4-21-2016

Author Support for Wiki Based Help Systems: Genre Conventions for Content Templates

Claire Lundeby
Minnesota State University Mankato

Author Support for Wiki Based Help Systems:
Genre Conventions for Content Templates

Claire Lundeby

English 696: Capstone Course in Technical Communication
Dr. Nancy MacKenzie
21 April 2016

Abstract

Author support systems are critical in wiki help systems. I provide a critical review of existing research on author support systems, content strategy, content management, and how these intersect, as well as a structural content analysis of five content templates in an effort to define genre conventions.

Introduction

Technical communication, as a profession, is continually shifting and redefining itself and the skillset of its practitioners. Technical communicators have had to hone their skills in areas such as content strategy and content management. Similarly, the genre of help documentation is changing, pushing the boundaries of media. As manuals and other print documents begin to fade, other media like wiki-based help are gaining momentum. Practices of content management and content strategy intersect at author support systems for wiki contributors.

Wiki-based social help systems are wiki sites where technical communicators, software developers, end-users, and others can collaborate to create a community of task-based, applicable help for a product. Wikis use content generated by a variety of authors, and therefore technical communicators may have somewhat less control over each content asset. Thusly, it is important that technical communicators support a variety of author personas by implementing author support systems. These author support systems are a toolkit to quickly train an author to write effectively and efficiently for the wiki. A toolkit can contain content definitions, content templates, definitions of workflows, editing guidelines, style guidelines, formatting guidelines, and many others.

I became interested in content templates, especially for wikis, when my former workplace implemented a WordPress wiki. When I was first given drafts of the new content templates to edit, I was aghast at the informality of the text and the detailed information on basic grammatical rules. Then, I began to realize that this type of writing may be as unfamiliar to a software engineer as writing in C++ may be to a non-programming technical writer. This unfamiliarity is what makes these author support systems so critical.

My paper focuses on an examination of the necessity for these systems and defining the conventions of the genre. My research fills a gap in the scholarly conversation by working to define genre conventions and looking at the functionality of such systems. I do so through a critical review of existing literature and through a structural content analysis of five content templates for open-source product wikis. While this paper covers author support systems as a whole, I focus on content templates as a sub-genre of author support.

Terminology

A particular aspect of this research that I have struggled with is how to articulate and title the various roles to which I am referring. While it would be easy, and at least partially accurate, to claim that these support systems are addressing under-skilled or untrained writers, this language may be termed insulting or offensive, and it does not articulate the wide breadth of individuals using these support materials and contributing to these social help systems.

I have chosen content writer to describe the untrained or non-designated writer, or the person who is utilizing the author support materials, for example, content templates, and is authoring content to the social help system. I use support author to describe the technical communicator who is creating these support materials. Author support refers, in broad strokes, to any material published to support the content writer, or any other involved person, including, but not limited to, content templates, content definitions, and editing guidelines. These materials are designed specifically for these content writers, covering anything from basic writing skills like grammar and syntax to more advanced topics. Table 1: Definitions reviews these terms concisely and can act as a reference for the remainder of this paper.

Table 1: Definitions

Term	Definition
Content writer	Non-designated or untrained individual who is authoring content or documentation for the help system
Support writer (or technical communicator)	Refers to the technical communicator who is authoring support
Author support	Materials published by the support writer to support the content author, essentially the documentation for the documentation
Content template	This is not a form based template, rather a text based one that is expressing what kinds of content should be placed where in a document and how they should be written.
End reader	This person is the one who is reading the final wiki entry. That is, the wiki entry that was written based on the content template or other author support.

Literature Review

The existing research about author support systems and content templates either establishes their existence or explains why they are necessary. Research into author support systems, both their efficacy and development, revolves around defining the genre. I argue that author support systems exist as the intersection of content strategy and content management, and that content templates are both a front end and a back end concern, and thus should be informal and lightly-constrained.

This literature review will examine existing literature about the changing role of the technical communicator, author support systems, content strategy, content management, the Author Support for Wiki Based Help Systems

intersection between those three areas, and content templates, as part of author support and as a genre. Author support materials are vital to the content production process. Authors like Luke Thominet, Erin Kissane, John Gallagher, and Angelo Di Iorio have begun this conversation, examining types of author support materials, particularly templates, and their structure, audience, and function. Present research indicates that content templates are an effective method of author support. When content templates are efficiently and effectively designed, they empower authors to author content quickly and well.

Technical Communicators as Content Strategists and Content Managers

The role of a technical communicator is constantly changing so that the technical communicator can continue to make a case for themselves, their usefulness, and their return on investment. This is why there is not typically a clear consensus on what precisely technical communicators do, or a consistent job description across technical communication roles. Technical communicators accomplish this goal of making a case for themselves by increasing their skills in usability, user experience, and content strategy. Instead of creating product documentation that is not communicating well to start with, technical writers can work more closely with developers and subject matter experts to create and manage content.

Laroche and Traynor indicate this need in their 2013 article, “Technical Communication on Life Support: Content Strategy and UX are the Reclamation.” As the title of their article indicates, Laroche and Traynor are claiming that the days of the technical communicator as writer are beginning to fade. Now, technical communicators often act as user experience design professionals and content strategists, rather than technical writers. Laroche and Traynor suggest that a technical communicator must be invested in producing a product and interface

that is intuitive and requires less documentation. By creating a quality product that communicates well to begin with, the technical communicator no longer needs to supplement that product with extensive documentation (Laroche and Traynor 2013).

James Dubinsky corroborates this shift in the nature of technical communication in his article, “Technical Communication Products and Processes.” Dubinsky conducted a modified Delphi study to uncover the changing role of the technical communicator, and how those technical communicators in managerial positions felt about these changes. Reflecting on the ways that technical communicators need to insert themselves into the broader processes at their workplace, especially ones involving development and design, to create better content, Dubinsky commented, “when management sees technical communicators as integrated members of teams with a diversity of skills (as ‘versatilists’) …then technical communicators have a much more viable future.” (Dubinsky 2015, 127) The role of the versatilist is essentially what Laroche and Traynor describe. By assuming the role of the versatilist, technical communicators can increase their return on investment and secure their future (Dubinsky 2015).

Pullman and Gu also discuss the changing role of the technical communicator in their introduction, “Rationalizing and Rhetoricizing Content Management.” Content management is the intersection of theory and practice, and thusly, content management is right where technical communicators need to be. Pullman and Gu suggest that technical communicators have been acting in content management type roles for years, e.g. single sourcing and knowledge management, in addition to their other work. These practices can be cemented into specific content management regimens by the technical communicator to produce and

maintain more effective, better organized content. The management of information assets as part of a technical communication role is key to the investments of a technical communicator, and so content management is an integral responsibility and position (Pullman and Gu 2007).

Shaun Slattery conducted research about the distribution and management of labor for technical writers, in what he describes as “complex information environments” (Slattery 2007, 311). Slattery observed and interviewed technical writing employees at one technical writing vendor business, chosen because their communication with their clients is all mediated by other employees. There is no interaction between client and technical writer. Slattery found that writers interact with and switch among many documents, of various lengths and types, many times during a task, switching back and forth regularly (Slattery 2007).

Because they are switching continually between many documents, Slattery calls their work undistributed. Eventually, to undistribute work, writers begin to reuse text that is authorized by subject matter experts. This reuse needs to be strategic, which is an opportunity to develop and use rhetorical skills. Slattery is concluding that distributed work can lead to writers having a smaller knowledge base, rather than technical communicators becoming so familiar with the product they document that they are nearly subject matter experts. This undistribution of work by reusing materials in this organization was a precursor to single-sourcing, which can be a large part of content management (Slattery 2007).

Technical communicators often act as moderators of wikis, working with large scale organization projects and determining what documentation projects still need to be completed and published that to the wiki, and creating samples of documentation types (which can exist as a content template using a clone function or be easily transformed into a content template).

Erik Berglund and Michael Priestly advocate for widespread user contribution to open source documentation, especially as a precursor to changes made to the source code, because flaws in documentation are less damaging than errors in in source code (Berglund and Priestly 2011).

Because of this constant shifting in the nature of the profession, content strategy, content management, and, by extension, the development of author support systems is of critical interest to technical communicators. Skills in these areas provide an increased opportunity to articulate the value of a technical communicator to a team or company and an increased return on investment of the technical communicator.

Defining the Genre

Luke Thominet, in his article, "Building Foundations for the Crowd: Minimalist Author Support Guide for Crowdsourced Documentation Wikis," described the author support guides he analyzed for five different crowdsourced documentation wikis. Thominet claims that "these texts have represented the best means for professional technical communicators to affect the consistency of the genre" (Thominet 2015, under "3. Author support"). Thominet conducted a structural content analysis on five different documentation wikis for open source products.

- The Blender Wiki.
- The Fedora Documentation Project Wiki.
- The Apache OpenOffice Documentation Project Wiki.
- The Tiny OS Documentation Wiki.
- The Ubuntu Community Help Wiki.

Thominet concluded that these author support guides supported several conventions of the genre. Essentially, Thominet examined the parts of the author support included in these wikis and what makes them functional.

Thominet denoted three categories of author support content elements: obligatory, conventional, and optional. Then he explored each of the author support guides listed above to ascertain if the guide followed both categories of these conventions. The results of Thominet's study can be found in Table 2 below, which is a modified version of Thominet's own results table (Thominet 2015).

Table 2: Results of Content Analysis (Thominet 2015)

Content Element	Obligatory	Conventional	Optional	Blender	Fedora	OpenOffice	TinyOS	Ubuntu
Categories		X		X		X	X	X
Contact	X			X	X	X	X	X
Graphics	X			X	X	X	X	X
Links		X		X		X	X	X
Mentoring			X		X			
Welcome	X			X	X	X	X	X
New pages			X			X	X	X
Wiki structure			X	X		X		
Templates		X		X		X		X
Titles/ headings				X		X	X	X
Wanted pages			X		X	X		
Workflows			X	X	X			

Content Element	Obligatory	Conventional	Optional	Blender	Fedora	OpenOffice	TinyOS	Ubuntu
Writing style	X			X	X	X		X

In Thominet's study, the term obligatory content is describing those author support pieces that are essential to an effective author support system and the subsequent ability of content writers to author quality, useful content. Discussing the use of graphics and the appropriate writing style for the particular brand of help documentation is integral. Conventional content is that content that may not be 100% essential for a content writer, but is nevertheless very helpful to that content writer and the presence of which helps generate better content. Optional content is referring to ideas, functions, and materials that were very interesting and would certainly assist content writers, especially inexperienced ones, but were not deemed necessary to the success of the content writer. I argue that content templates exist as conventional content, and so agree with Thominet (Thominet 2015).

Author Support is Intersectional

Content management systems are essentially, "transforming text into data" (Pullman and Gu 2007, 3). Content management is just that, the management of all content assets within an organization. Content assets refers to any individual piece of content, whether it is visual or textual, on the web or in print. For this methodology to become effective, it is important that technical communicators begin to consider all of their content as a whole, rather than individual pieces of documentation or small groups of publications. We must, "start thinking in terms of asset management…writing with multiple authors and multiple purposes feeding off of

and contributing to a conglomeration of assets that collectively make up a content archive" (Pullman and Gu 2007, 3). Content management can be accomplished either through a team of individuals or through complex software systems called Conent Management Systems, or CMS. The result of content management is that there is a large repository of effectively written content that is available as a single source for a variety of presentation types, which can be easily accessed and reused. However, it is vital that this reuse of single-source content be strategic (Pullman and Gu 2007).

Kristina Halvorson is one of the primary names in content strategy instruction and scholarship. She has published two editions of her popular text, *Content Strategy for the Web*. She describes content strategy as a large scale project that focuses on goals and strategic planning. Content strategy empowers a great understanding of content and ways to purpose it. Content strategy is a typically high level method of goal setting and decision making about an organization's content needs. Content management cannot function effectively without developing a specific plan for its implementation and maintenance, as well as defined goals. Content strategy accomplishes this by aligning mission and content. Content can often be spread too wide over form and tone, or it may not transition well from legacy content. Content strategy enables a company to ensure that their content is achieving their business goals (Halvorson and Rach 2012)

Craig Baehr, however, conceptualizes content strategy as the culmination of knowledge management, content modeling, and user experience (2013). Baehr defines content modeling as, "defining (and maintaining) the structure and granularity for the content assets contained within an information product" (Baehr 2013, 395). Content strategy is a practice that is

naturally developed during a stage of strategic planning, which may occur before the creation of any content or following a content audit. Baehr was responsible for creating the content strategy for the Technical Communication Body of Knowledge (TCBOK). For this project, Baehr focused his efforts on understanding patterns of issues and opportunities in the existing TCBOK, understanding user personas relating to information taxonomy, and "synthesizing findings in a discussion based format" (Baehr 2013, 298). While Baehr focused on creating a content strategy for a body of knowledge, these goals can be used, if moderately adapted, to develop a sustainable content strategy for any larger collection of content assets, like a wiki (Baehr 2013).

Author support systems represent the intersection of content strategy and content management. As Thominet stated, "these [author support guides] texts have represented the best means for professional technical communicators to affect consistency" (Thominet 2015, under "3. Author support"). Author support systems enable the technical communicator to ensure that a specific type of content is created in a way that complies with any planning they have done. For example, when content templates include things like headings with short content descriptions for the section beneath the headings, the support author has increased the likelihood that the content writer will appropriately organize their content in a way that is effective and understandable to the end reader.

Pullman and Gu emphasize Bill Hart-Davidson's point in "Coming to Content Management," that "organizations should view content management 'as a change in the technological and social infrastructure that makes their organization work" (quoted in Pullman and Gu 2007, 5). Content management, however, must be informed by content strategy, and vice versa. For example, both of these practices have determining structure in common, either

through structuring for purposes of single sourcing or metadata, or to display it most effectively to show an organization's mission. Structure would very easily be informed by author support materials. A content definition would suggest it and a content template, depending on the type, would enforce it.

Content Templates

The idea of using content templates as a means of author support is a rhetorically grounded one, as John Gallagher explains in his article, "The Rhetorical Template" (2015). Gallagher claims that templates do not actually completely separate form and content, as some may suggest, but rather that these templates can be used in a way that is effective. Gallagher asserts that effective content can still be generated using the template, in a happy marriage of form and content. It is important to ascertain the rhetorical situation of content writers using support writer generated content templates. The question is whether the templates themselves are the rhetorical situation, and then writers are participating in that situation, or if the templates are merely a piece of a larger situation, with the writers being another piece, acting together to form the situation in which the content is participating. Gallagher posits that in this instance, the situation and the roles are in no way inflexible, so neither are the writer and the template. One should "make the template flexible" (Gallagher 2015, 3). This template flexibility coincides with the lightly-constrained template that Di Iorio describes (Gallagher 2015).

Angelo Di Iorio, Fabio Vitali, and Stefano Zacchiroli analyzed two types of wiki content templates, functional and creational templates, using an engine-agnostic method, and used that analysis to produce their own template recommendation, lightly-constrained templates.

They also provide an implementation plan for their lightly-constrained templates model. Di Iorio, et. al. claim that providing content templates will enable authors to develop more in-depth wiki content more easily. Templates should be structured, readable, and safe (meaning that users can revert changes and know the impact of those changes). Di Iorio, et. al. outlines how lightly-constrained templates achieve these goals (Di Iorio, et. al. 2008).

Di Iorio et. al.'s templates are not precisely the templates involved in these author support systems when the author support system exists as a guide. Di Iorio et. al. describe templates that are an actual part of the content publishing platform. Di Iorio, et. al. describe the first of their template types – the *functional* template. This style of template is recognizable from sites like Wikipedia and MediaWiki powered wikis. The primary characteristic of a functional template is that there is a strong connection between the template and the instance. This means that the template is the end all be all, and that all instances to which the template is applied will resemble each other. This also means that any time the template is altered, then all instances will also be altered. This is positive for situations where form is essential. However, with this type of template, there are some significant difficulties with editing, both in template and instance, the most notable being interacting with placeholders. These templates are described as "straightforward," but that they also "require additional syntactic and conceptual knowledge, and produce as a result pages potentially troublesome for the average editor, due to markup non-linearity" (Di Iorio, et. al. 2008, 619). These functional templates are very common, as it resembles a form where there are fields to be filled in (Di Iorio, et. al. 2008).

The second type of existing template that Di Iorio et. al. describes is the *creational* template which is unlike the functional template in that it is a seeding page, rather than an

independent entity, at the start of the writing process. At their most basic, creational templates are the clone function in a wiki publishing platform. The source of the template or an article can be cloned and then edited more simply from the existing instance to produce a new article. When using a creational template, there is much less of a connection between template and instance than there is in a functional template (Di Iorio, et. al. 2008).

The final type of template, and the one that Di Iorio et. al. recommend, is the *lightly-constrained* template. This type combines features from both functional and creational templating. The lightly-constrained template is most similar to a creational template, but it acts almost as a failsafe to ensure that each cloned instance resembles the original template. Each page that is generated from a lightly-constrained template should remain entirely inline with the template, especially the organization. The lightly-constrained template's encoding acts as a validator, either as a confirmation message or an error message. The validator functionality empowers the content writer to correct their own errors, rather than relying on a support writer or any other technical communicator, writer or editor, to repair their content. This prevents disruptions to the workflow and enables better written content (Di Iorio, et. al. 2008).

Regardless of the type, templates serve an important function, in that they empower the content writer to generate better organized and more effective content. Content templates have an important secondary function: validation. Angelo Di Iorio, Francesco Draichio, Fabio Vitali, and Stefano Zacchiroli investigate this validation function in detail in, "Constrained wiki: The WikiWay to validating content" (2012). Wikis are so popular because of their open editing philosophy. However, that kind of openness can sometimes lead to lower quality content that may be riddled with errors or difficult to read. "The WikiWay is the open editing philosophy of

wikis …In spite of all of this freedom…[the] editing process is often bound by implicit rules…to match specific page templates" (Di Iorio, et. al. 2012, 1). Templates can easily be structured on the back end to act as validators for form, content, or both. That validation can consist of either detailed error messages or simply removing the save function until certain template parameters can be met (Di Iorio, et. al. 2012).

In Erin Kissane's 2009 article, "Content Templates to the Rescue," she describes methods for the development and implementation of content templates, especially in larger organizations. Kissane claims, "The bigger the organization, the harder it usually is to get content to flow smoothly from far-flung experts to web-savvy writers and editors to the website itself" (Kissane 2009, 2). Kissane begins by making a case for the existence of solidified content workflows early on. An organization should conduct a content audit and then develop author support systems to support the types of content that need to be created. Kissane uses templates which act almost as wizards, and typically include a page title, headings and a summary of what content should be beneath each heading, as well as the format, and then examples of each. While her language is certainly informal, that informality is somewhat the appeal of this brand of content template. It is usable and can enforce style and branding guidelines, while informal and friendly enough to not be pedantic or otherwise insulting to content writers (Kissane 2009).

Structure of Wiki Templates

Anja Haake, Stephan Lukosch, and Till Schummer claim that there are essentially two types of wiki authoring pages in templates, the "traditional publishing model" and an "enhanced page

model," (Haake, et. al. 2005) and provide equation models for them, which can be found in Table 3: Equation modeling.

Table 3: Equation modeling (Haake et. al. 2005, 47)

Model	Equation
Traditional Publishing Model	Page = Title x SourceText
Enhanced Page Model	Title x {I D -> SourceText}

The traditional publishing model complies with what is conventional for the author visual sandbox environment, where there are only two fields: title and wiki text. While this traditional publishing model is perhaps best suited to pages where a template will not be applied, they can be fitted so that a template can be applied to the SourceText area. In the enhanced page model, however, it is easier for the content writer to add any number of attributes to the page in line. With either of these models, the content writer can move away from template conventions, unlike the lightly-constrained templates that Di Iorio et. al. suggest, which act more strictly as validators, and do not allow the user that movement. This applies to form based templates as well. The traditional publishing model would contain only the fields within the existing template. In the enhanced page model, the content writer has the opportunity to add additional fields as needed to support the content they are creating (Haake, et. al. 2005).

Methodology

I have conducted a blended content analysis of five wiki content templates:

- Ubuntu Community Help Wiki's "Documentation Template"
- CentOS Wiki's "Template for Help Pages"
- XILINX Wiki's "Tech Tips Template"
- Darktable Wiki's "Plug-in Description Template"
- Python Wiki's "Web Programming Template"

My content analysis is a blended one in that it contains elements of conceptual content analysis, but it is primarily structure driven. This is very similar to the design of Luke Thominet's content analysis in his article, "Building Foundations for the crowd: Minimalist Author Support Guides for Crowdsourced Documentation Wikis" (2015). Whereas Thominet was establishing *which types* of author support are present, I am determining which elements are present in content templates specifically. One of the main gaps in scholarly research is defining genre conventions for these forms of author support. I have decided to focus on content templates, both out of their general importance and personal interest. I have decided to focus my study on wikis for open source software (Thominet 2015).

Data Collection

For this analysis, I have chosen the following elements of content templates to check for:

- Word count
- Number of headings
- Levels of headings
- Use of tables
- Use of lists
- Placeholder text type
- Version list
- Table of contents
- Breadcrumb navigation
- Number of mentions of the product's name
- Links to other author support

Word count indicates the length of the template and whether it can be considered somewhat minimalist, in line with Thominet's recommendation. I obtained this data by copying all textual content of the content template into a Microsoft Word document and using the Word Count

tool. The number of headings and levels of headings were counted twice. These can indicate both the level of organization and of detail. The use of tables and lists impacts both the clarity of the template and the clarity of the generated content. Placeholder text can be lipsum, parenthetical fields, other, or none.

Some content templates are designed to only be applicable for a certain version of a software. For example, one template may heavily emphasize documenting features that have been deprecated in other versions. Table of contents and breadcrumb navigation can act as an important organizational and navigational tool. Tracking the number of mentions of the product's name can be indicative of whether or not the content template has been designed entirely as author support for this product's wiki or if it is external and part of a larger set, like MediaWiki. Links to other author support materials somewhere on the page, whether in the template or in navigation spaces, are important and can show the level of integration between different author support materials. Lastly, I have included what wiki service, or other, is powering the wiki.

Results

The results of the analysis can be found in Table 3: Results.

Table 3: Results

	Ubuntu	CentOS	XILINX	Darktable	Python
Word Count	184	9	532	985	31
# of Headings	2	2	14	12	15

	Ubuntu	CentOS	XILINX	Darktable	Python
# of Heading Levels	1	1	3	2	2
Tables	No	No	Yes	No	No
Lists	Yes	No	Yes	Yes	No
Placeholder text	None	Other	None	Other	Other
Version list	No	No	No	Yes	Yes
Table of contents	No	No	Yes	Yes	No
Breadcrumb navigation	No	No	No	Yes	Yes
# of product mentions	2	0	2	5	1
Links to other author support	No	No	Yes	No	No
Links to External documentation	Yes	No	Yes	No	No
Powered by	Ubuntu	CentOS	Wikispaces	Redmine	MoinMoin

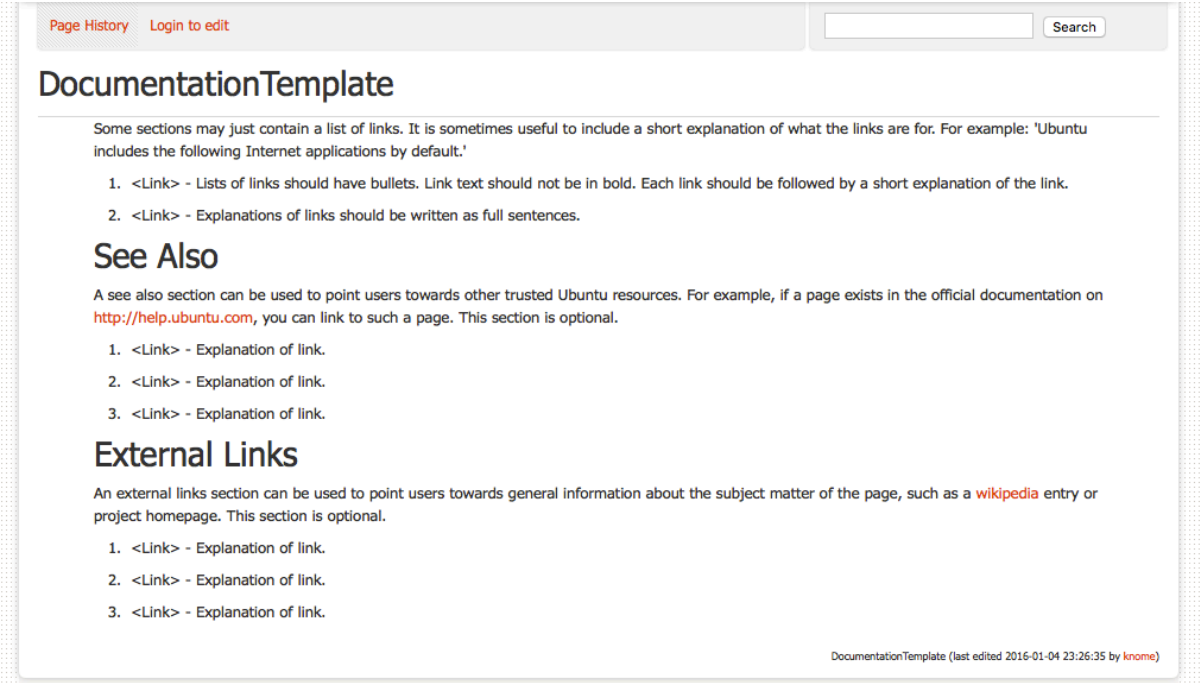
Discussion

Ubuntu's "Documentation Template"

Ubuntu's "Documentation Template" (Figure 1) is most representative of the middle ground of content templates. While it is not as bare as the CentOS template, it is not quite as detailed as the XILINX template. It actually somewhat resembles the form of a Wikipedia entry, but the site

is not powered by MediaWiki (the service that powers sites like Wikipedia). This type of wiki entry is primarily supported by a series of link lists. The link lists are what makes Ubuntu's template similar to CentOS. However, Ubuntu's template includes explanations of each link. This provides more information and a preview of sorts for the end reader of the wiki entry. If the content writer follows these guidelines to include link explanations, the end reader will have an ultimately more fulfilling and informational experience with the wiki entry (Ubuntu).

Figure 1 Ubuntu template

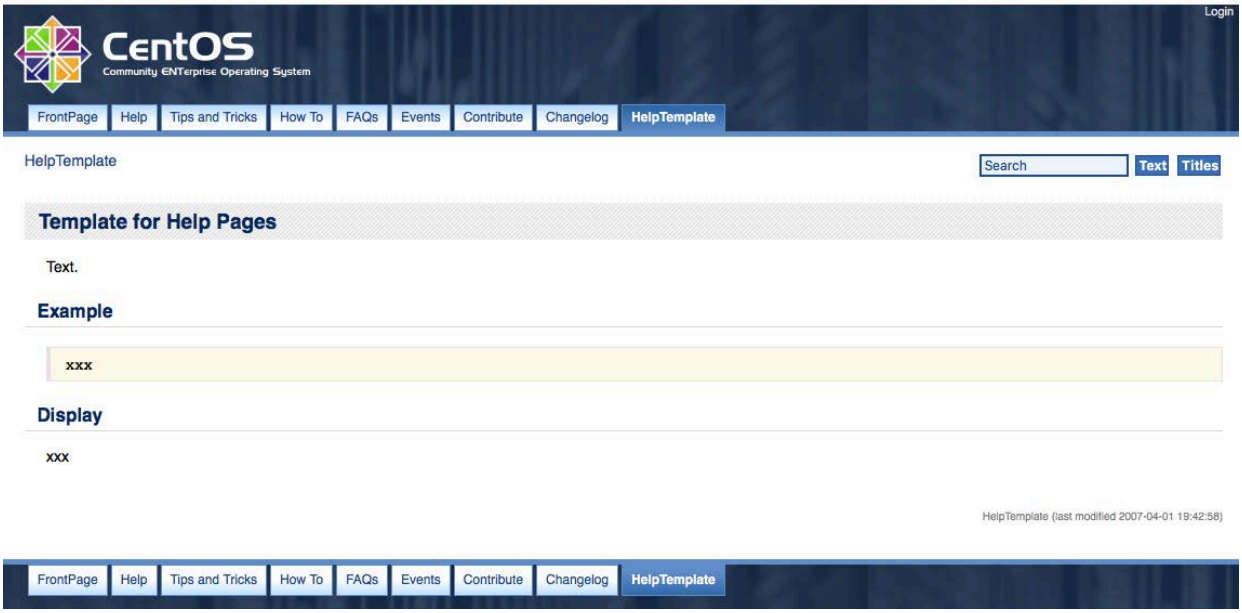


CentOS's "Template for Help Pages"

The CentOS template, pictured below in Figure 2, with only nine words, is perhaps the most minimal version of a content template. It has no advisory text, instead using only placeholders that are representative of neither length or content. What is most confusing with this particular template is that it uses the same placeholder text, "xxx," to represent code, in the yellow box below the "Example" heading, and to represent the rendering below the "Display" heading.

Logically, these should be represented differently. However, this confusion is offset by the <pre> HTML tag applied to the code placeholder, the style for which provides the somewhat conventional container for the code sample (CentOS).

Figure 2 CentOS template

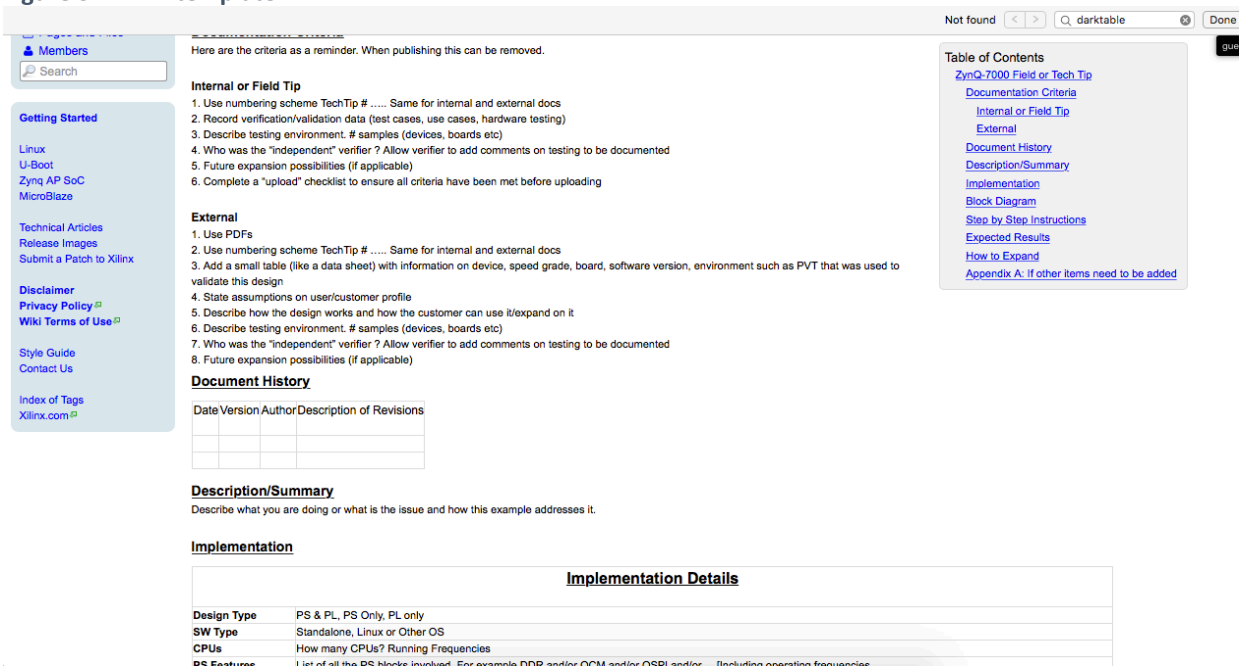


XILINX's "Tech Tips Template"

The XILINX "Tech Tips Template" (Figure 3) is perhaps the most thorough of these five templates, and it is likely more representative of non-open-source product wikis. This particular template, and the other templates provided by XILINX on their wiki, opens with two lists that include general, across the board content recommendations, and then proceeding to the rest of the template. This particular template utilizes several tables, which have content explanations in each cell of the table. These are excellent in that they provide a visually usable template for the content writer and a scannable wiki entry for the end reader. The content descriptions appearing below each heading in this template are distinctive in that they are concise, imperative sentences, rather than descriptive sentences. Imperative sentences can increase the clarity of the content description for the content writer. These imperative sentences are also

somewhat reminiscent of other trendy documentation styles, as these commands are focusing on how to accomplish a task (writing this section of the wiki entry) rather than how to utilize the feature (how to use the template) (CentOS).

Figure 3 XILINX template



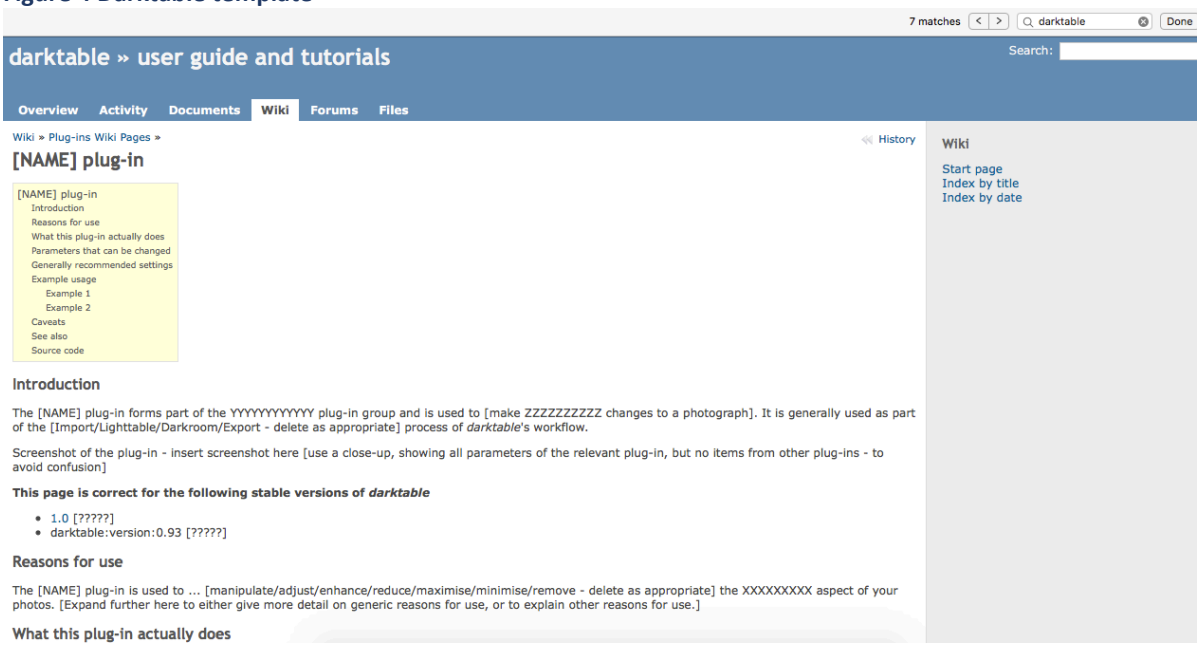
Darktable's "Plug in Description Template"

The Darktable "Plug in Description Template" (Figure 4) is the least readable of the five templates, and as an extension of that, it may be the least usable. This template uses repeated alpha characters in no discernable pattern, e.g. xxxxxx or ZZZZZZZZZZZZ (Darktable), as a placeholder. Because these placeholders are in line with the descriptive text, it is both difficult to read and less usable. However, the headings in this template follow the most logical function for their purpose (plug in description): "Reasons for Use," "What this plugin actually does," "Parameters that can actually be changed," (Darktable) and so on. This easy to follow template sequence compensates for the awkward placeholders. This template also provides bracketed verb choices, e.g. "[manipulates, adjusts/enhances/reduces/maximizes/minimizes/removes]."

Author Support for Wiki Based Help Systems

This shows the great potential of wiki content templates to regulate elements of content like word choice (Darktable).

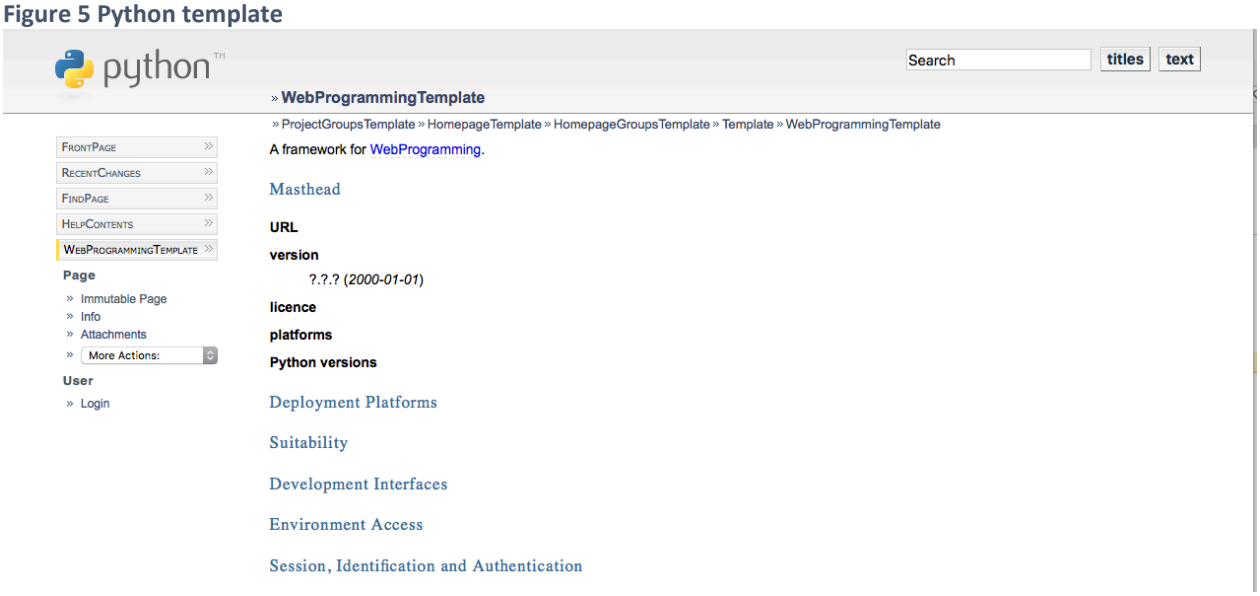
Figure 4 Darktable template



Python's "Web Programming Template"

Unlike the previous four templates, Python's "Web Programming Template" (Figure 5) is almost entirely heading driven. The only guidance outside of headings is the format to use when including the version number the wiki entry will be referencing. While this may come across as too bare, it may actually increase the usability of this template for the content writer. With this minimalistic content template, the content writer can exercise a great deal of license in what content they include in their wiki entry. However, it can also lead to ambiguity for the content writer and a lack of consistency in templated wiki entries. This template may also rival Darktable's template for compliance with Erin Kissane's recommendations for content templates (Python).

Figure 5 Python template



There are certainly some similarities between these templates. All five of them utilize headings. This usage is very productive because it provides visual organization cues as well as guidance about arranging sub-topics within a wiki entry. However, these five content templates are each very different, which is representative of content templates for open source software wikis as a whole. While a wider study with more participating wikis would statistically expose trends that I could utilize to develop genre conventions, the disparities between these five expose how widely varied the genre itself is, so perhaps it would not be productive to fully *establish* genre conventions. It may instead be more valuable to attempt to uncover best practices so that content writers and end readers alike can have a higher quality experience with wiki based help systems.

Conclusion

Technical communicators experience constant change in the parameters of their profession. Technical communicators will more often find themselves in roles working with wiki based

social help systems. In complying with an open source documentation model, technical communicators act more so as a guides than they do as content writers in this instance. So, they become support authors—creating author support materials to support a variety of content writer personas. I have examined the necessity for these systems, author support as intersectional, and content templates. This has begun to fill a gap in the scholarly conversation—the lack of easily defined genre conventions for wiki content templates. However, this research has started to reveal that creating a single set of genre conventions may not be viable. Instead, content templates may need to broken down in to sub-genres prior to forming a set of genre conventions or best practices because their varied types are both a front end and a back end consideration.

Further Research

There are vast opportunities left here for further research. While I had hoped to uncover consistent genre conventions across content templates, I instead found that with only these five samples there are vastly different types of templates. I would recommend conducting a similar study with a much larger sample size. Following that, there is potential to conduct a study with participants to determine the efficacy of the content template and its reception by the content writer.

Bibliography

Baehr, Craig. 2013. “Developing a sustainable content strategy for a technical communication body of knowledge.” Technical Communication. 60 (4): 293– 306.

Berglund, Erik and Michael Priestly. 2001. “Open-source documentation: In search of user driven, just-in-time writing.” In SIGDOC ’01 Proceedings of the 19th Annual International Conference on Computer Documentation. ACM. 132– 141.

Di Iorio, Angelo, Francesco Draichio, Fabio Vitali, and Stefano Zacchiroli. 2012. “Constrained wiki: The WikiWay to validating content.” *Advances in Human-Computer Interaction*. 2012: 1– 19.

Di Iorio, Angelo, Fabio Vitali, and Stefano Zacchiroli. 2008. “Wiki content templating.” In WWW ’08 Proceedings of the 17th International Conference on World Wide Web. ACM. 615–624.

Ubuntu. "Documentation Template." Ubuntu Help Community. Accessed March 2016. https://help.ubuntu.com/community/DocumentationTemplate.

Dubinsky, James M. 2015. “Products and processes: Transition from ‘product documentation to integrated technical content’.” Technical Communication. 62 (2): 118– 134.

Gallagher, John R. 2015. “The rhetorical template.” Computers and Composition. 35: 1–11.

Haake, Anja, Stephan Lukosch, and Till Schummer. 2005. “Wiki-Templates: Adding structure support to wikis on demand.” In Proceedings of WikiSym ’05. ACM: 41– 51.

Halvorson, Kristina, and Melissa Rach. 2012. Content Strategy for the Web. 2nd ed. Berkeley: New Riders.

CentOS. "Help Template." CentOS Wiki. Accessed March 2016. https://wiki.centos.org/HelpTemplate.

Kissane, Erin. “Content templates to the rescue.” A List Apart. http://alistapart.com/article/content-templates-to-the-rescue (Accessed September 9, 2015).

Laroche, Chris, and Brian Traynor. 2013. "Technical Communication on Life Support: Content Strategy and UX are the Reclamation." In IEEE International Professional Communication 2013 Conference. 1–6.

Darktable. "Plug-in Description Template." Darktable » User Guide and Tutorials. Accessed March 2016. http://redmine.darktable.org/projects/users/wiki/Plug-in_description_template.

Pullman, George, and Baotong Gu. 2007. “Guest Editors’ Introduction: Rationalizing and Rhetoricizing Content Management.” Technical Communication Quarterly. 17 (1): 1–9.

Slattery, Shaun. 2007. “Undistributing work through writing: how technical writers manage texts in complex information environments.” Technical Communication Quarterly, 16 (3): 311–325.

XILINX. "Tech Tips Template." Xilinx Wiki. Accessed March 2016. http://www.wiki.xilinx.com/Tech Tips Template.

Thominet, Luke. 2013. "Building foundations for the crowd: minimalist author support guides for crowdsourced documentation wikis." In Proceedings of the 33rd Annual International Conference on the Design of Communication, ACM: 43.

Python. "Web Programming Template." Python. Accessed March 2016. https://wiki.python.org/moin/WebProgrammingTemplate.