Managing Documentation Projects in a Collaborative World

Larry Kunz

The traditional documentation development process, which remained more or less unchanged for decades, is changing. Two trends, collaborative authoring (or Web 2.0) and Agile, are hastening this change and are forcing documentation project managers to revamp the way we manage projects.

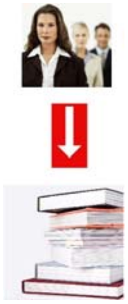
Fortunately, these trends share much in common. Some of the adjustments we're making in our processes can be applied on all projects, not just those that use Web 2.0 and Agile. Content strategy—and the role of the content strategist—emerge as vital new concepts.

THE TRADITIONAL PROCESS

Teaching in the Technical Communication certificate program at Duke University, I've told my students for years that the document development process has remained essentially the same throughout my career. Despite substantial changes in our working environment, in tools, and in media, the basic process—plan, research, write, publish—hadn't changed. And, I assured my students, it wouldn't change in the foreseeable future.

For the documentation project manager, most projects were characterized by long development cycles. After the documentation plan was finalized and agreed to by all interested parties, a months-long process began in which writers collected information, primarily from subject-matter experts (SMEs), and created static documentation products. At the beginning of my career, these products were almost always user manuals.

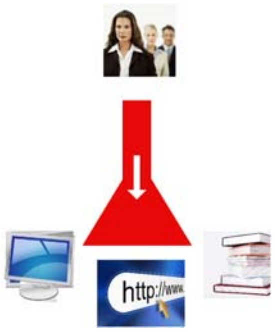
The traditional process looked like a pipe through which information flowed from the SME and arrived in the customer's hands in the form of a user manual.



HOW THE PROCESS IS CHANGING

A few years ago, however, that process began to change. Project managers, by and large, worked with the same long development cycles to which they were accustomed. The writer's primary source of information was still the SME.

The process, however, looked like an inverted funnel. Information still flowed from the SME to the customer, but the customer received it in different formats: embedded help systems, web-based tutorials, and the tried-and-true user manual.



In the last couple of years, the pace of this change has accelerated. The number of output formats is growing. The line between technical documentation and other kinds of company-produced information, like marketing white papers, has blurred. More significantly, writers are now receiving content from sources outside the enterprise.

Content is now coming from a *community* of contributors instead of from a relatively small number of SMEs. The community often includes customers who might not even understand the community to which they belong and who doubtless wouldn't consider themselves to be part of the product documentation team. They simply responded to a feedback link on the company's web site. Or they saw a question in a forum and decided to post an answer.

Today, a depiction of the process looks like a double funnel: information flows in from a variety of sources, of which the SME is only one of many. Simultaneously, the information is distributed in a growing number of output formats, sometimes even reaching the customers through social media outlets like Twitter feeds and Facebook fan pages.

http://www
FAQ
twitter
facebook

I no longer tell my students that the document-production process isn't changing.

TRENDS THAT ARE CHANGING THE PROCESS

The upending of the traditional development process can to a large extent be attributed to two new trends: collaborative, community-based authoring, or Web 2.0; and Agile, in which long development cycles are replaced by short, self-contained "sprints."

Collaboration or Web 2.0

"Web 2.0," an informal term, refers to applications on the web that facilitate collaboration and information sharing. In Anne Gentle's words, "Web 2.0 involves embracing user-created content and the communities that emerge around that content" (note 1).

When documentation is published on Web 2.0 applications, information comes into the process from many more sources, including from customers and others outside the enterprise who have an interest in the product. Information may be published in an incomplete or preliminary form, after which it is updated frequently. Documentation reviews are less formal, more frequent, and smaller in scope (that is, each review covers less material).

With the explosion of content sources comes the need for a *content strategy*, which Kristina Halvorson defines as "recommendations about how to create, deliver, and govern web content" during what she calls the *web content lifecycle* (note 2).

Creating content might mean that content originates in the technical writing team as in the past. But it can also mean assembling and assimilating content from many sources, both inside and outside the organization.

Delivering content means selecting from the wide range of delivery formats that are available. These choices are influenced by the kind of content and the needs of the customer.

Governing content is something we didn't need to consider before. Governing involves several tasks: doing a content inventory to determine what already exists and how much of it is useful; monitoring the community's contributions to ensure that everything meets the organization's standards for style and format; and maintaining the content to ensure that it remains current and useful.

In fact, Halvorson identifies no less than 15 steps in the content lifecycle (three of which are "revise"; note 3). As Halvorson points out, this is many more steps than were in the traditional content development process: design, create, revise, approve.

Rahel Bailie adds that content strategy "includes aligning content to business goals, analysis, and modeling, and influences the development, production, presentation, evaluation, measurement, and sunsetting of content, including governance." It is not, however, the same as project management because it concerns itself with strategic questions rather than tactical ones (note 4).

The person responsible for doing all of this work is usually referred to as the *content strategist*. I'll describe this emerging role in detail later in this paper.

Agile or "Just in Time" Development

Agile began as a software methodology, but the same principles are gaining acceptance in other industries as well. Agile emphasizes flexibility: rather than long, monolithic product development cycles, Agile projects are developed in short "sprints" with great attention paid to customer requirements and a realization that the requirements can change on short notice.

The Agile Manifesto, set forth in 2001, stipulates four core values:

> Individuals and interactions over processes and tools
>
> Working software over comprehensive documentation
>
> Customer collaboration over contract negotiation
>
> Responding to change over following a plan

"[W]hile there is value in the items on the right," the Manifesto says, "we value the items on the left more" (note 5).

As a result, an Agile project is usually characterized by:

> Short "sprints" in place of long development cycles.
>
> Products and documentation that are considered "just good enough" to ship to customers—they aren't necessarily tested and edited thoroughly.
>
> Small, tightly knit development teams on which technical writers are considered to be full members. In scrum meetings, documentation problems can be addressed and resolved quickly (note 6).

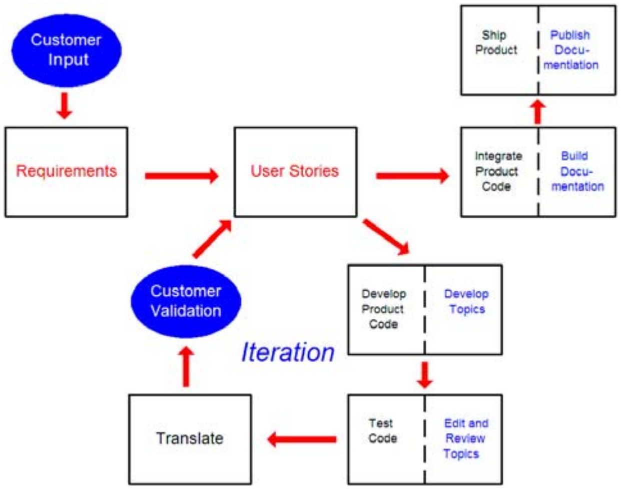
Anne Gentle's "Just Write Click" blog (note 7) offers a Top 10 list for managing the documentation on Agile products. While some of Anne's tips are offered tongue-in-cheek, she includes several valuable ones, including these:

> Only deliver things that an actual customer would find useful.
>
> List and prioritize all tasks, large and small, that get you incrementally closer to your goals.
>
> Understand the business goals. Clarify when needed by asking questions and seeking the details.
>
> Deliver something that the team considers to be done, shippable, and customer- ready.

For me, the most important piece of advice is the last one. At every stage of the process—or in Agile terms, at every iteration—my team delivers information that's "done, shippable, and customer-ready." No more rough drafts that we'll polish later on. But, conversely, no more emphasis on making sure that every detail is perfect.

This approach—carving the long process into a series of smaller processes that are complete in themselves—is a key distinguishing feature of Agile compared to other processes.

In this depiction of a well-run Agile project, program code and documentation are developed in parallel, looping through multiple iterations that are based on task scenarios or "user stories." At the end of any iteration, product code and documentation can be built and then published for customer use.



WHAT THESE TRENDS MEAN TO THE DOCUMENTATION DEVELOPMENT PROCESS

Both trends, Web 2.0 and Agile, bring similar sets of changes to the traditional documentation development process:

> The writing work is often fragmented. Rather than working together as a team to produce a well-defined set of information, writers now produce chunks of content, often working remotely from each other. (They might not even know each other.)
>
> Instead of being thorough, carefully regulated, and comprehensive, reviews are often ad hoc and in many cases involve only one SME.
>
> Schedules have to be flexible to accommodate new information sources and new user stories.

Perhaps the most significant change to the process is reflected in the emerging role of the content strategist.

THE ROLE OF THE CONTENT STRATEGIST

The content strategist must consider the big picture—handling all of the information that comes from the community and aligning it with the organization's values and strategies in a way that meets the needs and expectations of the customers.

Halvorson identifies three key planning activities for which the content strategist is responsible (note 8):

Audit: Take the time to figure out what content you have, where it's coming from (including nontraditional sources like blogs), and whether it's useful.

Analysis: Define objectives—including what will constitute success—and get all stakeholders to agree with them.

Strategy: Make specific, actionable recommendations, and describe what effect each recommendation will have on the organization.

In *Conversation and Community,* Anne Gentle identifies two related roles: the *content curator*: someone who takes inventory of what content is on hand, like a curator at an art museum, and then decides which pieces to display and how to arrange them; and the *community manager*: someone who builds the community by helping each contributor find his or her place in the community and by providing tools and processes to expedite the contributions (note 9).

CHALLENGES WE STILL FACE

As we move forward in an Agile, Web 2.0 world, we're still looking for the ways to assimilate elements of the traditional processes with the new processes. I summarize a few of them here, in hopes of starting a conversation within the profession about the best ways of approaching each of these issues.

Editing

The editing task traditionally was listed as a distinct item on the schedule and was comprehensive in nature, covering an entire manual or library of manuals. Web 2.0 and Agile have forced major changes to this model. Today editing is usually done in small stages and covers a relatively small set of topics rather than an entire set of information.

In the past, project managers have sought whenever possible to include their editors on the writing team. This is often impractical, however, in Agile environments where small, close-knit teams are the rule. As a result, the editor might not be well-versed in the technical aspects of the product and might not know what the project team expects of her.

Without the day-to-day presence of an editor, the content strategist takes on the vital role of setting overall guidelines for terminology and style. This person must provide a fairly detailed, yet easily understood, framework or template against which new material can be checked to ensure that it fits the overall content strategy.

Additionally, because the editor is no longer a daily presence on the project team, the importance of style guides is magnified. They help ensure that all writers follow the same rules of terminology, phraseology, and punctuation as they write the various pieces of the documentation package.

With the ascendancy of Web 2.0 and Agile, many project managers are still seeking to discover the best way to integrate editing into the process. One especially thorny issue: the "just good enough" approach used in Agile seems at odds with values that editors have traditionally espoused, like thoroughness and attention to detail.

Legacy Documentation

Documentation from a predecessor product, or from a previous release of the product in question, is easy to overlook in sprint-based documentation reviews. Yet this material ends up in the final published product, and in fact to a customer it's indistinguishable from the new material that's being developed.

Because scrum members have limited time and probably limited expertise, it's not reasonable to ask them to review legacy documentation. The best approach might be to enlist the aid of an experienced subject-matter expert. True, it's hard to find one of those, and harder to find one who has the time. But the return on investment, in terms of errors avoided, can be huge.

The SME might be more amenable to conducting such a review if:

The SME knows that he or she is the only person doing the review. The SME has more invested in the project, and he or she also knows that other reviewers won't question or try to "fine tune" the things that they recommend.

The project manager clearly delineates exactly what material the SME should review, and what material can be left out of the review.

Documentation reviews can be done at the SME's convenience. Usually there's flexibility as to when these kinds of reviews can be scheduled.

Localization

For most documentation projects in which I've been involved, the translation took place in one big gulp at the end of the project. It was always a challenge to "freeze" the documentation (and the user interface) while the software product was still undergoing testing.

The iterative nature of Agile forces us to alter this approach. It's simply impossible to fit a big, one-time translation step into a project schedule that's built around sprints.

Hamilton recommends breaking the translation job into small pieces in every situation—whether you're following an Agile process or a traditional process. In his words: "You never have enough time to simply shoehorn translation into the schedule as a single milestone between completing a document and delivering it" (note 10).

Once again, modular or topic-based writing fits extremely well with this model of breaking up the translation into smaller stages.

WHERE DO WE GO FROM HERE?

Collaborative or community-based authoring, in which a wide range of stakeholders participate, is fast becoming the norm for large segments of our audience. Enterprises are moving to accommodate this trend by modifying their documentation development processes.

Similarly, Agile is fast becoming an accepted way of developing software. Many of its underlying concepts—short development cycles, frequent updates, and quick responsiveness to customers—are becoming more widespread in other industries as well.

As project managers, we're learning new best practices to accommodate those trends. We're learning to break our documentation, and the processes we use to develop it, into smaller pieces so that we have flexibility in performing the steps within the process and in packaging the documentation for our customers. We're learning new ways to manage our content and the ways in which it's generated.

We're still learning every day. I hope that this presentation will help stimulate a conversation among documentation project managers so that we can learn together.

REFERENCES

1) Gentle, Anne: "Embrace the 'un'." Just Write Click blog, 17 Jan 2009. http://justwriteclick.com/2009/01/17/embrace-the-un/
2) Halvorson, Kristina: *Content Strategy for the Web*. Berkeley, CA, New Riders (an imprint of Peachpit), 2010. Page 83
3) Halvorson, page 16
4) Bailie, *Rahel: Rahel Bailie Provides A Content Strategy Primer*, September 2009. http://thecontentwrangler.com/2009/09/13/rahel-bailie-provides-a-content-strategy-primer/
5) Agile Manifesto. http://agilemanifesto.org/
6) Hamilton, Richard: *Managing Writers: A Real World Guide to Managing Technical Documentation*. Fort Collins, CO, XML Press, 2009. Page 98
7) Gentle, Anne: "Last sprint, first step." Just Write Click blog, 18 Jan 2010. http://justwriteclick.com/2010/01/18/last-sprint-first-step/
8) Halvorson, pages 46-108
9) Gentle, Anne: *Conversation and Community: The Social Web for Documentation*. Fort Collins, CO, XML Press, 2009. Pages 104-105.
10) Hamilton, page 141

Other sources consulted:

Harvey, Peggy: *Enabling User Interactivity with Documentation* (research paper published on acrobat.com)
Sheffield, Richard: *The Web Content Strategist's Bible: The Complete Guide To a New and Lucrative Career for Writers of All Kinds*. CreateSpace, 2009.

Larry Kunz
Project Manager
Systems Documentation, Inc.
1005 Slater Road, Suite 220
Durham, NC 27703
+1.919.354.1104

Larry Kunz is a project manager and information architect with Systems Documentation, Inc. (SDI) Global Solutions. In more than 30 years as a writer, manager, and planner he has experienced the transition from book-based documentation to today's integrated delivery of information from a wide range of sources using different formats and media.

Larry has managed and provided content for technical writing projects and marketing projects. He holds a Masters Certificate in Project Management from the George Washington University and teaches the course *Managing the Information Development Process* in the Technical Communication certification program at Duke University. He is a Fellow in the Society for Technical Communication (STC) and currently heads the Society's strategic planning effort.