The future of technical documentation starts with its *recent* past

Carlos Evia

Abstract

This keynote presentation addresses how recent trends to align technical documentation practices with "developer-friendly" workflows may be detrimental to documentation authors and their users. A proposed solution is in the recent past of technical documentation as a discipline, where tools and ideas rooted in structured authoring and markup, reuse, and personalization can still provide solutions to present — and future — needs related to technical content.

1 Introduction

As I was preparing this keynote presentation, I realized that a subtitle — or even alternative title for it — should be "We all owe something to a former IBMer", or the more complimentary "Damn, those former IBMers were right".

I do love talking about the future of technical documentation, but I also enjoy talking about its past, and particularly its *recent* past. Technical documentation is still a very much needed and important product, or genre, of technical communication. As a bigger umbrella term to contain the processes of developing and conveying technical information from experts to a non-technical audience (or the dreaded "laypeople" noun), technical communication is pretty much concerned with documentation, but mainly as part of a universe of genres or products that require that mediation between user and expert that characterizes the job of technical communicators.

And here is where I will go back to the recent past of technical communication. Ten years ago, the Adobe Technical Communication Suite (TCS) team distributed on several social media channels a video titled "Future of Tech Comm". The video used stop-motion animation and fast-draw techniques to summarize the features of "Adobe's Tools and Services" for technical communication. As a pair of rapidly animated hands assembles Lego pieces, the video's narrator describes that "for some, [the future of technical communication] is all about more and more structured content and the ability to work faster and smarter with XML and DITA constructs". Approaching the end of its 2:30-minutes runtime, the video claims that "it is most certainly an exciting future to be in" [1].

Carlos Evia

doi.org/10.47397/tb/43-2/tb134evia-techdoc

2 The Darwin Information Typing Architecture

At this point we encounter a first batch of former IBMers, as the main engine behind the future of tech comm heralded by Adobe in that video was DITA (Darwin Information Typing Architecture). This should be interesting for you as LaTeX users, because it involves a markup language. DITA is an open standard for structuring and publishing technical information. Sounds familiar? DITA was — big surprise — developed by IBM in the late 1990s, and is very much alive as an open standard maintained by the non-profit consortium OASIS (Organization for the Advancement of Structured Information Standards).

In an interview for the website "DITAWriter", Don Day, a former IBMer who was one of the original developers of DITA, chronicled the origins of his XML experiments while working for IBM in the last decade of the 20th century as follows:

> With the advent of XML as a new markup standard in 1998, the Customer and Service Information (C&SI) group began adopting a Tools and Technology mantra under Dave Schell who was the strategy lead. By 1999, Dave was aware of my participation as IBM's primary representative with the XSLT and CSS standards activities at the World Wide Web Consortium, and I delivered a presentation at a formative meeting in California that forecast the possibility of XML to solve IBM's still-lingering problems with variant tools and markup usage [2].

DITA consists of a set of design principles for creating "information-typed" modules at a topic level and for using that content in delivery modes such as online help, technical documentation, and product support portals on the Web. Day explained that, when naming the standard, DITA "represented a great deal of messaging in a compact and memorable acronym":

- **Darwin**: for specialization and how things could "evolve" from a base;
- **Information Typing**: for representation of knowledge as typed units;
- **Architecture**: a statement that this was not just a monolithic design but an extensible tool that could support many uses [2].

IBM eventually donated DITA as an open standard, which is currently maintained by OASIS. DITA, however, "has evolved substantially since that initial donation to encompass a very wide scope of requirements indeed" [7, p. 6]. At the OASIS DITA Technical Committee, the standard continually evolves with the purpose "to define and maintain the Darwin Information Typing Architecture (DITA) and to promote the use of the architecture for creating standard information types and domain-specific markup vocabularies" [8]. At OASIS, a small army of former IBMers (including Kris Eberlein, Eliot Kimber, and Michael Priestley) has kept the DITA standard evolving and with very healthy adoption and usage figures.

Just as in LaTeX we declare a document class to start a file, in DITA we can use different topic types that have specific semantic elements to structure, and later publish, technical information. The literature focuses on three topic types that "represent the vast majority of content produced to support users of technical information" [4, p. 7]: concept, task, and reference, which Pringle & O'Keefe define succinctly as follows:

- **Concept:** contains background information and examples;
- **Task:** includes procedures ("how to" information);
- **Reference**: describes commands, parameters, and other features [9, p. 235].

For authors of technical documentation, these foundational topic types provide constraints and structures beyond a presentation-oriented template. In DITA, authors can create consistent topics to assemble collections of information with elements that can be reused even at the phrase level. For example, a concept could be an introduction to a particular software package, while tasks can provide instructions on how to install and use the software package, and a reference topic can list common extensions and tools associated with the package.

In practical terms, DITA's topic types include XML tags for content "moves" or strategies (such as a short description, steps, and examples) frequently used in technical publications. Pure XML does not provide a defined set of tags, but DITA does offer a catalog of elements and attributes relevant for technical communicators.

The *Darwin* component of DITA is one of its main "selling" points. DITA is customizable for specific situations that will still keep it as part of the standard. Maybe the element types included in the default *task* topic type are too generic for CompanyX. The company can then *specialize* the task type to create, rename its own structural elements that will still validate in DITA-aware tools.

And that is the recent past of technical documentation: workflows and tools based on DITA became mainstream and enabled practices such as

single sourcing, modularity, and multi- or omnichannel publishing. Let me spend a few minutes describing those, and you will see that you can do similar things with LaTeX-based workflows for academic and scientific publication.

2.1 Single sourcing

A team can have a common repository of topics, or even element types (a common legal disclaimer in a paragraph) that many files can use. By referencing the single source, all files that mention it would be automatically updated if there's a change in the source. Much better than copy-paste. We can, of course, do some of that with LaTeX.

2.2 Modularity

Nothing new for LaTeX users here, but in DITA authoring a whole "document" can be composed with pieces from different sources. In the particular case of DITA, most of these aggregation processes will use a file type known as a map, which includes hyper references to topics and other resources (internal or external).

2.3 Multi- and omnichannel publishing

Like DVI on steroids, multichannel publishing is one of DITA's key features. By default, the DITA Open Toolkit can publish to PDF, HTML, Markdown, Eclipse Help, HTML Help, and other formats. Similar to LaTeX, the user community has also contributed with plugins that enable publishing to many other formats. Surprising no one, many of those publishing pipelines involve a stop in LaTeX-land.

Omnichannel publishing is even more interesting, as it enables content-as-a-service approaches that "serve" DITA topics or components via APIs. DITA is particularly good at this because of the semantic value that its XML tags and attributes can provide as metadata for filtering and customization.

3 The present

And this is when we get to the present of technical documentation. Adoption numbers and success stories should not hide that the evolution of technical documentation takes place on a slightly rocky path; even in practitioner circles, there has been pushback and criticism against XML and its relationship with technical communication. In blogs and social media exchanges, some practitioners have questioned the status of XML, and DITA, as the main markup language for information products. While acknowledging DITA's effectiveness as a replacement for large user manuals in complex industries, a few authors lament that "this form of structured content can feel cold and clinical, especially to those from the editorial or marketing side of content" [10, p. 20]. Others argue that in the world of computing code verbose languages are becoming obsolete, but intelligent content still relies on XML and its nested tag structures.

Those are valid concerns. DITA, as it evolves as an open standard, needs to address them and learn from its users. And here we have another former IBMer to the rescue. Michael Priestley, one of the key architects of DITA back in the late 1990s, has been working on a simplified version of the standard known as Lightweight DITA (LwDITA). As a disclaimer, I was involved in the development of LwDITA and worked closely with Priestley for many years. Although I am not an active member of OASIS any more, I am still a DITA and LwDITA peddler (see this keynote presentation as an example!).

LwDITA is a topic-based architecture for tagging and structuring intelligent content using flexible markup options [3]. Lightweight DITA aims to streamline the DITA authoring experience by presenting three formats for content creation:

- **XDITA**, an XML format with a subset of DITA elements that can be used for validated authoring and complex publishing chains;
- **HDITA**, an HTML5 format that can be used for either authoring or displaying content;
- **MDITA**, a Markdown format with a subset of XDITA elements that can be used for maximizing input readability while maintaining structure in content.

An author does not need to use all three "flavors" at the same time to adopt LwDITA. They can work in HDITA all the time and they would still be using LwDITA. Authors can live in an MDITA environment without XML or HTML tags and would still be using LwDITA. All three LwDITA formats are compatible with each other and with DITA XML. For a team of authors with diverse technical backgrounds and communication skills, the different formats of LwDITA allow collaboration and content exchange in a centralized solution. For example, CompanyX can hire a technical writer to create instructions in XDITA (based on XML) while a marketing professional writes a description of the app's features in HDITA (based on HTML5), and an engineer uses MDITA (based on Markdown) to create a basic API reference. All their topics are treated as DITA and can take advantage of the standard's reuse, filtering, and single-sourcing capabilities.

Now, yet another former IBMer has come out to warn users about going too lightweight. Michael

Iantosca warns about the impending doom of Markdown and reStructuredText (another lightweight authoring format that has become popular with developers and some technical authors). Iantosca [6] states that the degree of granularity and flexibility that what he labels as cognitive content requires cannot happen with lightweight languages. Iantosca defines cognitive content as follows:

> [...] a strategy, an architecture, and an operational model. It enables dynamic, machine-based discovery, mining, analysis, retrieval, assembly, and delivery of non-linear content objects using advanced semantic technologies that rely on predictive relationships between content objects and inbound signals [5].

3.1 Conclusion

And that's where DITA excels. But this requires going back to the recent past of technical documentation and to the future of "tech comm" that Adobe heralded in its promotion video. Let lightweight languages work in a LwDITA-like environment and save the intense structure for DITA.

3.2 What does this mean for LaTeX users?

I am a big proponent of ventilating silos, and here I see Markdown as a good bridge to connect the academic and scientific typesetting of LaTeX with the technical documentation of DITA and LwDITA. The `markdown` package is your friend if you need (or want) to use modules from a single source repository. If you need stronger features to achieve cognitive content or omnichannel publishing, then you can move to XDITA and full DITA XML. That's how I see the future of technical documentation.

References

[1] Adobe TCS. Future of Tech Comm, Jul 2012. https://www.youtube.com/watch?v=dSdhnyDF0YY

[2] DITA Writer. Don Day and Michael Priestly on the beginnings of DITA: Part 2, Oct 2018. https://www.ditawriter.com/don-day-and-michael-priestly-on-the-beginnings-of-dita-part-2/

[3] C. Evia. *Creating intelligent content with Lightweight DITA*. Routledge, New York, NY, 2019.

[4] J.T. Hackos. *Introduction to DITA: A user guide to the Darwin Information Typing Architecture including DITA 1.2*. Comtech Services, Denver, CO, 2011.

[5] M. Iantosca. A future powered by knowledge graphs. BrightTALK. https://www.brighttalk.com/webcast/9273/525482

[6] M. Iantosca. Mark-Duh? The impending doom of Markdown and reStructred Text, Mar 2022. https://thinkingdocumentation.com/blog/f/mark-duh-the-impending-doom-of-markdown-md-and-restrucured-tex

[7] E. Kimber. *DITA for practitioners*. XML Press, Laguna Hills, CA, 2012.

[8] OASIS Open. OASIS Darwin Information Typing Architecture (DITA) TC. https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=dita

[9] A.S. Pringle, S. O'Keefe. *Technical writing 101: A real-world guide to planning and writing technical content*. Scriptorium Press, Research Triangle Park, NC, 2009.

[10] S. Wachter-Boettcher. *Content everywhere: Strategy and structure for future-ready content*. Rosenfeld Media, Brooklyn, NY, 2012.

◇ Carlos Evia
Virginia Tech
Blacksburg, VA 24061
USA
cevia (at) vt dot edu
https://carlosevia.com