Product Content in Practice

By Peter Zogas

Table of Contents

Introduction

Typing indicators—those little animated dots in a messaging app that tell you the other person is writing—are nothing new. They were first used in the late 1990s and were pretty common by the time that Slack, a workplace messaging product, was released in 2013.

Slack made a small update to the typing indicator. When someone, say Bob, was typing a message, you'd see a note that said "Bob is typing." What Slack updated was what that note would read when multiple people were responding at the same time: "several people are typing" (Fig X.X).

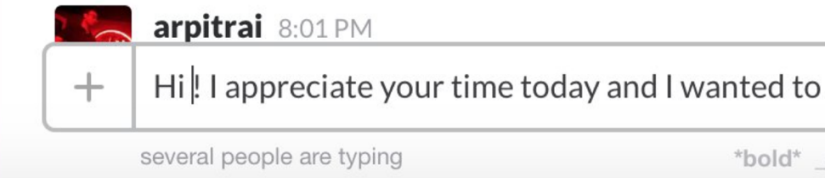


Fig X.X: Slack's indicator message for multiple people typing at once.

I remember seeing Slack's typing indicator sometime in 2014, working remotely across the country from my colleagues at the first technology company I joined. Whatever we were talking about must have seemed critical and urgent, and "several people are typing" struck me as an elegant solution for that moment. It balanced cleanly between the general and the specific. Slack didn't need to list *everyone* who was actively typing—that would be chaotic and not particularly useful—but it did tell you enough to know that communication was moving quickly. It was time to pay attention.

The copy for the typing indicator was successful in eliciting the feeling of commotion that can happen in a room full of people working together. It managed to convey a sense of lively discussion despite the fact that everyone was silently sitting at their desks in different parts of the country. And that feeling reinforced the goal of Slack as a product: to foster and improve collaboration in organizations.

At Slack, they liked the message enough that they named their blog after it. There's a novel written entirely in Slack messages titled—you guessed it—*Several People are Typing*. It's been copied, subjected to contrarian think pieces, and memed endlessly at this point, which is all pretty impressive for four words below a text box.

"Several people are typing" is an example of product content: language used in digital products that helps people use them effectively while also contributing to the feel and character of those products. Content like this is often overlooked or lumped in with all of the other types of digital content that's out there—movies in a streaming service library, articles published to a news site, comments posted to a social media app, or any other of a nearly endless list.

Those pieces of content are essential to the success of products of all kinds—you can't have a movie streaming service without movies, after all. But the content and the product are distinct. One way to think of content of this type is that it exists in a container. Swap out the content—replace one movie with another—and the product will still work.

Product content, on the other hand, is essential to the container itself. It's the collection of writing and assets that help users succeed with a product. It's an integral part of user experience that supports users as they accomplish whatever it is they want to accomplish. We can think of it like this: while the movie is the content contained in the streaming library, product content allows you to effectively use the streaming library. It's the language you use when navigating the sign-up flow for the streaming service, and it's the messaging that helps you reset your password if you forget it. It's the words and logic that allow you to navigate the library using sort, filter, or search functionality. And it's the articles and videos on the service's help site that allow you to solve any problems you're having.

Without anyone paying close attention to all of this—all of the messaging and language decisions, term definitions and action hierarchies, support documentation and error messages—a product will feel haphazard and confusing. It won't have a center, or at least not one a user could understand. Product organizations and design teams are increasingly focused on product content precisely because it's so important to a user's overall success and, in turn, driving revenue for a business. But just knowing that this work is essential isn't enough to ensure that it gets done well.

Product to Practice

If you want to create great content for a product, you need to build an effective content practice within your product organization. This is easier said than done, however, and there are two common hurdles teams face when trying to solve problems with their product content.

The first is that product content is treated as ancillary to product design. Teams often think of content as something that supports or augments a design rather than as an integral part of it. But design and content are two sides of the same coin. Just as designs depend on content, content depends on design. It's all part of the same communication effort.

Fig X.X: Apple used "slide to unlock" on the original iPhone.

Take a look at the slider on the original iPhone home screen (Fig X.X). The iPhone was first released in 2007, and it's easy to forget how novel the device was, especially when it came to using touch gestures to navigate and manipulate the screen.

Today, the slider would probably be intuitive for users, but it certainly wasn't in 2007. The "slide to unlock" label, combined with the left-to-right arrow on the slider, work together to communicate the action. It teaches us that a gesture can be continuous across an area of the screen and explains that the phone can be in both a locked or an unlocked state. The label isn't there to simply explain the slider—it's a vital part of how the overall component communicates actions and states.

This type of effective communication isn't possible if designers and writers sit on different sides of a wall. But that model is surprisingly frequent on product teams, which brings us to the second hurdle that I mentioned earlier: organizations often don't know how to enable content work to be effective at all stages of the product development process.

Product content isn't a step in the late stages of the design process; it's a layer of the product development process that requires effort at every stage. Organizations often have no structure in place for researching content or planning deliverables, and there's often not an adequate understanding of the different steps needed for content work to be effective. If there's no structure in place to enable content work to succeed, it won't.

To build a strong product content practice, you need to get over these two hurdles. Product work and design work need to be integrated, and teams need to create space to pursue content projects at all phases of the product development process. That's what I mean by a content practice, and that's what this book is about. It's a guide to crafting effective product content, but it's not just that. It's a guide to building a practice within product organizations and alongside design teams that ensures long-term content success.

The *Lorem Ipsum* Trap

It's no simple task to implement a strong product content practice, and it can be tough to know where to start. The key, I think, is to be attentive to all of the opportunities for content work to be more inclusive, collaborative, and proactive. A good example of an opportunity for this is the *Lorem ipsum* trap.

You're probably familiar with *Lorem ipsum*. It's the block of Latin text used in designs to stand in for text—it looks like real copy, but it's just filler (unless you can read Latin, in which case I bet it's pretty distracting). The whole idea is that you can drop something text-like into a design without doing the unnecessary work of writing anything relevant to the design (Fig X.X).

Fig X.X: *Lorem ipsum* is used as filler text when designing layouts.

Lorem ipsum has been used in publishing for hundreds of years, and it works wonderfully if you're only concerned with layout. But if you're using it for product design, it's a trap. It allows a designer to work with the idea of text without worrying about any of the challenges the text needs to solve in the designs.

The problem with *Lorem ipsum*, from a writer's perspective, is that it paints you into a corner. Everything you need to communicate needs to fit into the space provided, but that space was created without any of the demands or complexity of the text in mind. It'd be like designing a kitchen without any idea of how big refrigerators or ovens are. The way content interacts with design elements, functions across parts of your product, and lines up with your users' industry is vitally important to usability. *Lorem Ipsum* lets you ignore all of this and allows you to pretend that once you finally get to actually writing, you'll be able to drop everything neatly into place.

I call *Lorem ipsum* a trap because while it streamlines design work, doing so makes content creation more difficult. Rather than be a continuous stream of work, content becomes a single step in the development process. In doing so, you miss all of the complexities and potential points of confusion that you could have identified at earlier stages. What's worse, you may severely limit your ability to effectively solve these problems.

I don't want to malign designers who use *Lorem ipsum*—it is certainly a useful tool and has its place in design work. But from the perspective of product development, it signals an opportunity for organizational improvements. When I see *Lorem ipsum*, I see an instance where content is understood as something to be plugged in rather than an integral part of the overall product development process. If you start looking at your design process from this perspective, you'll probably see more instances. *Lorem ipsum* is easy to spot—so it's useful for illustrating the problem—but the same type of trap exists all over the place.

The real problem here isn't *Lorem ipsum*, it's that content work is separated from other product development work. There are a couple of themes at work here that we'll see at all stages of the product development process—the need to be proactive and anticipate content work, the importance of cross-functional collaboration, and strategies for driving consensus.

Who This Book is For

If the challenges I described with the *Lorem ipsum* trap sound familiar, this book is for you. I wrote it because there isn't much guidance for content practitioners tackling product-specific content work for the first time. And there's even less guidance for how to manage this work in the real-world contexts of product teams.

This book is for content practitioners who are building products alongside designers, product managers, and developers. Whether you're a sole content strategist on a team or part of a team, you face unique challenges in developing and maintaining successful content for a product. Beyond that, there are questions about how to get this work completed effectively alongside your team's overall working process.

It's common for design teams to recognize the need to have someone tackle the tough problems around content and decide to hire a writer or content strategist to bring order to the chaos. It's less common for those teams to have a playbook for how to integrate this writer into their team's working process.

I wrote this book to solve that exact problem. In addition to strategies for how to tackle content projects, I focus on how content work can be effectively integrated into a team's product development process. Collaboration is essential to strong product content, and we'll focus on the best ways to build successful partnerships with designers, developers, and product managers—in addition to other cross-functional partners within an organization—to create not only strong content, but a great product content practice.

What to Expect

I titled this book *Product Content in Practice* because I wanted it to account for the real-world contexts and organizational demands of doing content work in product teams. In other words, it's not just about crafting high-quality content (though that's part of it). It's about doing this work well by benefiting from cross-functional partners and being attentive to the process-related challenges that arise in product teams.

The book has three sections, each focusing on a different aspect of building a successful content practice.

The first section, chapters 1 and 2, is all about defining content work in the context of product teams and the ways they work. The goal is to build the collaborative structures needed to execute work well at the right time. We'll look at how content practitioners can build collaborative relationships with other functions and plan their work in alignment with their organization's product development process.

Chapters 3 and 4 make up the second section, which looks at how to build the knowledge and standards needed to craft great content work. Chapter 3 focuses on ways to learn about your user and problem space to better inform your content, as well as how to learn about the current state of content in your product. Chapter 4 is all about how to build and maintain content standards, with a particular focus on how they're used by various people throughout your organization.

The final section turns toward execution of content work, particularly how you do this alongside cross-functional partners, and builds on all of the planning, learning, and standards work we look at in the earlier sections. Chapter 5 discusses product writing strategies and how you can rely on them in your work broadly. Chapter 6 applies those strategies to UX writing for components, and chapter 7 dives into the specifics of labels in products and verb structures to use for actions. Finally, chapter 8 looks specifically at help sites and the full ecosystem of user education they support.

Throughout the book, I address both process-related strategies and more concrete executional techniques. My hope is that in framing common organizational challenges, exploring solution options, and looking at real-world execution examples, you'll find tools and ideas you need to build a strong content practice and do the work effectively and efficiently.

1. Where Content Fits

There's a range of roles and titles used for people who work on product content. *Content strategist* and *content designer* are popular ones, but you'll also see *UX writer*, *technical writer*, and *copywriter*. To muddy the waters a bit more, you'll sometimes see the role of *content strategist* or *content specialist* used by marketing teams who aren't working on a product at all.

The confusion around what to call someone working on product content reflects the general confusion about what the work itself actually is and where it fits in product teams. I've certainly encountered people who've expressed surprise—and once or twice even a hint of dismay—that someone had been hired just to think about words in the product. Doesn't everyone know how to write? Can't the designers just do that?

Thankfully, that's becoming less common, and there's often strong support for resources dedicated to content. It's usually the people within an organization who are most invested in creating a strong user experience who advocate for having content practitioners. They understand that there's *something* wrong with how their product is communicating with users, and they'll pull out examples to demonstrate some of those problems. What's less clear to them, however, is exactly how to put all the pieces in place to fix things. And that's exactly where you and I come in.

To build a practice to do this requires that you get clear for yourself, first, where your work fits within your organization—both with respect to who it supports and when to execute on it. Then it's a matter of helping everyone else understand the same. Whether you're working as a single content practitioner on a team or as part of a small content team, the onus will be on you to define your work and create ways to be more effective over time.

If people aren't sure what you do, how you can help, and what you need to succeed, it won't be a surprise if you can't get much done at all. Overcoming this lack of knowledge is an early step in creating a content practice. When your colleagues know exactly how you'll contribute—and what you need to contribute effectively—and understand how your work benefits users, you'll be able to do the work more effectively.

This chapter is about getting clear on how product content work relates to the people who rely on your work and to the overall process of how products are built in your organization. Having a strong handle on the broad contours of how your work fits within your organization is a prerequisite for a content practice built on accountability and collaboration.

Content Partners

The *Lorem ipsum* trap we talked about earlier is just one instance of missed opportunities to build a more collaborative content practice. It's also common for teams to tackle redundant projects or need to rework projects because of a missed connection or crossed wire. Communication, setting expectations, and building accountability are all ways to overcome silos like this.

These problems aren't at all unique to content, of course. But because of how poorly understood product content work can be throughout an organization, it can be incredibly common. Product managers, designers, and developers often aren't familiar with the work a content strategist does and where they can add value. People outside of a product team—like those in sales, marketing, and customer support—may not even be aware that your organization has dedicated product content resources.

Stakeholder is an unfortunate bit of organizational jargon. The name always makes me think of vampire hunters, the original holder of stakes, and scarecrows, who're famous for just standing around. I like the less staid and hierarchical *partner*, which seems like a better fit for the people who rely on your work and on whom you rely to do that work well. When you begin to map all of them, you'll find that there's a potentially large list of potential partners that are affected by decisions you make:

- Product managers
- Product designers
- Developers
- Researchers
- Brand designers
- Sales team members
- Quality assurance
- Customer support
- Legal team members
- Finance team members

A good way to begin building a true partnership is to ask the question, what does your work look like from their perspective?

Standards and style guidance for content is one of the primary responsibilities of content practitioners. Style guides and voice and tone guides for writing are often shared across an organization for a wide range of writing, from sales briefs to marketing blog posts and product documentation. Glossaries of terms used in a product are essential for aligning the truth of what's in the product to the way that people within your organization communicate with users—the last thing you want is your customer support team using different names for features or workflows than what your users actually see.

The research and discovery artifacts that content practitioners create—content journey maps, content audits, and parts of personas—inform the work they do. But they're also vital for product managers, designers, researchers, and developers to better understand users' language and communication preferences. These are facets of user education that're too easy to overlook. This work enables your entire product team to have a more complete picture of your users and how to help them succeed.

Much of the in-product work content practitioners do is centered on UX writing, but this is also the work that is most closely aligned with design decisions. What to communicate, and in how much detail, is as much about visual presentation as it is about the language we use, something we'll talk a lot more about later on, in chapter 5. But what you communicate to users can sometimes have legal implications, particularly for products that are heavily regulated (like those with FDA approval). You may also need to keep your language aligned with certain guidance from a finance team.

So much of the messaging in a product, from empty states and error messages to the terms chosen for key actions, consist of translating what your product is doing from a systems-perspective to a form that is comfortable for your users. To do this writing effectively means working closely with developers to ensure that the translation you do is accurate. It also means working with product managers to make sure that this aligns with what your users actually need.

Despite all of that work, the happy path doesn't always play out. At that point, it's time for course correction and finding ways to help users figure out how to overcome problems. Help site, in-product tooltips and contextual help, onboarding, and escalation paths all fall under this umbrella.

Here, content strategists lean heavily on customer support teams. Few people in an organization talk to users as frequently as customer support agents, and fewer still communicate with users when they're blocked, frustrated, and unable to get what they paid for. If product content is crafted with this context in mind, it's already left a lot on the table, and working with your customer support team is the only way to truly understand what your users are experiencing at their lowest product-using moments.

The Product Development Process

The second aspect of understanding the landscape of your organization and how content fits is to look more closely at your product development process. A good content strategist will identify challenges before they arise so that the team can avoid spot-fixes with copy. They'll do research to help better understand language that's familiar to customers and craft content user journeys to better understand where to best help users and when. They'll help designers solve design problems by suggesting approaches that emerge from thinking about communication structures differently. But that doesn't just happen—it's a result of a good understanding of how to structure work alongside that of their partners.

The exact shape of each product development process is unique. Teams need to consider who's working on a project, along with their function and skill sets. The process of adding a small new feature to a mature product will differ from one needed to build an entirely new product. You could have a large, highly standardized organization with a relatively calcified process or be part of a smaller organization with more fluidity. You may be lucky enough to have this mapped out already in clean documentation shared by product teams, or it may be all ad-hoc and on the fly. But getting a high-level sense of the steps in the process will allow you to proactively plan and prepare for content projects.

Because they're so context-dependent, I can't detail exactly what the product development process looks like for you. But luckily, product teams often follow the same general pattern, which we can break down into three stages:

1) Discover and research
2) Design and build
3) Launch and post-launch

Some processes break things down into five stages or seven, but overall they cover the same type of work. I've narrowed it down to three stages for simplicity's sake, but also because we're most focused on where product content practitioners can add value throughout this process.

Discover and research

Discovery and research are usually considered the realm of UX researchers and designers—all nebulous problem definitions and affinity mapping with lots of sticky notes on whiteboards. Content practitioners can add to this work immeasurably simply by participating, and they should. But there's a good deal of content-specific work that will allow you to execute more effectively later on.

The big opportunity here is to be able to anticipate the challenges that are going to come up and get a better sense of the scope of work you'll need to do. If you're doing discovery and research work for a brand new product, this could mean digging into the language your customers use or understanding if localization is going to be a large challenge or something you won't need to think much about. For smaller discovery and research projects, like those for an updated feature, you may have a good sense of these things already. But you may be able to use the time to understand messaging or terminology that's causing user confusion.

That means undergoing content audits to better understand your existing content, what works, and what doesn't. Or it could be crafting content journey maps to learn more about how users find the educational support they need to troubleshoot problems. If you use the discovery and research stage of the product development process for this work, you won't be on your heels when it comes time to implement solutions later on.

There are a lot of research and discovery techniques that your team can employ to learn about your users, define problems, and explore solution approaches. We can lean into the same type of techniques to ensure that your content work benefits from similar knowledge (and we'll dig more into specifics in chapter 3). Content practitioners are rarely involved in this stage of the process, but any effort you can make to change that will result in a stronger practice.

Design and build

When your team is confident in their understanding of the problem to solve, you can begin exploring solutions by prototyping. When I was making educational games for mobile, we'd do this stage extremely quickly by making wireframes on paper and testing the game with

colleagues. Later in my career with bigger design teams, designers often began by sketching on paper, but then quickly prototyped interactive versions that felt quite real and allowed for easier testing and more specific feedback.

This is where content work becomes more concrete and also where it pays to be fully integrated with design and product partners. As prototyping moves toward final designs, you can do the bulk of the UX writing and messaging around errors. Content reviews allow you to validate the work as you go, and you can gain additional feedback through testing or sharing work with sales or marketing teams.

This stage is when content usually comes up most for people, since it's when all of the language needs to be finalized. Designers want to know whether the headers should be title or sentence case. Developers want to make sure that each error thrown has a logical message attached. And PMs want to know that the link to the help site is going to work as intended.

It's important to integrate content work with your overall design process here, both to ensure that design and content are working toward the same ends, but also to facilitate useful feedback across your team. Beyond this collaboration, the design and building phase offers the opportunity to codify style decisions and best practices, as well as begin building out user success ecosystems.

Launch and post-launch

When you're finally ready to launch, it's likely that in the back of your mind you're thinking about all the great ideas you had to leave out, the shortcuts you had to take, and the documentation that needs writing. At this point, developers will have technical debt—elements of code that they'd like to clean up to make more efficient—and designers will know of design debt, all of the inconsistencies that accumulate as features are built out.

Content's debt is similar and equally important to track for improvements. You may have had to write last-minute error messages that don't conform to style or find lingering labels that should have been updated to reflect a late decision to change a name. This is the time to take note of these problems so you don't lose track of them later.

Launch is when you're able to ensure that all help site articles are ready to go and reflect the latest state of the product and to polish onboarding flows. Your partners in marketing and sales may need to be informed about any last-minute changes to the release, as well as full details for What's New updates for users. After launch, you could have time to catch your breath enough to devote more effort to shoring up style guidelines and glossaries to bring greater consistency to your product as you scale up.

Make your own map

To integrate with your organization's product development process, you need to understand how it works and who is involved in which stages. Many organizations have some version of this written out, while in others it's more of a set of norms. Teams have a huge range of how they plan out this work and track it over time, and in the next chapter we'll turn to how you can incorporate content work into that process. For now, the

important part is that you gain a high-level understanding of how all the parts—all of the people and stages—come together to ship a product in your organization.

At this point, the prospect of mapping out how your organization works may seem pretty daunting. You may be overwhelmed and scratching your head about where to possibly begin. Alternatively, you could be fired up and ready to fix every content problem your product has ever seen. But as as tempting as it may be to tackle *everything right now*, I don't recommend it. It's going to take a lot of effort to put together all of these pieces and a lot of iterations to get things working as effectively as possible.

What I do recommend is identifying a single project or feature to see through the full product development process. That probably means ignoring some problems you know exist, at least for now. It may seem counterintuitive, but it's better to focus on a set of projects that you can see from start to finish rather than trying to handle every instance of a single problem.

In talking to your partners, you may have discovered that poorly written error messages are causing your users a lot of pain. But rather than unearthing every potential error message and digging into rewrites, focus on errors in one part of the product and figure out what needs to get done at different parts of the development process to make improvements.

Don't worry—we're going to work through all the steps for how to do this type of thing. My point here is to think about your content practice from start to finish and from the perspectives of your partners. When you focus on your practice like this, you'll be able to figure out what works and what doesn't before applying it more broadly.

What exactly this looks like depends on if you're a single content practitioner or a part of a team of them. In either case, you're probably staring down more work than you could possibly handle. But by focusing on the product development process end-to-end, you'll build a practice that will make all of that work more manageable over time.

For a single content practitioner, try working with your partners to identify a good candidate for a feature or update where you can pursue projects during all phases of the product development process. Doing this will give you a deeper understanding of how to move beyond the *Lorem ipsum* trap to demonstrating the full value of product content for a user, and also how that work makes projects easier for a range of people in your organization.

For a content team looking to solve similar problems, the same principle holds: it's better to tackle the full scope of projects across the development process rather than focus completely on a single point in that process. It's still likely that you have more work than you can handle, but the solution here is to try embedding your content strategists with product teams over their full development process, rather than having them airdrop in to fix problems.

Content teams can easily fall into silos or work on agency models that cut them off from the product development process. But by pairing strategists with the teams working directly

on features, you'll give everyone more opportunities to anticipate challenges and build systems to meet them.

Conclusion

It's a relatively new phenomenon to have dedicated content strategists for products, which means taking a bit more time to explain what you do and how you work. It's not obvious to a lot of people how product content work fits in with all of the rest of the work happening on product teams. And that's just one layer of an organization. Content work is most susceptible to the *Lorem ipsum* trap the more siloed it is—there's a tendency for people to just toss things over a wall to punch up the writing. When you're in that situation, it's incredibly difficult to provide benefits to everyone and have opportunities to leverage what's happening at different stages in the product development process to create better outcomes.

The real goal in understanding the overall layout of your organization's product development process is that it allows you to build a more proactive content practice. Instead of waiting for work to come over the wall, you'll know what's coming. Instead of being caught flat footed with no time for research or iteration, you'll have the knowledge and tools you need to work effectively. Setting up this type of practice takes a bit of effort, but it's worth it in the long run. And it starts with setting up a system that makes you accountable to others and establishes strong techniques for collaboration throughout the product development process.

2. Collaboration

I've worked with many product managers who had never previously had access to a content strategist. I always had early conversations about what, exactly, I could help them with and what I couldn't. My goal was to help the PMs understand where I'd be able to lighten their load and contribute to the success of the product. This allows them to understand the work I was going to do—yes. But in helping them to better understand my work, it was also going to help them provide me with more effective feedback and input.

Just as importantly, I would clarify what I would need from them or their teams to do work effectively. It's impossible to deliver final copy if mocks aren't completed, and you can't create help documentation for a new feature without being able to access it. This would help them understand the requirements and timelines for my work, giving them more clarity in planning out the project and coordinating all the moving pieces.

My point is this: the accountability is a two-way street. It allows teams to understand what you'll deliver, the scope of the deliverable, and the timeline for delivery. It also gives teams the information you need to do all of the work well and on time. If anything breaks down—and things will definitely break down—the deliverables you created will help get things back on track or find ways to meet updated deadlines.

Strong collaborative processes don't just happen organically. It takes conscious effort to clarify goals, set expectations, and create norms for how people will work together. And what works for collaboration between content and design can be different from what works between content and marketing, or customer support, or anyone else.

In this chapter, I want to talk through concrete strategies to do this so that you can execute on product work well. [Getting beyond silos and Lorem ipsum]. This starts with planning the work by defining what you'll deliver and what you'll need from others to get it done effectively. Then we'll take a look at staging the work with others and building in space for effective review and feedback. Finally, we'll talk through the tools to use and how to use them.

Plans that Work

Planning is never perfect. Dependencies change, unexpected challenges crop up, and priorities shift. You may open your computer one morning to find that your carefully assembled Gantt chart has fallen to ruin. Don't beat yourself up—it doesn't mean your planning failed.

The great benefit of planning is that it creates the lines of communication and collaboration that you'll need to execute effectively. It's about setting expectations with others about what you'll deliver and when, along with what you need to get that work done. When things change, your plan hasn't failed. You just use what you already have to course correct, reset expectations, and keep going from there.

Every success I've had in planning work has resulted from this type of flexibility, and failures have always come when I've defined things too narrowly and pushed forward too much without input from others who rely on my work.

Deliverables

To get a handle on content planning, start thinking in terms of deliverables. It's incredibly helpful for you and your partners to have a document to reference that outlines what it is that you're going to create, its component parts, the steps involved to get there, and any help you'll need along the way. It's a way to clearly set expectations and establish communication about a project.

Over time, you'll hone in on the most useful details to include in your deliverables. But to start, think about these factors:

- Final output: what can your partners expect?
- Format: how will it be delivered?
- Intermediary deliverables: what steps will you need to accomplish in order to reach the final output?
- Reviews or feedback stages: who should be involved in providing feedback or reviewing work?
- Final sign-off: who is ultimately responsible for approving the project on completion?
- Level of effort: how long do you expect this to take?

Bring these into a format that you can share with others so that they have a full understanding of the project from your perspective (Fig X.X). Deliverables like these are incredibly useful for helping all of your partners appreciate the scope of work involved. Intermediary stages are important because others often don't have a strong understanding of what's involved in a content project outside of the final deliverable.

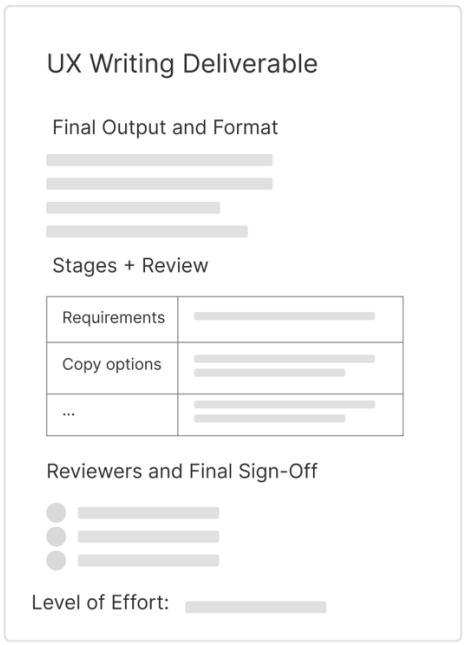


Fig X.X: A deliverable for a UX writing project maps out the output and breaks the work down into stages.

As you build out the deliverable, consider how long each of these stages is going to take. These all scale up to an overall level of effort, or the amount of time. At first, coming up with levels of effort is more art than science—I like to say the final number should make you a bit nervous but not nauseous—but over time you'll dial in a good understanding of how long certain projects should take.

Working alone and working on a team

If you're a solo content strategist, you may be pretty flexible with these or find you're creating them for every large project you embark on. Sometimes you may feel like you don't need a full deliverable document at all, if you're comfortable working with a team and have done that type of project before.

For larger teams, deliverables like these are a path toward standardization. I've had success with teams by building deliverable templates for the typical projects we were handling and then modifying those for specific situations. Each help site update for a feature release, for instance, would have a standard final output and level of effort. But for particularly complex updates, the content strategist working on the project would modify the deliverable to account for specifics. Maybe the update required that a lot of older videos be updated, or there might be a particularly complex set of workflows in play. The deliverable and level of effort could be adjusted accordingly.

As you go through a couple of cycles of making deliverables and delivering projects, you'll get much more alignment across the team on how long projects take. You can roll up to the total levels of effort for a full team over a month or a quarter to get a better understanding of your capacity and output. For a maturing content team, this allows you to better

communicate your work as a whole, set expectations, and justify any additional resources you need to complete your work.

Staging work

Deliverables do a great job of getting everyone aligned on how things *should* go during a project, but they don't address how to keep that alignment throughout a project. In one instance early on with a company, I published some documentation to customers live before the feature it supported was actually released. The feature had been delayed at the last-minute, but nobody had let me know. And in a rush to get through a list of tasks, I published without double checking. The result was that customers thought it was available and were contacting our support team about it, causing a bit of a mess for everyone.

This isn't even close to the worst mistake I've made when making information available to customers, but it was an easy one to avoid. The problem was that, while I kicked off the work well with the product team, we didn't have mechanisms for reviewing work or gating the release. It was a relatively small project from my perspective, but the product manager and I went back to make some changes in our process so that even small items like this would be fully integrated into his team's process.

While most product teams use an agile methodology for project management, rather than a waterfall one, I won't pretend to know all of the different ways that teams put it into practice (I've included a couple of resources on this at the end, if these terms are new to you). There are all sorts of models for this and tooling used to track the work, assign tasks, and facilitate communication that are beyond our scope.

What I can do is suggest some ways to connect important parts of your deliverables to an existing project management approach. No matter what your product team is using, it includes a way to break larger bodies of work into smaller tasks and indicate who is responsible for those tasks. The key for you is to understand where parts of your work align with parts of the tasks being planned and managed by your team.

If you've broken down your deliverable into intermediary steps, this is easy to do. For a UX writing deliverable like the one we talked about above, those intermediary stages line up closely with how a designer usually works (Fig X.X).

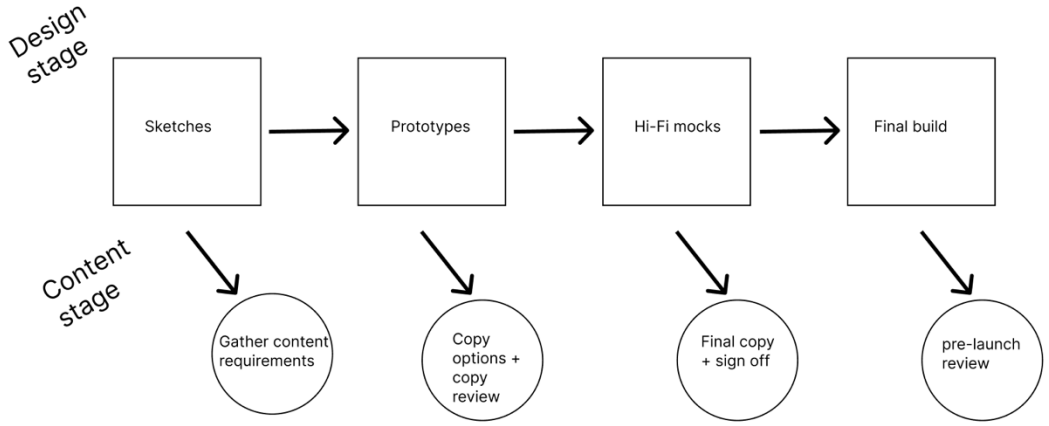


Fig X.X: Map the work that scales up to your deliverable alongside the design work it supports.

For each stage in the design process, we can map a stage in the UX writing deliverable so we know exactly when collaboration should occur. We can do the same thing with any type of project that requires input, feedback, review, or sign-offs from others. While I don't think it's always necessary to literally map this out and include it with a deliverable, it is necessary to know about these touchpoints and adhere to them throughout the project.

Content reviews and sign offs

Design review sessions are incredibly important tools for design teams. They allow designers to show their work, test ideas, and get feedback in a supportive environment. The collaborative environment allows everyone to learn from each other and, just as importantly, the designs improve.

There's an entire art to running design reviews and presenting in them, and if your team doesn't have a process established, it's worth taking a look at Donna Spencer's concise *Presenting Design Work* to help you put together a detailed gameplan. But even with a good design review process, content can slip through the cracks. It's often only discussed if something doesn't make sense.

It's much more effective to carve out specific time during design reviews to address content, even if it's not causing anyone confusion on first glance. That's how you'll get from content that's good enough to content that can really help users succeed. Set aside time during a design review to talk about content specifically, either as a specific presentation or present alongside a designer.

One of the important points that Spencer emphasizes is the need to get people in design reviews to contribute their expertise, not just their opinions. If you stick to asking people to identify unclear copy, you're more likely to get their opinion of what's unclear. It can be more effective to ask questions that drive toward the particular types of knowledge people have.

Instead of asking what's unclear, you could ask how your peers think users will respond to certain messages or terminology choices. You're much more likely to get feedback that emerges from real research and experience that way. If you ask about other potential ways to solve a problem that you've addressed, your peers may have ideas more anchored in visual thinking.

Design and content reviews follow the same principles, and they're most productive when they follow a clear structure aimed at achieving specific goals. The more effort you give to helping reviewers understand your goals and the context of your work, the better the feedback will be.

When I began working on product content after spending time in publishing and academia, I was pretty surprised to learn that sometimes no one looked closely at my work before it went live. I was used to a long process with lots of editorial steps before my writing went in

front of a reader. All of a sudden I could push something out to hundreds of thousands of people with no oversight.

Final sign-offs for product content aren't always necessary. It's often better to get a help site update pushed live to help users and correct small details later. For writing, it's also likely that you're the most qualified person to review work, and so it usually makes sense to just review it yourself.

The best solution is to include a review of any writing that appears in a product along with designs before they go live. Teams do this in different ways, but there should always be a step in the process where a designer reviews their work as it was built (not just in the mocks). You can do the same or, if you're on a content team, share that work among yourselves. Quality Assurance (QA) is usually the last step before things go live to users, and it's one last chance to double check details. Though be aware that at this stage, it's usually only worthwhile to fix blockers—things that you simply can't put in front of users. Anything else needs to wait for a subsequent update.

Tools for Collaboration

The practices I've been describing are all about getting your work through stages of the product development process while staying aligned with your product team's work. My goal with all of it is to make sure that product content work isn't siloed or just checked in on at the end of the process—it's all about how to build and foster collaboration with the rest of the team.

But the tools we traditionally use to write—word processing documents, in particular, are not particularly well suited to this work, simply because they don't handle visuals very well. It's likely that you'll still end up using a word processor for some work, but when it comes to collaborating with designers, product managers, and engineers, there are some better options.

As we move from discussing how to define and plan work, as well as build collaborative processes for it, it'll be helpful to touch on some of these options, as well as other tools that'll make your life easier.

Design Files

Design software has come a long way in becoming accessible and useful to non-designers. Tools like Figma and Sketch now have lots of functionality around commenting that make them great choices for collaboration—not just for design teams, but for product managers and developers as well. Once you get used to the tools, they're just as easy to use as word processors, and a growing ecosystem of plugins is making them even more useful for writers.

So there's no reason not to work in design files when crafting product content. The biggest benefit of writing directly in design files is that you can write in the context and immediately evaluate how well your text is contributing to the overall communication goals of the design. The same is true for designers or anyone else reviewing the work—

everything is in the proper context, so you can give feedback based on the overall user experience rather than only the content.

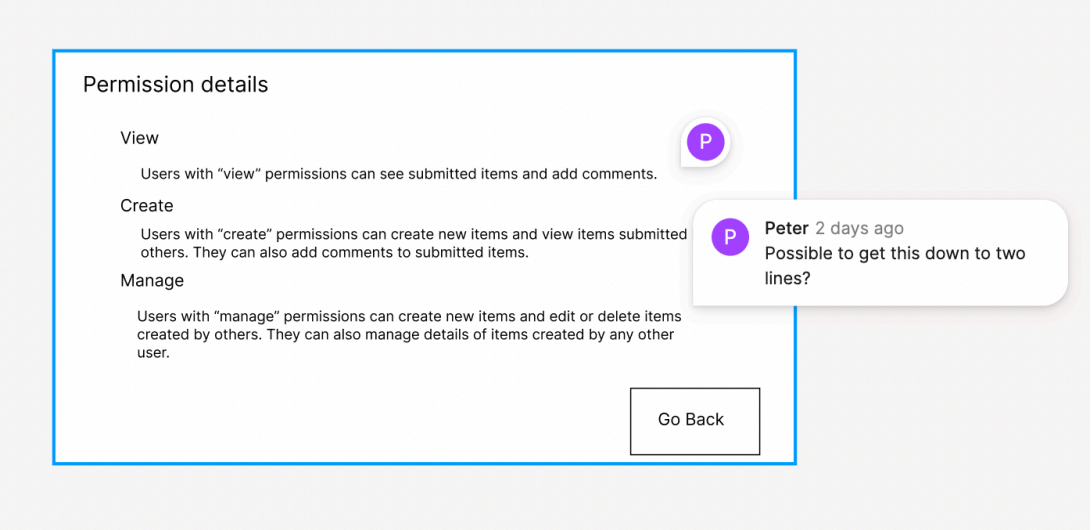


Fig X.X: Writing directly in design files allows you to best think within the context of the design. Comments are also easy to follow.

On a screen explaining three permission tiers, "view," "create," and "manage," each description covers what a user can do if assigned to the corresponding role, and it's easy to see how the content operates with the layout of the dialog (Fig X.X). We can see all of this in relation to the header, the permission level names, and the button, and we can easily see what content patterns each part is following (the header is sentence case, for instance, and the button is title case).

While the first two descriptions—for "view" and "create"—are relatively short, the third, for "manage," spills onto a third line of text. This is causing a bit of clutter, and there's an opportunity to reign in the text to be more concise. The comment in the design file calls this out and asks for a revision. A content practitioner working in the file could easily test out different options to find one that covers all of the details while also fitting more neatly into the design.

This example really only scratches the surface of what's possible when you shift more writing into design files, and it's an important way to craft strong UX writing (we'll talk a lot more about how text works with visuals later on, in chapter 5). There are even ways that writing in design files can help solve sticky problems for content and design like accounting for string-length changes due to localization by using a plugin to automatically translate copy into different languages.

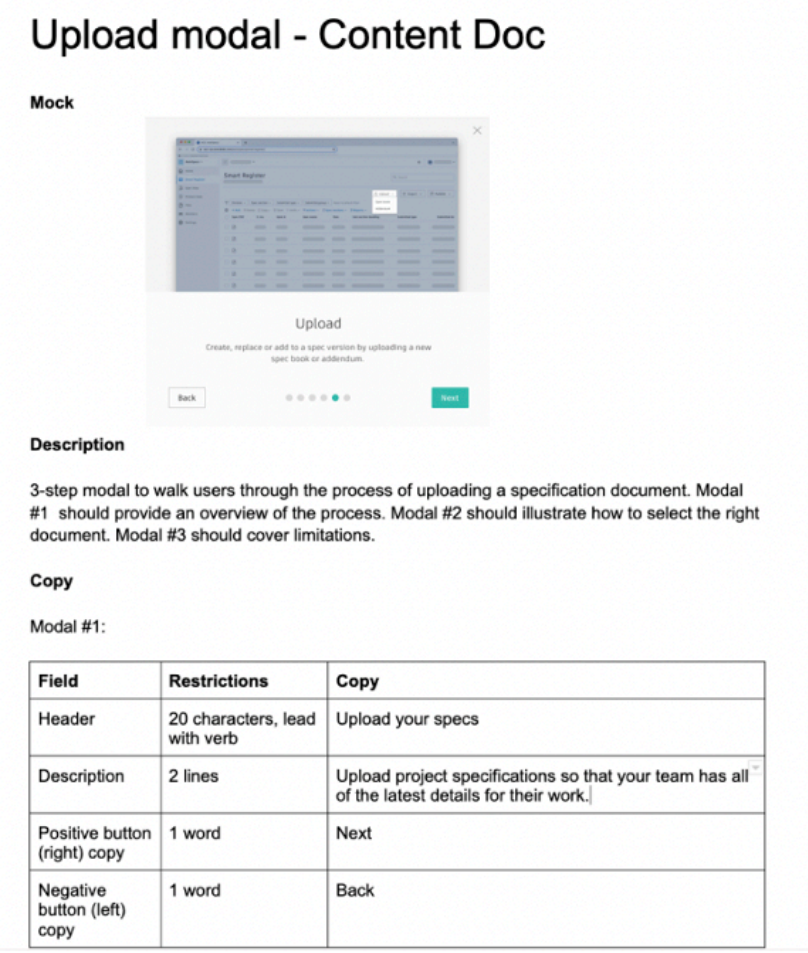
When there's no design file

Sometimes, there's simply no design file to collaborate in. This happens for all sorts of reasons. It could be that you need to rework copy for an existing feature, and the designs are long gone or never existed. Sometimes things are built using just a spec rather than a full design. And I've found it useful to work in other ways with teams in very different time

zones if we want to avoid a lot of back and forth, or if a design manager is parceling out tasks to very junior designers.

A good option for these situations is to build out *content documents*. These are stand-alone documents where you can reference designs, if they exist, while also delivering content and providing explanations and context for it.

When building a content document, the important thing is to make the document as easy for a designer to read and work with as possible. I like to describe what I'm delivering and provide any images I have. For instance, if I'm passing along updates for a screen that exists in the product, I'll provide a screenshot for reference. Then I create a table based on the field that need copy updates. That's where I list out the new copy, along with any restrictions I'm following and details I want the designer to be aware of (Fig X.X). Breaking out updated copy into their own fields is great for this, since it makes all of your work easy to copy and paste.



Upload modal - Content Doc

Mock

Description

3-step modal to walk users through the process of uploading a specification document. Modal #1 should provide an overview of the process. Modal #2 should illustrate how to select the right document. Modal #3 should cover limitations.

Copy

Modal #1:

Field	Restrictions	Copy
Header	20 characters, lead with verb	Upload your specs
Description	2 lines	Upload project specifications so that your team has all of the latest details for their work.
Positive button (right) copy	1 word	Next
Negative button (left) copy	1 word	Back

Fig X.X: A content document is a clear way to communicate content needs to a designer if you can't work directly in a design file.

You can tailor these to whatever needs you have for your project. If you're working on a content team, you may want to templatize your content documents, both to make them easier to create and so that others on your team get used to the format.

When there's no design at all

There are also instances where you won't have a design or images, and you may not even need them. I find that this is common with toasts and error messages, neither of which need to be mocked up for each instance. Imagine a user can take five actions for an item on a screen, such as add, edit, delete, download, and export. Each of those actions could succeed or fail, and your UI is meant to show a toast indicating what happened: "Item added successfully." There's no need to mock these all up, since they're so simple, but you do need to provide accurate copy that corresponds to the right options.

	Error ID	Error description	Error alert header	Header length	Error alert body	Body length	Button copy	Comment
2	UP-001	Server error - request timed out.	Submission could not be completed	33	There was a problem connecting to the server. Please try again later.	69	OK	
3	UP-002	Form submission contained an invalid character (e.g. ~, #, &, *)	Invalid character submitted	27	Your submission contains an invalid charater. Please remove all non-alphanumeric characters and try again.	107	OK	Can we give users a list of all invalid characters? Or we could link out to a help article that has this information too.
4								

Fig X.X: Spreadsheets are useful for delivering copy when there's no design and you need to track many different messages.

For these situations, it's often easiest to use a spreadsheet and coordinate directly with your developers. The goal here is the same as for content documents: you want to make things as easy to follow for the person using it. So list of all possible scenarios and all relevant details as well as any limitations that exist on the copy (like character maximums). Use some sort of unique identifier for each triggered event, and then provide the copy in the next cell over.

This is usually a very quick and easy way for developers to work with your copy (Fig X.X). And if you need an easy way to see if you're over character count, spreadsheets are great for that—use =len(cell#) to quickly output character length for a cell in Microsoft Excel or Google Sheets.

Writing tools

There are a host of other tools used in product content, either for collaboration or for you to work more effectively on your own. We've moved beyond the thesaurus and dictionary and now have a host of plugins and standalone tooling that can be effectively integrated into content work. This tendency should only speed up in the coming years.

Thesauruses and dictionaries are trivial to consult—you have them built right into your operating system. Be careful defaulting to dictionaries when settling disputes over meaning, however. I once had a developer demand that a certain word be used because of a single connotation he had in mind and defended with a dictionary. I had to point out that the word had a variety of connotations, and meanings differed between dictionaries. So it was better for us to listen more closely to what our users thought of the word, rather than a single dictionary entry.

There are also incredibly useful plugins for other tools that writers can use. Figma, for instance, supports plugins that can apply spelling, grammar, and readability checks to your product writing. And many people use something Grammarly or a similar all-in-one writing support tool to provide editorial functionality to everything they write on a computer.

Before jumping into these tools, be sure to check some of the details of how they work, as they could violate your organization's security practices. If spelling or grammar checks are happening server-side, that means all text you feed into a tool can be accessed by that tool (though specifics of encryption vary). For this reason, many companies prohibit employees from using Grammarly or similar tools.

When I first started working on this book, I planned to mention tools that rewrite or paraphrase text based on inputs you control. But at the time of writing, those tools have become effectively obsolete. We've moved pretty quickly from having some predictive text functions available in text messaging and email products to the much more robust capabilities of large language models. Those models can quickly generate their own writing, summarize text, and rewrite text based on instructions you deliver in natural language. And there's no reason to expect that developments in the field are going to slow down any time soon.

AI-based tools present the opportunity to speed up content creation. One benefit of all of these tools is that they allow you to explore different options for text very quickly. By increasing the speed of iteration, writers and designers will be able to consider more solution approaches and apply their judgments to decide on the best one. Hopefully we'll be able to increase efficiency and reduce the time spent on predictable, repetitive writing tasks, allowing us to devote more brain power to especially challenging writing problems.

Conclusion

It's much easier to espouse collaboration as a value than to actually make it happen. When you combine that with the fact that many people aren't familiar with what content practitioners do, it makes it doubly difficult to build a collaborative practice that works.

On the bright side, however, much of the work that you do as a content practitioner is a huge benefit for your partners, and often in ways they couldn't anticipate. It saves them time, solves some of their problems, and helps them clarify their own work. When you help them understand how, you may find that you run into a new challenge: they'll try to pull you in to help with everything they're working on.

One great thing about focusing on building a strong practice is that it allows you to do more with your time. And now that we have the broad contours of a good practice down—knowing your partners, where your work fits, and how to work on those projects collaboratively—we can dig into the work of product content itself. First we'll look at how to learn more about your users so you can write more effectively, and then address standards that govern your content and allow you to write quickly and scale your work over time.