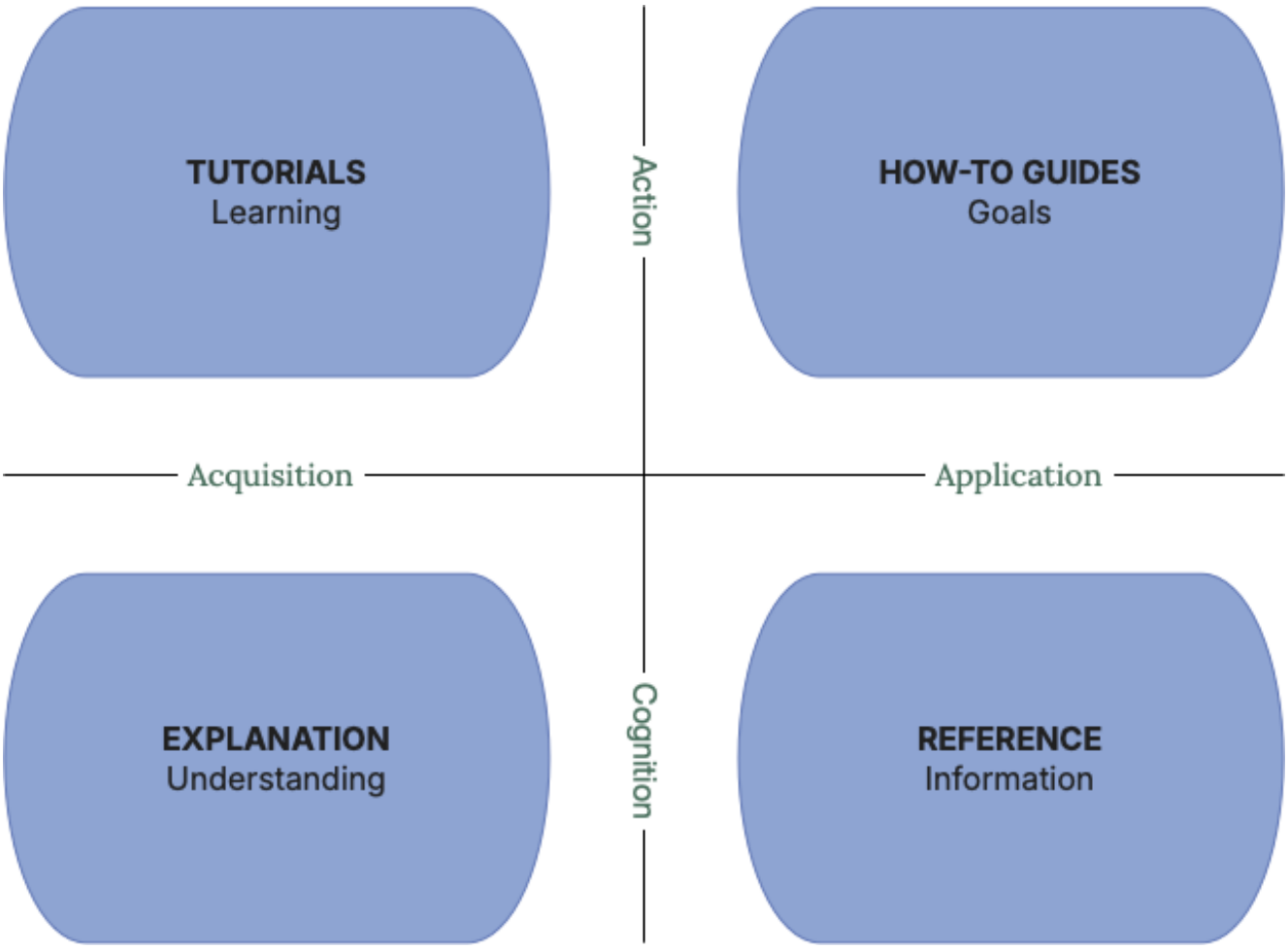
---

[1] https://www.information-age.com/developers-spending-less-than-an-hour-on-coding-123504692/

[2] https://faculty.washington.edu/ajko/papers/Ko2006SeekRelateCollect.pdf

The Solution: Unified, Layered Documentation

Diátaxis (diataxis.fr) is an industry standard approach to information architecture for technical documentation. It proposes organizing content around four distinct reader goals: Tutorials and How-To Guides provide action-oriented documentation that help someone understand how to complete a task, while Explanation and Reference documents help people develop a cognitive model of the system.



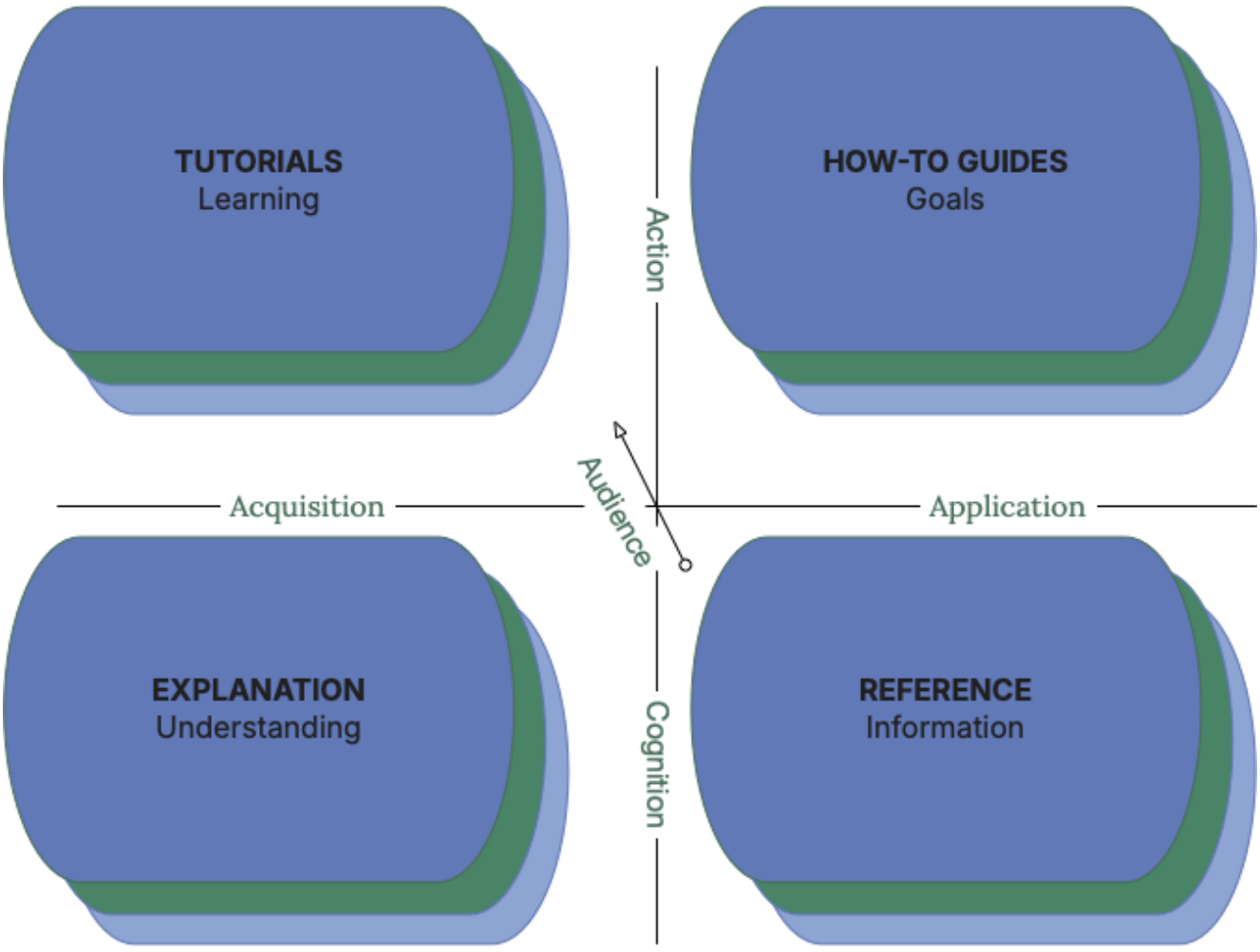
Four categories of documentation in the Diátaxis framework. Adapted from diataxis.fr.

Diátaxis helps to structure the information to maximize utility, but it doesn't address documentation silos, stale information, and distinct glossaries. To fix those problems, we introduce **Layered Diátaxis**: a unified approach in which a single documentation repository contains documentation that's organized according to the audience's *goals and roles*. The four components of Diátaxis are replicated to cover the needs of customers, operations personnel, support staff, developers, and other stakeholders.

Customers need to read the user manual and deployment guide, but don't need—and shouldn't have access to—information about engineering internals. Developers do need that

understanding of what's going on under the hood, and they need to connect it to the customers' goals so they understand the risks and values of their changes. Operations engineers need to learn how to deploy the software, and connect that with information about the code to understand the impact of updates and rollbacks. Support engineers need to be able to put the whole picture together.

Crucially, everybody needs to use the same terminology across the documentation—the ubiquitous language of your customers' problem domain. Using a shared repository, the unified, layered documentation approach supports sharing a glossary across multiple teams, and with external stakeholders.



The Layered Diátaxis approach adds different views on the documentation for different audiences, structuring documentation by the readers' goals and roles.

The Platform: Continuous Publication On an Open Source Core

We use industry-standard technologies—git version control, Sphinx, and Doxygen—and gitops pipelines to ensure that your documentation evolves with your software, and that each audience always has relevant, up-to-date information. Store the documentation repository alongside your source code so that developers can continue to use in-source docs comments,

that get published right alongside your other documentation. Deploy the audience-specific views to separate locations: hosted on your website, on community sites like GitHub or ReadTheDocs, or to an internal site protected by SSO.

The Competitive Advantage: Thoughtful AI Augmentation

Through our work on the Chiron Codex pattern language for AI-augmented software development, we know how to get significant productivity gains over larger technical writing agencies without taking undue risks. We use LLM-based tools to understand your code and review the documentation, and we understand the risks associated with intellectual property ownership and inaccurate documentation that come from relying on generated output. We work with you to understand your appetite for using AI in your documentation, and adapt to your needs.

The Business Case: Better Documentation Unlocks Engineering Efficiency

With relevant, up-to-date information, your customers find the features they need and learn how to use them, so you retain their business and they spend less time raising bug reports and support tickets.

Recalling the figures on lost productivity due to information searching shared on page 2, fixing your documentation can unlock huge efficiency savings, growing your team's engineering capacity without increasing headcount.

And when you do grow your team, your engineers can onboard new hires quickly, and create changes with less risk and less rework. The time you invest in unified, consistent documentation pays dividends in saved person-hours, reduced support costs, and happier customers who evangelize your products.

Your Next Step: The Documentation Audit

Visit https://scheduler.zoom.us/chiron-codex/fix-your-docs to book a no-obligation discovery call, where you can arrange a fixed-fee documentation audit. This audit maps out your current documentation silos, and identifies gaps in your information architecture. You receive a bespoke report with a risk-focused prioritization of your documentation needs and a roadmap to achieving unified, layered documentation that empowers your engineers and customers alike.

About the Author

Dr. Graham Lee is an accomplished software engineering author, with more than 20 years' experience improving software and documentation at Apple, Facebook, ARM, Oxygen8, and beyond. He's the author of multiple technical titles on software engineering, including *Professional Cocoa Application Security*, *Test-Driven iOS Development*, *The Python Workshop*, and *Modern Programming: Object-Oriented Programming and Best Practices*.

He gained his D.Phil. in Computer Science from Oxford University in January 2026, where he studied the role of Research Software Engineers in improving academic software outcomes.

Currently, Graham runs Chiron Codex, an initiative to help software engineers become centaurs through thoughtful application of AI-augmented tools. He also takes a deep dive through the history of software engineering, sharing his discoveries with the community on the Structure and Interpretation of Computer Programmers podcast: https://sicpers.info/podcast.