The Components of Content: Meet Patrick Bosek

BY SCOTT ABEL | STC Associate Fellow

EACH YEAR MY FIRM, The Content Wrangler, surveys technical communication teams around the globe to learn how they create, manage, translate, localize, and deliver technical content. Over the past decade, we’ve spotted a number of trends—XML authoring, single-source multi-channel publishing, and the move to dynamic publishing—that have positioned the discipline of technical communication at the forefront of publishing innovation.

Modern technical documentation shops use a variety of approaches, standards, and tools to accomplish their work. Not every techcomm team or squad works in the same manner. Differences in strategy, methodology, and software platforms vary for a variety of reasons. Some teams lag behind. Others are way out in front, crafting innovative content solutions that other parts of the organizations that they serve recognize, respect, and sometimes want to replicate.

In this month’s installment of “Meet the Change Agents,” I interview technical content management maven Patrick Bosek, Co-Founder and Chief Executive Officer of Jorsek, Inc. makers of easyDITA, a component content management system used by technical communication teams.

Scott Abel: Patrick, thanks for taking the time to speak with me about the issues impacting the effective and efficient management of technical documentation content. For those readers who don’t know who you are or what you do for a living, can you take a moment to tell us about yourself and your connection to the field of technical communication?

Patrick Bosek: Thanks, Scott. I think your intro covers who I am and where I work. In terms of my connection to tech comm, it’s where I’ve worked for most of my career. I’ve been working to build tools for technical writers for about 10 years now. It’s a form of content that feels really worthwhile to me. I started my career dealing with more marketing-focused content, websites and such, but I found that dealing with knowledge-oriented content was a lot more interesting to me, so here I am.

SA: What do you think are the biggest challenges facing technical documentation teams today?

PB: Becoming something new. We’ve known for a while that tech comm was going to look different in the future, and I think that future is here (even if all of those differences are not evenly distributed across all organizations). Technical writers need to become information developers. The skills are largely the same, but the focus is more on building knowledge and information experiences.

The biggest practical change might be that information developers

In the digital age, change happens quickly. This column features interviews with the movers and shakers—the folks behind new ideas, standards, methods, products, and amazing technologies that are changing the way we live and interact in our modern world. Got questions, suggestions, or feedback? Email them to scottabel@mac.com.
can be measured on their efficacy, whereas technical writers are generally only measured on their throughput (if they’re measured at all). Because information developers generally publish content directly to the places users consume it, they can measure use and engagement. This is a new, and significant, change for an industry that has been built around non-measurable deliverables like books and hard copy PDFs. And this trend is only going to accelerate as content and knowledge systems become better connected. In the near future, you’ll be able to cross-check a user’s content interactions with their support requests and system use. When you have that level of data available, you’ll be able to truly measure the impact of information developers and that knowledge can help you drive improvements.

SA: How has content management changed over the past decade? Have there been any major innovations or technological advancements that have (or perhaps, will) change the way content management systems are designed or what they’re capable of helping us do?

PB: Wow, decade? Yeah, a ton has changed in the past decade. I think the biggest change is the shift to the cloud. Ten years ago, cloud was a small portion of the content management system (CMS) landscape, and today I think it’s the default. And it makes a lot more sense that way. From a pure cost perspective, internal IT management of CMS technology is expensive and generally unnecessary, especially when you’re looking at specialized CMS technology like component content management systems (CCMSs). And from a strategic perspective, the cloud enables connection. Legacy on-premise systems are more difficult to connect, and isolated content systems can’t offer as much value as those in the cloud.

On the topic of component content management, it’s been around for more than 10 years, but the real traction has been more recent. Today, smart companies realize that having a CMS is not sufficient, the total content ecosystem requires multiple major components, many of which broadly fall under the umbrella of “CMS.” Good CCMSs act as the true center point for knowledge in an organization. Since CCMSs are generally built around presentation-agnostic, structured content (most often XML), they can be the system of record for knowledge and the source for all end user systems. End user systems can be Web systems like documentation portals, knowledge bases, chatbots, websites, or learning management systems, but they can also be PDFs or printed copies. The reality is that almost all organizations have knowledge content deployed to many places, both internally and externally, and this will never change, because different use cases demand different delivery systems, but the core content across these systems needs to be consistent. The only way to achieve this is with a good CCMS.

SA: In our annual survey of technical communication professionals, we ask respondents what their biggest challenges are. One of the biggest challenges facing technical communication teams are inadequate or inappropriate tool sets. Words used to negatively describe technical communication tools are common, and those tools are often are characterized as “outdated,” “difficult,” “cumbersome,” and “old school.” Many of the most vocal respondents complain that vendors could provide better experiences, but it doesn’t seem to be a priority.

Your professional biography says you are a “skilled developer, thoughtful manager, and passionate customer advocate.” What lessons have you learned about serving your customers and what changes have you made to your product, easyDITA, to address their concerns about usability, maintainability, and scalability?

PB: We’re constantly iterating on easyDITA with the intention of providing a better user experience. It’s a challenging process. Technical content is itself often quite complex, thus building tools that can handle this complexity while still providing an intuitive experience to a broad range of users is challenging. In the process of doing this, I think the biggest lesson we’ve learned is how to balance adaptability and focus. easyDITA is a platform for everyone in the information development workflow. Technical writers need a different experience than SME-reviewers, for example. Our primary focus now is providing the right experience for the right user. A technical content platform doesn’t get to be a single experience system.

SA: You’ve written and presented about several of my favorite content topics over the years. I always enjoy your take on a subject. I find your views on what makes a “good content creation system” (the subject of a blog post you authored on the easyDITA blog) of interest, and I’m certain our readers will as well.

Can you share with us your thoughts on what really matters when creating a content production platform? What are the pillars of good content creation?

PB: I think the article you’re referring to (https://easydata.com/3-pillars-of-a-good-content-creation-system/) does a good job of outlining the three pillars, which are efficiency, flexibility, and scalability.

I think the term scalability as it relates to content systems needs a bit of reframing. People tend to think of scalability as how much stuff can be stored in or moved through a technology, but to me this definition is out of date. Quantity scalability is certainly still important, but when you’re considering modern, tier one applications, it should be table stakes, or an implementation detail, rather than a key deciding factor between one tool and another.
The scalability that matters is the ability to scale the whole system: people, process, and technology. When you start looking at scalability this way, you realize that the methods of collaboration and organization matter a lot more than how many gigabytes your CMS can store.

On a related note, Mark Baker left a very interesting comment on a blog post about a year ago (https://medium.com/@mbakeranalecta/david-there-are-two-fundamental-modes-of-collaboration-showing-and-hiding-3a54397e0be4). While I don’t agree with his point 100 percent, the basic premise that approaching collaboration as an open system doesn’t scale is something I strongly agree with. Collaboration needs structure to scale. Really everything needs structure to scale, it’s a basic law of nature; it’s true in skyscrapers, and it’s true in content and knowledge delivery.

**SA:** There’s been a movement among a subset of technical communicators to promote and adopt the lightweight markup language known as Markdown. As someone whose career has been based in part on helping large organizations move from creating unstructured content to crafting modules of structured, semantically rich, intelligent content using XML markup, the idea of working in Markdown seems both nonsensical and ill-conceived.

I realize creating semantically rich, intelligent XML content is not the solution to every problem, but intelligent content solves many problems, and when implemented correctly, it helps organizations do things with their content that seemed previously unimaginable. I worry that those who aim for working in Markdown are moving in the wrong direction, especially as we’re moving headfirst into the Fourth Industrial Revolution where content must be machine readable and processable, in addition to being fine-tuned for the humans we hope will consume it.

**PB:** There is so much to say on this topic. You could write an entire book about it (and some have). Before I directly answer your question, I think it’s best to reframe it a little bit. I think it’s more accurate to say people are promoting and adopting docs-as-code, rather than just Markdown. Markdown is really just the doc type that fits into docs-as-code best, so on its own, there is little reason to consider it for technical documentation. It’s basically just wiki text. Not that there is anything inherently wrong with that, it’s just to say that its place in a docs-as-code ecosystem is its strength more than any innate property.

Looking at docs-as-code using Markdown, there are definitely pros and cons. I think the docs-as-code methodology has a lot of significant benefits, but I think representing it as the only future for all of technical communication (as some people are right now) is not only invalid, it’s also dangerous.

I see this debate happening in many channels: In some places it’s Markdown vs. DITA, in some places it’s docs-as-code vs CCMSs or “traditional tools,” but it’s basically the same debate.

My perspective is that there is no “single correct format and method.” The reality is that docs-as-code attracts people in a bubble of technical writers who are not only in software-focused organizations but are often in software organizations with a start-up mentality, which creates a set of circumstances that naturally lend themselves heavily to docs-as-code.

API and single-product, low-complexity reference content is best handled with a docs-as-code methodology, but that’s a pretty small subset of tech writing, and if you try to extrapolate to all cases—even all software-focused cases—you’re going to run into problems.

The more content you have, the more semantics you need to drive high-quality search. The more varied your audience and product configurations, the more you need strong reuse and filtering mechanisms—if you care about reader experience, that is. The more places you need to deliver your content, the more consistent it needs to be, and the structure provided by DITA/XML is extremely helpful for this consistency.

Markdown and docs-as-code is great if your situation is characterized by:
- A single product
- A single audience
- A single language
- A single (or very few) target outputs
- Simple metadata requirements

DITA and CCMSs are often a better choice if your situation is characterized by:
- Multiple products sharing content
- Multiple audiences
- Many languages
- Complex output requirements
- Complex or significant metadata requirements

These aren’t mutually exclusive. Your organization may have both scenarios at different times, different points in the development and maintenance lifecycle, or different product groups.

One is not better than the other; they are just different. I think we’re doing everyone a favor by helping each other understand the right match between tools and situations, rather than approaching this subject as dogma.

**SA:** A thoughtfully chosen product name can differentiate a product from its competitors and quickly help consumers choose the product that they believe is right for them. While I don’t understand why a CMS vendor would include the name of a standard (for example, DITA) in their product
name (what happens if another standard takes hold in the future?), I love that you included the word “easy” in your product name.

As you attempt to convince those outside of the technical communication sector of the importance of creating, managing, and delivering semantically-rich intelligent content, do you think your product name gets in the way?

**PB**: *(Laughs)* Our product name has been the subject of much debate. We chose it because we wanted something straightforward, simple, and easy to remember. I think it is that. At the same time, we’ve had people choose other systems, because the name didn’t sound “heavy duty enough”—that’s a quote.

**SA**: Chatbots and conversational artificial intelligence provide big opportunities for technical communication professionals. You’ve written that you believe that chatbots are hard to get right, even harder than autonomous vehicles. Can you explain your thinking on this subject?

**PB**: Sure. Let’s start by qualifying that comparison a bit more. When I say I think chatbots might be harder than autonomous vehicles, what I’m really saying is that I think creating—and I use that term very consciously—a human-equivalent computer system to perform a task is easier for autonomous vehicles than chatbots. It’s meant to be a bit cheeky and provocative to drive a point.

Notice that I carefully use the term “creating” and not “deploying” here. Autonomous vehicles are much higher consequence than chatbots, so deploying them is infinitely harder than building them, as deploying autonomous vehicles requires vastly more effort in testing and regulating.

I believe the best quote I’ve heard about autonomous vehicles came from an engineer working on a vehicle for DARPA Grand Challenge. It’s going back a while, now, so I can’t find the link, but he basically said, “People want to call this AI, but that’s not really an accurate characterization; what we’re building is just really advanced calculators.”

This quote shaped a lot of my thinking about AI over the years. Everyone wants their thing to be AI, because they perceive that it’s worth more money when it is, but many of the things we’re calling AI aren’t really AI.

This matters, because I don’t think you need true AI to create a human-equivalent autonomous vehicle (they’ve existed for some time now), but I think you might need true AI to really produce a human-equivalent chatbot.

There are fairly well-established rules for driving a car, you can conceive of how these rules can be implemented into a system, and I don’t think you really need true AI to do it. But when you think about deploying human-equivalent chat technology, you have to remember that the rules are much fuzzier. Human interaction is incredibly complex. This complexity might require true AI, which we don’t have yet.

All of that said, I don’t think you necessarily need a human-equivalent chatbot to have a useful, chat-style content system. My real point with this comparison is to warn people that chatbot systems aren’t plug-and-play, they’re not magic, and they require a lot of thought and work to get right.

**SA**: I’m afraid we’re out of time. Thank you on behalf of our readers for taking a few moments to share a little about yourself and your views on technical communication content.

**PB**: Any time.