Common Technical Writing Mistakes and How to Avoid Them

Objective: spot recurring documentation defects and apply prevention habits

Scope: product docs, API references, release notes, UI copy, internal specs

Audience: technical writers, engineers, product teams, and editors

Cost: rework, support tickets, onboarding delays, safety and compliance risk

Core message: most defects are process failures, and therefore preventable

Course map: audience, structure, clarity, procedures, terminology, review, AI, action plan



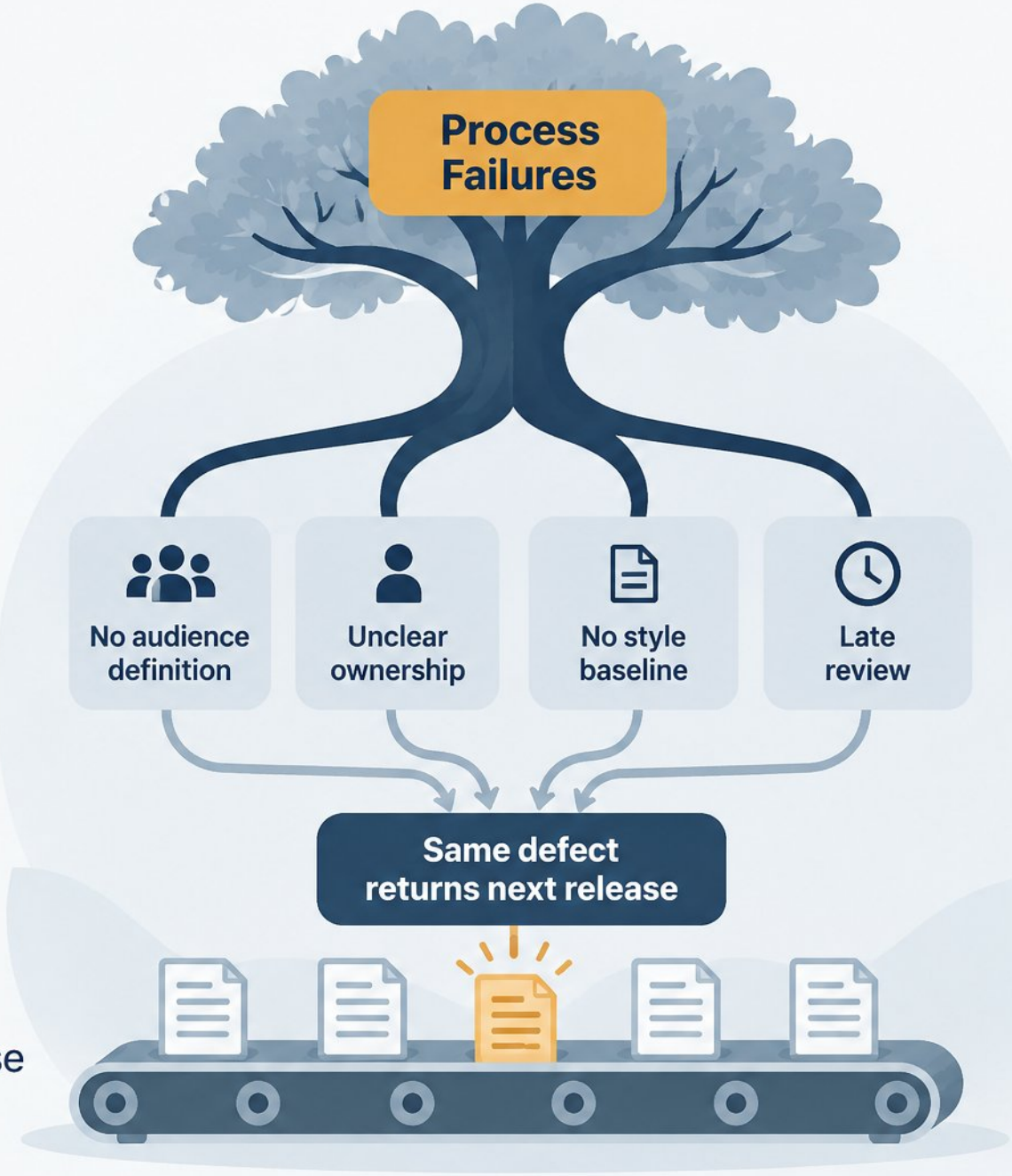
Background: Why Documentation Quality Breaks Down

- Keeping docs in sync is the top workflow challenge: 30%, rising to 43% among engineers
- Trust is asymmetric: many correct pages build it, one wrong instruction breaks it
- One in three reader questions to AI assistants could not be fully answered from docs
- 83% of active doc sites had at least one page flagged possibly incorrect or outdated
- Average site had five dead URLs—about 50 per 1,000 published pages
- Measure quality from both sides: editorial review plus post-publication reader signals



Root Causes: Most Errors Are Process Failures

- **Four root causes:** no audience definition, unclear ownership, no style baseline, late review
- **Curse of knowledge:** experts skip prerequisites, use undefined jargon, and assume context
- **Deadline pressure:** docs arrive as an afterthought, often right before release
- Lack of documentation time is flagged as important by 65% of practitioners
- Unowned pages rot; stale docs mislead more than no docs
- Diagnose the root cause first, or the same defect returns next release



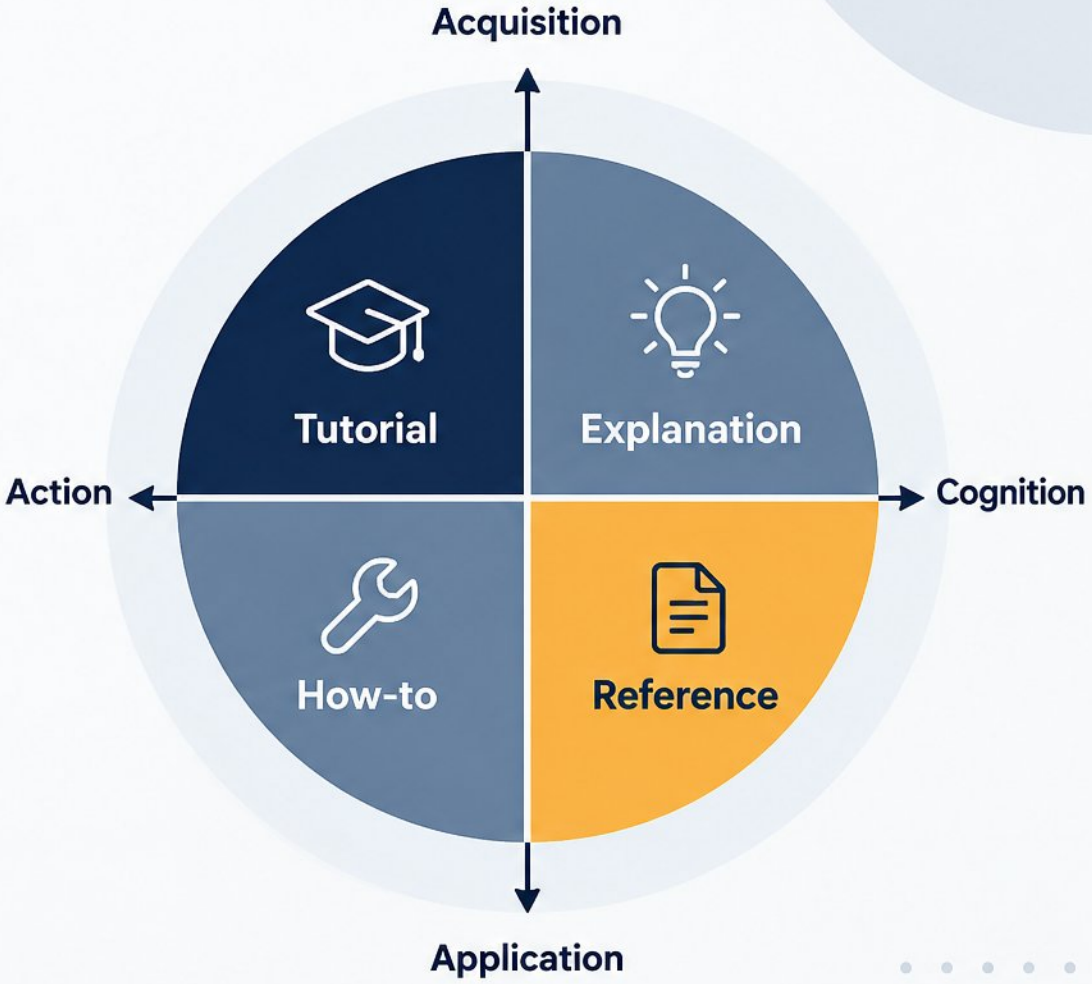
Audience and Purpose Failures

- Writing for yourself, not the reader's task
- Mixed-audience pages: definitions drift, steps skipped or over-explained
- Good docs = what the audience needs minus what they know
- State assumed starting knowledge explicitly at the top
- Pre-draft: named audience profile, task statement, purpose sentence



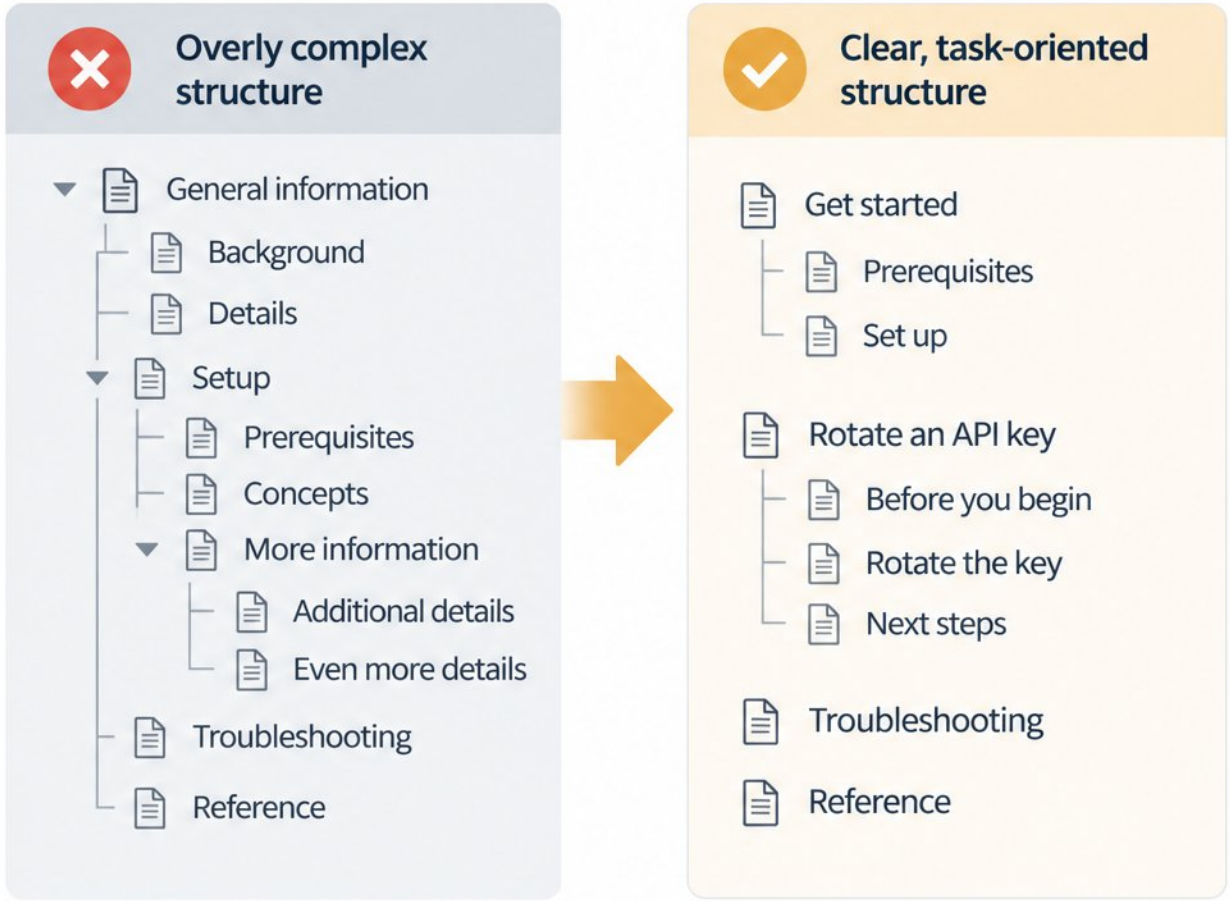
Picking the Right Content Type

- Diataxis compass: ask action or cognition, then acquisition or application
- Tutorials teach by doing; how-tos solve a problem; reference specifies; explanation gives context
- One page, one mode: a page that teaches, instructs, and specifies does all three badly
- Split mixed pages and cross-link instead of blending modes
- Explanation is most often skipped and cuts the most support load
- Reference can be generated from an API definition; tutorials and concepts need a human



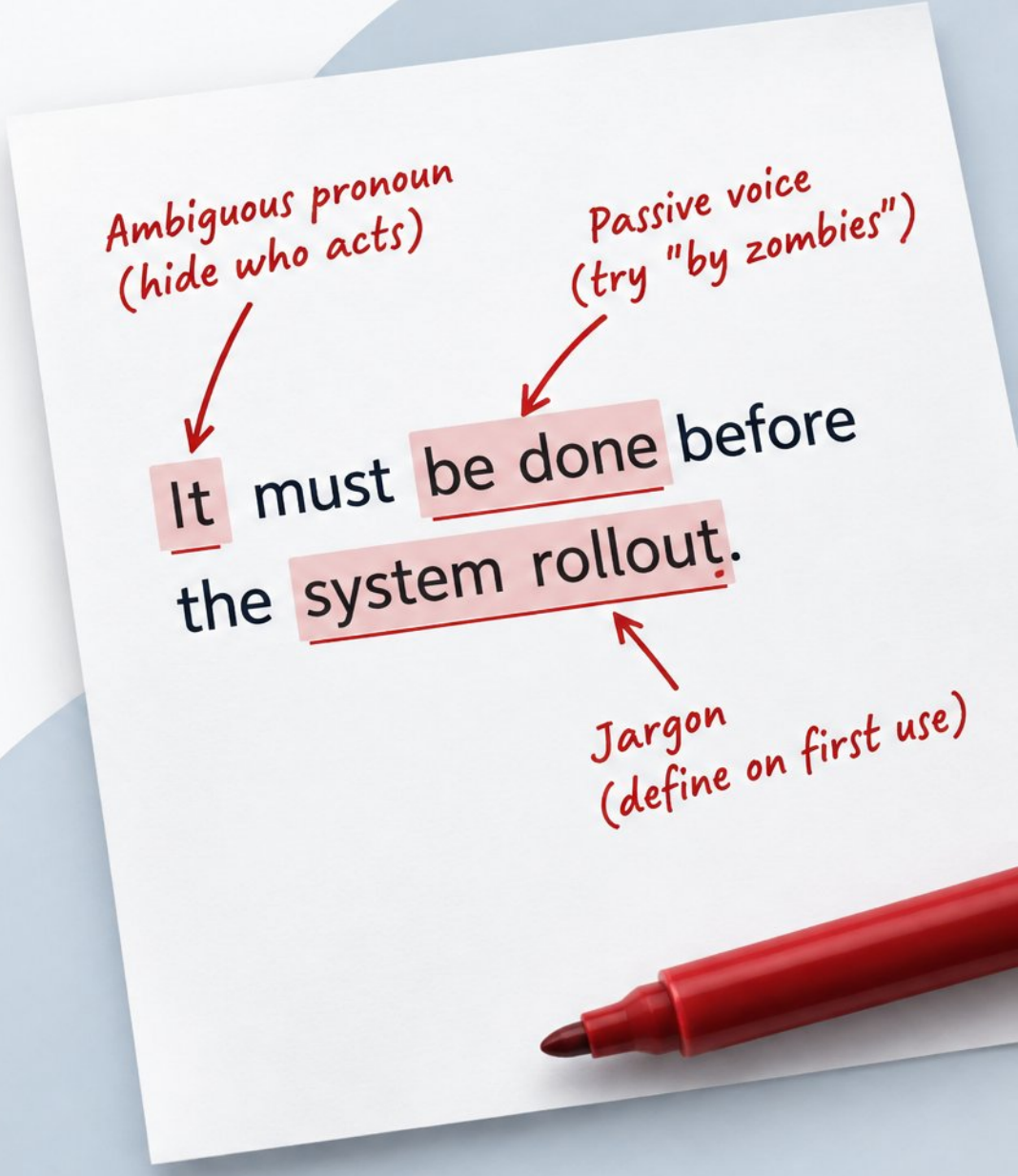
Structure and Information Architecture Mistakes

- Opening with background instead of action; readers want to get working
- Prerequisites and critical steps buried in narrative prose
- Inconsistent hierarchy, vague headings like "General information," overly deep nesting
- Kitchen-sink pages mixing setup, concept, and troubleshooting
- Write task-oriented headings: "Rotate an API key," not "Key management"
- Prevent with task-based outlines, max two subpage levels, pre-review heading audit



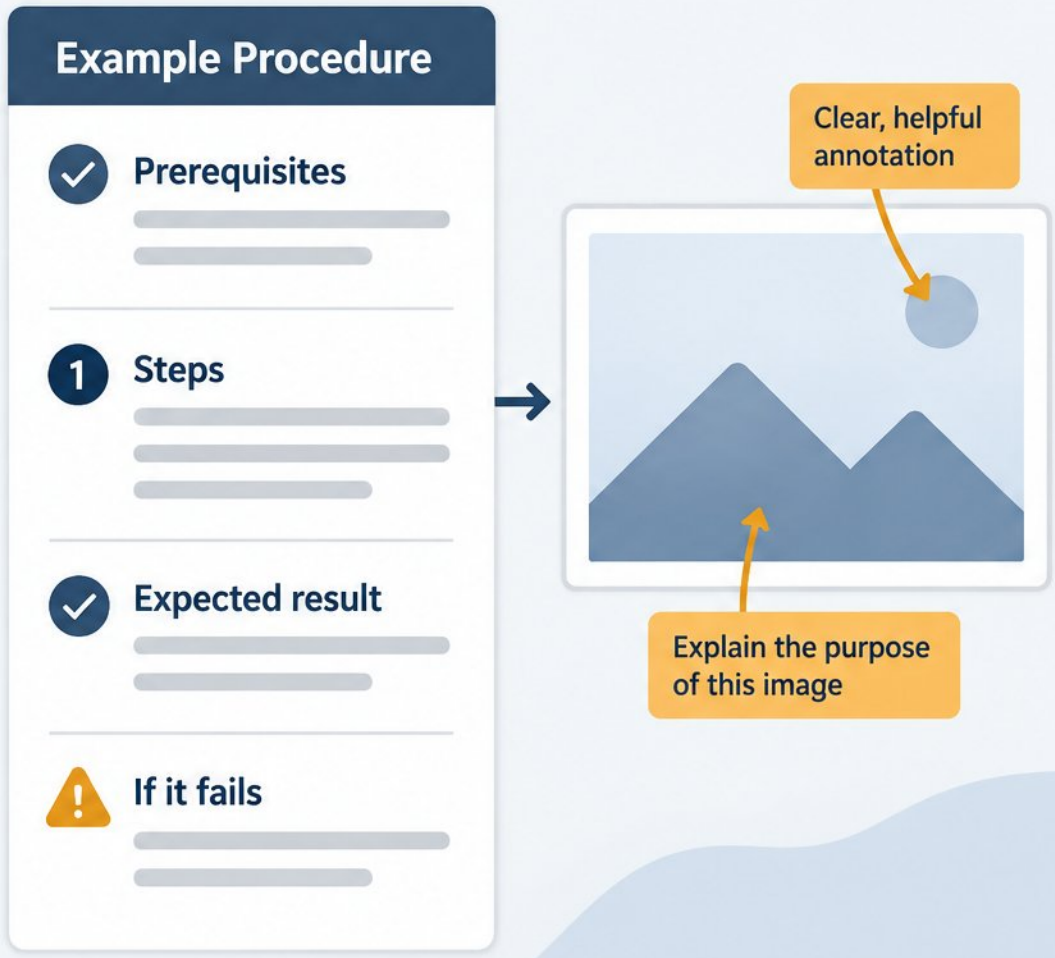
Sentence-Level Clarity: Ambiguity, Passive Voice, and Jargon

- Ambiguous pronouns hide who acts in procedural steps
- Prefer "You must do it" over "It must be done"
- Zombies test: add "by zombies" after the verb to spot passive voice
- Use verbs, not stacked nouns and nominalizations
- Define jargon and acronyms on first use
- Revise with actor-action sentences under 25 words, one idea each



Weak Procedures, Examples, and Visuals

- Procedures skip preconditions, expected results, and recovery steps
- Happy-path-only docs fail readers at the first error; add common errors and permissions
- Code samples are the most frequently wrong content type—run and test every sample
- Screenshots that decorate quickly go stale; annotate purposefully
- Accessibility: alt text on every image, no information only in images, no "click here" or "above"



Clear procedures help people succeed.

Terminology Drift and Style Inconsistency

- Same concept, three names: API key, API token, access token
- Readers wonder whether these labels mean different things
- Inconsistent capitalization of product and UI names erodes trust
- Use interface terms exactly as shown, including button labels
- Build a term list: definitions, banned variants, named owner
- Enforce with linters like Vale in CI, not checklists



Review and Collaboration Breakdowns



- Five review stages: self, technical, editorial, play testing, post-publication
- Match depth to risk: typo fix versus new API reference
- Late reviews and rubber-stamp LGTMs add cost, not quality
- Docs-as-code: same pull request, automated lint and link checks
- Exit criteria: comments resolved, named approval, correct version namespace



AI-Assisted Drafting: New Mistakes and Guardrails



Confident but unverified claims: commands that fail, options that vanished, fabricated references

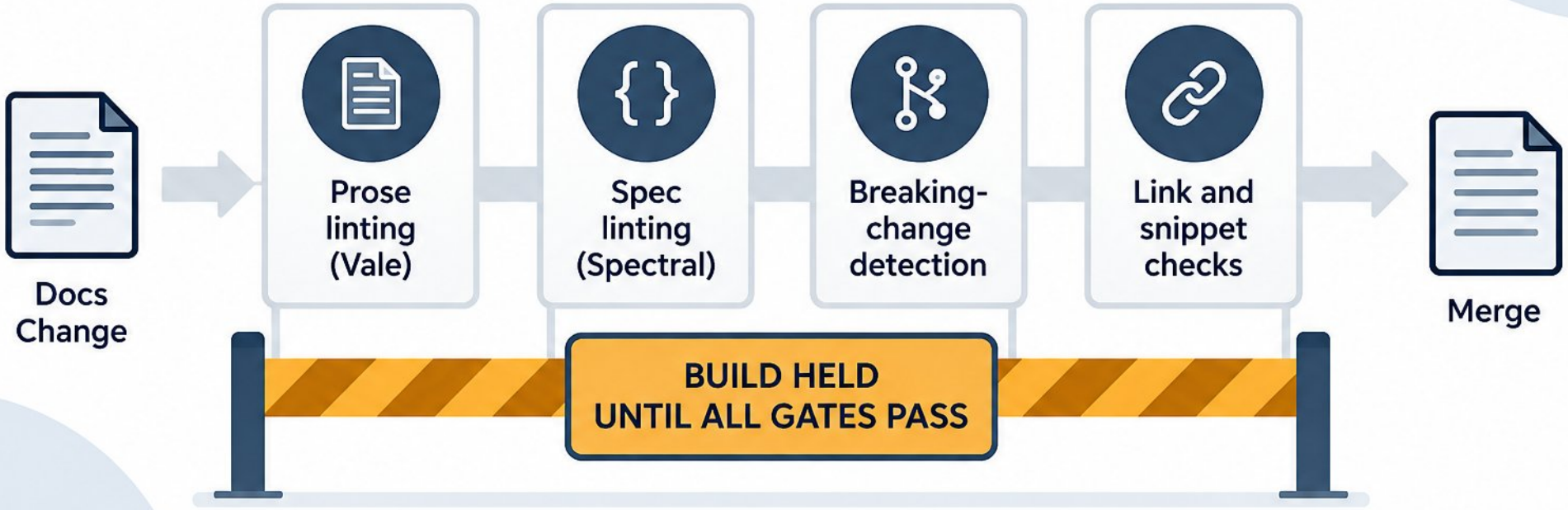
Generic filler erases product voice and exact UI terminology

Polished prose hides errors, so no one checks whether it's true

Run code, verify versions and UI labels, test every citation URL

Guardrails: human accuracy review, terminology control, disclosed AI use, no auto-publishing

Editorial Quality Gates in CI



- Checks that fail a build hold; checks that rely on noticing do not scale.
- Four gates: prose linting (Vale), spec linting (Spectral), breaking-change detection, link and snippet checks.
- Style, link, and spec checks belong in CI, not a review checklist.
- Generated reference docs stay current from a single API definition.
- Monitor requested URLs to find dead ends authors never hit.
- Smaller docs PRs get better automated coverage and more reliable review.

Prevention Playbook: Ten Mistakes, Ten Habits

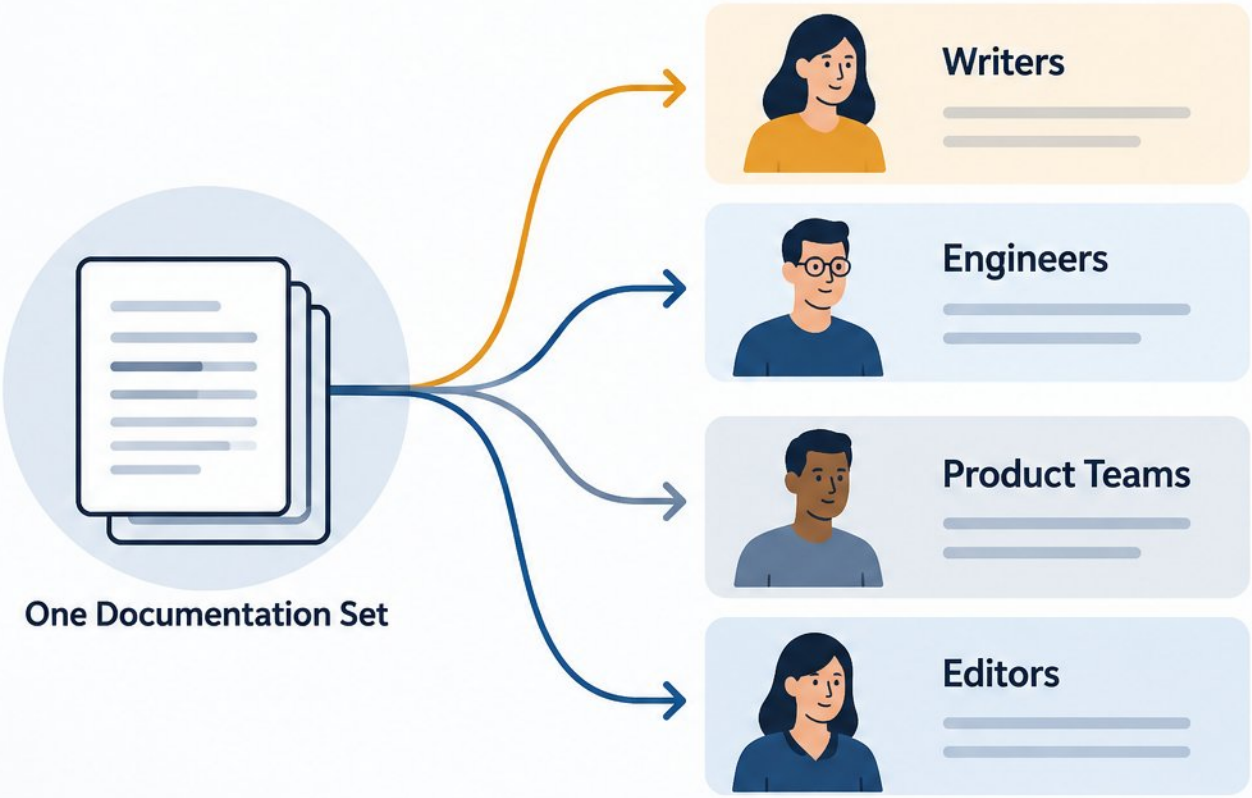
	Common Mistakes	Habits to Adopt
Pre-draft	Ten mistakes recap: pair each with one habit to adopt now	Ten mistakes recap: pair each with one habit to adopt now
	Pre-draft: audience profile, task statement, content type, outline	Pre-draft: audience profile, task statement, content type, outline
Draft	Draft: actor-action sentences, tested examples, defined terms, purposeful visuals	Draft: actor-action sentences, tested examples, defined terms, purposeful visuals
	Style guide: voice, terminology list, heading style, code-example rules	Style guide: voice, terminology list, heading style, code-example rules
Review	Review: automated lint first, then role-specific human review with exit criteria	Review: automated lint first, then role-specific human review with exit criteria
	Ownership: named owner and visible last-reviewed date on every page	Ownership: named owner and visible last-reviewed date on every page

Better content starts with better habits.

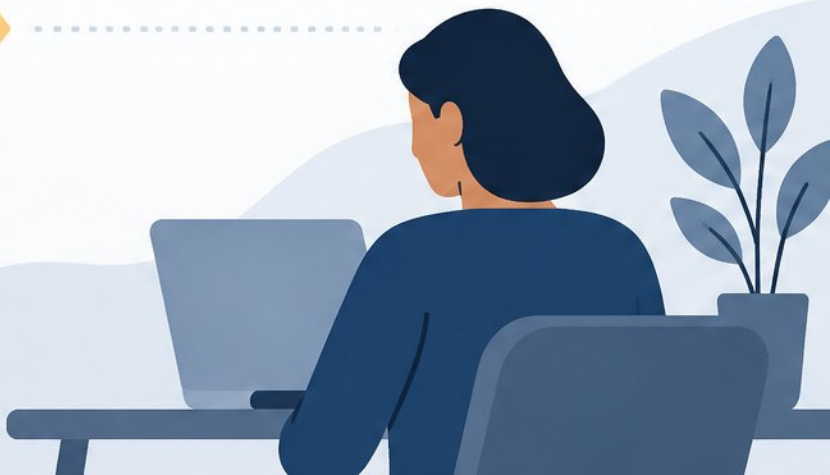


Measuring Progress and a Role-Specific Action Plan

- Measure seven quality dimensions separately; weight accuracy highest at 25%
- Track defect patterns, review-cycle time, first-pass approval, escaped defects, zero-result searches
- Writers own style baseline, glossary, and audience profiles
- Engineers test code samples and update docs in the same PR
- First 30 days: pick one doc set, add a link checker, make docs part of done



First 30 days



PersonWise.

PersonWise • AI digital-human course platform

Scan the code or click anywhere to watch this course live, with voice Q&A

personwise.ai/t/79d7e7cf/public