Choosing a Documentation Tool

By John Green – for James Rosewell

Before I start, I should point out that I have discounted the following:

- **Word processors (e.g. Microsoft Word)**: Whilst you can write technical documentation in Word, it was not designed with technical writing in mind, and, in particular, is not suited to long documents. I would not recommend using it for anything other than short, one-off documents.
- **Documentation generators (e.g. Doxygen, Javadoc)**: These are specifically for documenting software source code. What they do is go through your code and compile a document based on code patterns and comments. This helps other programmers quickly make sense of your code.

Tool Categories

The tools that are currently available to Technical Writers typically fall into the following 4 categories, in increasing order of ease of use:

- **Static Site Generators (SSGs)**
- **SaaS/PaaS Hosted Solutions**
- **Help Authoring Tools (HATs)**
- **Wiki Platforms**

SSGs

Examples are: **Jekyll**, **Hyde**, **Sphinx**, **Asciidoctor** and **Hugo**.

This is the 'techie' category. Static site generators are often associated with the **Docs as Code movement**. In a typical scenario, Technical Writers, in collaboration with Developers, create content in a lightweight markup language such as **Markdown** or **reStructuredText** (for **Sphinx**) and store it in a source code repository such as **Git** or **Subversion**. The SSG turns this content into a set of static HTML files, which are then uploaded to a website.

This approach offers considerable flexibility: you can choose your own text editor, your own markup language (provided that your SSG supports it), your own source control environment, and your own hosting solution. It is also especially suited to software documentation: the content format is familiar to programmers, which encourages contributions, and most SSGs offer software-specific features, such as support for code snippets and UML diagrams. Last but not least, many popular SSGs are open source.

However, SSGs are not for everyone. A documentation toolchain incorporating an SSG requires a system administrator with a high degree of technical knowledge to maintain it. In addition, you need to be comfortable with **command line interfaces**, CSS and checking code into, and out of, repositories.

More about static site generators:

- **Top Open-Source Static Site Generators**

SaaS/PaaS Hosted Solutions

Examples are: **ReadTheDocs, Siteleaf** (with **Jekyll**) and **VersionPress** (an offshoot of **WordPress** that fully syncs with Git).

SaaS (Software as a Service) is a licensing model under which you pay a periodic subscription fee in return for access to a piece of software/hosting, as well as support and service from the publisher.

When you apply the SaaS model to documentation software, what you get is a fully hosted, scalable solution for authoring, reviewing, and publishing. This is obviously an attractive option for organisations that lack the resources to set up and maintain an entire toolchain themselves. This type of software is also sometimes called PaaS, or Platform as a Service. The big advantage of these solutions would be that they can be configured so that, whenever you push code to your preferred version control system, including Git, your docs are built automatically, so that your code and documentation are never out of sync

The other side of the SaaS/PaaS coin is that it may be difficult (if not impossible) to migrate your content to another solution if you ever need to. Also, most of the newer SaaS tools seem aimed towards software developers and other non-Writers. They appear to trade sophistication for ease of use, lacking many of the essential features that a professional Technical Writer has come to expect. Still, a SaaS tool could be appropriate for a modestly-sized development team with minimal documentation needs.

HATs

These are listed **here**, with the prominent ones of which I have recent experience being MadCap Flare and, of course, **Paligo** (which, albeit a SaaS solution in terms of pricing - refer to **SaaS/hosted solutions** - is technically based on the HAT model).

These tools are aimed at professional Technical Authors and Writers, enabling them to manage virtually every aspect of producing the documentation themselves, from writing, to design, to publishing, to translation. This usually requires little knowledge of web design or coding, although XML knowledge is particularly useful as the GUIs do not allow you to do everything (each tool has an XML-type ‘back door’). A HAT is therefore especially useful for a Technical Writer working solo, or in a non-technical team. However, this means that content does tend to be isolated in the tool, which can make reviews and other cooperative tasks counter-intuitive.

Note: Paligo does offer collaboration via the Cloud so that the technical team can make comments, and even originate topics using Markdown (but licensing is not cheap!). However, it may prove difficult (if not impossible) to migrate your content to another solution if you ever need to.

More about HATs:

- **Choosing a HAT**
- **List of HATs (Wikipedia)**

Wiki Platforms

Examples of these are: **Confluence**, **MindTouch** and **MediaWiki**.

Wiki platforms are aimed at collaboration. They tend to focus on ease of use and removing as many barriers to contribution as possible. This makes them a quick way to get from nothing to something: it's a matter of a few clicks to create, review and publish a page of content. A wiki is great for teams without a dedicated writer who need an easy way to create content and are not overly worried about having a solid information architecture.

However, whilst wikis enable everyone to contribute, they do not always allow a Technical Writer to curate and structure content effectively. Wikis also commonly suffer from the **bystander effect** – that is to say, when everyone is able to create and update content, almost no one does. In summary, wikis are more useful for internal knowledge sharing than for external publishing.

However, Confluence is something of an exception, in that you **can** build a structured site with a bit of care (extensive instructions are available on the Web), and it **does** have a Cloud version that could be turned into a website.

Confluence also has plug-ins for both Git and Markdown (although these are only currently available in the Server incarnation of Confluence).

More about wikis:

- **Wiki Choice Wizard**
- **List of Wiki Software (Wikipedia)**

Conclusion

I am currently leaning towards Confluence as the best tool for hosting 51Degrees' documentation, with the proviso that there must be strong encouragement for everyone to contribute.

However, **ReadTheDocs** and **Siteleaf** are also contenders because of the close integration with Git. However, I would personally find them a bit limited in terms of layout.

However, if we need **context-sensitive help** and/or translation into languages other than English (whether UK or US), then the picture changes and we need to go for MadCap Flare.

DITA

Darwin Information Typing Architecture (DITA) is a type of XML for technical writing. If you need to write in DITA (and the choice may not be up to you), there are only a few tools you can use. Some popular ones are Oxygen and XMetal for writing, and DITAToo for source control.