Structured Authoring without XML: Evaluating Lightweight DITA for Technical Documentation

By Carlos Evia, Virginia Tech and Michael Priestley, IBM

Abstract

Purpose: We present a proposal for HDITA, an HTML5-based version of the Darwin Information Typing Architecture (DITA). We report on an exploratory study about the feasibility of using HDITA as an authoring platform for technical content. We asked how novice technical writers describe and evaluate the complexity and difficulty of the different stages of the HDITA authoring process and if novice technical writers can author effective topic-based technical content in HTML5 (HDITA) without full knowledge of XML (DITA).

Method: To evaluate the feasibility of authoring and publishing with HDITA, we modified the Instructions assignment of an introductory college course called Technical Writing. Students wrote blog posts during the authoring process and completed a survey on the perceived difficulty of HDITA. We evaluated the quality of HDITA Web deliverables with college students from diverse technical and academic backgrounds.

Results: Most author students were somewhat confident authoring technical content with HDITA, and most said they were very likely to somewhat likely to use HDITA in the future for technical writing projects. Students reported that the most difficult part of using HDITA involved Web templates and not HDITA itself. Twenty-seven students evaluated HDITA deliverables and gave them positive scores using a rubric for assessing quality technical information.

Conclusion: Acknowledging the small number of student authors involved in this feasibility study, we can still conclude that novice technical writers did not perceive creating technical documentation with HDITA as difficult or highly complex. Most student evaluators were able to complete the assigned tasks following the instructions created in HDITA.

Keywords: DITA, structured authoring, XML, HTML5, feasibility study

Practitioner's take-away:

- HDITA is an experimental version of Lightweight DITA that uses HTML5 tags instead of XML for structuring technical content.
- Students in the Technical Writing course, perceived as novice technical authors, created sample procedural projects using HDITA for Web deliverables.
- Novice technical writers did not find the HDITA authoring process particularly difficult, and their deliverables received positive comments and feedback from evaluators.
- HDITA can simplify the technical authoring process while producing effective deliverables. Authors without XML experience (such as Web developers, programmers, etc.) can use DITA for projects.

Structured Authoring without XML

Introduction

Publications from technical communication research and practice frequently discuss the pros and cons of structured authoring, single sourcing, topic-based architecture, and component content management systems. Extensible Markup Language (XML) is often mentioned as a key enabler of that kind of authoring and publication work. Not without problems and criticism, XML has been in a mainly stable relationship with the field of technical communication for over a decade. Some have even said that a "thorough understanding" of XML is essential for students of technical writing (McDaniel, 2009, p. 6). However, XML has also been labeled "cumbersome and complex" (Johnson, 2015, para. 24) and even had to be defended (O'Keefe, 2010) when it was blamed for the death of creativity in technical writing.

The Darwin Information Typing Architecture (DITA) is an XML-based, "end-to-end architecture for creating and delivering modular technical information" (Priestley et al., 2001, p. 354). Originally developed by IBM, DITA is now an open standard maintained by the non-profit Organization for the Advancement of Structured Information Standards (OASIS). As one of the main XML grammars used for technical communication purposes, DITA is also the recipient of mixed comments from practitioners.

For creators of technical content, DITA offers the following benefits: it streamlines the content creation process, it increases the quality of content by standardizing it, and it allows authors to leverage content in many different ways, which include reusing it, publishing it in different formats, and translating it (Samuels, 2014). From a managerial perspective, Hackos presents a list of DITA's business advantages, which suggest that DITA will "[promote] the reuse of information quickly and easily across multiple deliverables," "reduce the cost of maintaining and updating information," "enable continuous publishing," "share information across the global enterprise," "reduce the cost of localization," and "reduce the technical debt caused by inadequate, incorrect, and unusable legacy information," among others (2011, p. 10). Despite DITA's benefits for the technical communication workflow, and its many industry evangelizers and users, recent articles from the technical communication blogosphere have characterized DITA as "overly complex" (Kaplan, 2014, para. 13) and "far more complex than it needs to be in almost every dimension" (Baker, 2014a, Responses, para. 7).

Identifying (and reducing) DITA's complexity is more difficult than it might appear: each of DITA's features has its adopters and defenders, and DITA's detractors rarely agree on which features are too complex for technical authors. But within the OASIS DITA Technical Committee, there is general agreement on the value of a lightweight entry point to DITA that preserves some key features, including DITA differentiators such as specialization and robust reuse mechanisms, while providing an easy upgrade path to a more complete feature set. Co-author Priestley formed the Lightweight DITA subcommittee at OASIS to focus on trimming the element and feature list of DITA to the minimum necessary to enable core reuse capabilities in areas such as education, marketing, and manufacturing. Another focus of the subcommittee is to free the lightweight specification from a dependency on XML, which allows DITA as a standard to exist across formats, including HTML5 and Markdown.

Lightweight DITA is still in development and not ready for mass dissemination. However, many of the ideas it represents are already mature enough to be tested.

In this paper, we present a proposal for an HTML5-based version of Lightweight DITA, named HDITA, and report on an exploratory study about the feasibility of using HDITA as an authoring platform for technical content. Our study took place during an online introductory technical writing course, taught by co-author Evia, at a major research university in the United States. Our main objective in sharing this proposal and preliminary results with readers of *Technical Communication* in industry and academia is to discuss the feasibility of creating technical content in HDITA and obtain feedback on the potential impact of Lightweight DITA for their work processes and environments. The Lightweight DITA subcommittee will also use that feedback for further evaluation of simplified DITA concepts with larger groups and industry settings.

We designed and conducted this research project primarily as an authoring study. Although we evaluated the quality and effectiveness of technical content produced in the HDITA workflow, our main objective was to focus on the author experience as non-expert technical writers encountered structured authoring

for the first time. We elaborate on the strengths and limitations of DITA in the practice and outcomes of technical communication.

Technical Communication and DITA XML

The relationship between technical communication and XML (and its predecessor SGML) covers almost two decades. A historical perspective of XML's contributions to the field of technical communication is beyond the scope of this feasibility study. However, some major milestones worth noting in this relationship include the explicit role that XML took in *Information Development: Managing Your Documentation Projects, Portfolio, and People* (2007), the revised version of JoAnn Hackos's seminal book *Managing your Documentation Projects* (1994). Whereas the 1994 book talked about electronic publishing in general terms, the 2007 book includes explicit connections to XML and DITA, not just as tools for writing but also as part of a documentation management methodology. Rockley (2003; 2012), Pringle & O'Keefe (2003; 2009), Self (2011), Vazquez (2009), Kimber (2012), White (2013), Bellamy et al. (2012), Hackos (2011) and others have published and revised books for practitioners emphasizing and expanding coverage on the importance of XML (and DITA) in technical documentation and content management work environments.

DITA "is a technical documentation authoring and publishing architecture that is based on principles of modular reuse and extensibility" (Priestley et al., 2001, p. 352). DITA's modular structure is based on a generic topic type that could describe almost any content, from which are derived three main information or topic types: concept, task, and reference, which "represent the vast majority of content produced to support users of technical information" (Hackos, 2011, p. 7). In a DITA authoring environment, writers create "technical content by assembling topic-oriented information types or blocks of information that serve particular functions in a document. A step in a set of instructions and an ingredient in a recipe are examples of information types" (Swarts, 2010, p. 133). As part of technical genre development, a DITA-based authoring system ensures that all the parts are present and that parallel information can be recognized.

Bellamy et al. summarize the main benefits of writing in a topic-based environment with DITA for users and authors as follows: DITA allows users to "find the information they need faster, accomplish their goals more efficiently, [and] read only the information they need to read." DITA enables technical writers to "maintain and reuse topics more effectively, organize or reorganize topics more quickly, [and] share and distribute the work on topic files more easily, which increases writer productivity" (2011, p. 17).

There are no official figures about the usage and adoption of DITA as a platform for authoring technical documentation. Keith Schengili-Roberts, an information architect known online as *DITAWriter,* maintains an "informal list of firms that are using DITA XML in some form in their documentation efforts" (2015a, para. 1). As of July 2015, the list included 532 companies. Schengili-Roberts analyzed LinkedIn profiles of thousands of technical writers, and his findings imply "that potentially there are 1,400–3,000 firms worldwide currently using DITA" (2015b, para. 3).

The next section presents critical perspectives against XML and DITA. Some of the counterarguments presented in the following section inspired the simplified structured authoring approach evaluated in this study.

The case against XML and DITA

As members of the OASIS DITA Technical Committee, we work on advancing the DITA standard and teaching novice technical writers about its many benefits for authors, managers, and users. However, we cannot ignore feedback and counterarguments. Despite XML and DITA's strong presence in technical communication, some practitioners argue, among other things, about the complexity of DITA's many XML tags, the need for specialized tools to produce end-user deliverables, and the high learning curve for specializing DITA beyond the core concept, task, and reference topic types.

In 2007, technical and fiction writer Larry Kollar hit a nerve in the field of technical communication with a series of blog posts titled "XML Heresies." The Yahoo! blogging platform hosting the posts has gone offline since. Nevertheless, their author agreed to share the original "heresies" via email (L. Kollar, personal communication, February 22, 2015). Kollar questioned the need for XML in documentation projects and pondered if the fixed structured imposed by XML was actually beneficial to the profession: "Do (XML-based publishing systems) free us to write better documentation, or do they stifle the creativity that's

essential for human-to-human knowledge transfer?" (Kollar).

Consultants and structured authoring advocates reacted to Kollar's comments, and some said he displayed "a very myopic view of technical writing," as reported by Abel (2007, para. 5). A few years later, Sarah O'Keefe wrote a column for *Intercom* defending XML from critics who called it "the death of creativity in technical writing." In XML-based authoring environments, O'Keefe argued, technical communicators "have the most opportunity for creativity in crafting sentences, paragraphs, topics, and groups of topics that explain complex concepts to their readers" (O'Keefe, 2010, p. 37).

More recently, practitioners lament that, whereas programming and scripting languages move toward simplified syntax and tagging systems, technical communication continues to rely on XML and complex, nested tag structures.

> What are we seeing? Simplification. Ease of use. A learning curve that gets less steep every time. Languages that drop features that aren't used, or aren't used often. And what has techcomm poured resources into? DITA. An arcane, overly complex language with a massive learning curve that requires specialized tools. (Kaplan, 2014, para. 13)

Popular technical communication bloggers Mark Baker and Tom Johnson have articulated arguments about the perceived complexity of XML and, particularly, DITA. According to Baker, "XML is not the answer. Structured writing may be the answer. XML is one way to implement structured writing" (2014b, para. 1). After experimenting with DITA as an authoring platform, Johnson authored a blog post titled "10 reasons for moving away from DITA." Although some of Johnson's claims come from his confounding DITA the XML standard with DITA-aware tools, Johnson's message is loud and clear: "Writing in XML is more cumbersome and complex" (2015, para. 24).

No single technical writer, especially not a novice one, is expected to learn all available DITA tags in order to use DITA as an authoring and publishing platform. While a generic DITA topic only requires the XML identifier and a title, the main complaints against the standard focus on the increasing page count for its spec, which was in part due to adding a smaller "starter" document type to the package. Therefore, the challenge for the DITA Technical Committee was how to decrease complexity for new users of DITA when efforts to provide easier entry points were just perceived as added complexity.

The answer was to target Lightweight DITA not as another set of document types within the DITA specification but as a separate specification that would be judged on its own merits. The following section introduces the proposal for HDITA, which is an HTML5-based version of Lightweight DITA.

HDITA: A Proposal

The idea of a simplified version of DITA to reduce the documentation standard's learning curve has been around for a few years. Back in 2011, the DITA Technical Committee was talking about a "limited DITA profile," which was still XML-based, but depended heavily on HTML tags (such as <p> and <li>) to simplify many semantic structures of full DITA. As the concept of Lightweight DITA developed further, at one point it became an XML sub-set of DITA that included, for example, 27 possible elements inside a topic, whereas full DITA includes a possible combination of 90+ elements. Originally, Lightweight DITA was planned as a component of the DITA 1.3 specification, but interest from members of the DITA Technical Committee, vendors, and researchers pushed it out of the main specification and into its own parallel and compatible standard. The purpose of Lightweight DITA is not to replace full DITA XML. If anything, Lightweight DITA provides basic DITA access to authors who do not need all the standard's features but whose deliverables should be compatible with full DITA XML, if needed.

In 2014, co-author Priestley introduced a proposal to align a lightweight DITA profile in XML with an equivalent markup specification based on HTML5. While XML-based publishing chains remain the industry standard for many content-centric industries (such as publishing, pharmaceutical, and aerospace), this proposal responded to concerns about their complexity, especially as a barrier to new adopters or contributing authors.

The challenges with the HTML5-based approach were based on a lack of standardization: each new extension of HTML5 introduces its own additional semantics and constraints, locking the content into a particular tool or vendor pipeline. The additional

semantics and constraints may also require a custom authoring environment, resulting in another barrier to content portability, without the advantages of authoring-time validation that an XML-based approach provides. Finally, even though the approach may eliminate processing steps for the case of simple content, more complex content scenarios—such as content reuse and filtering, or indexing and link redirection—require additional processing steps that reintroduce the complexity of an XML-based approach, without the advantage of existing standards-based solutions.

This proposal suggests a third way: defining both a lightweight XML model based on DITA that can be used for validated authoring and complex publishing chains and a lightweight HTML5 model that can be used for either authoring or display.

The two schemes—provisionally named XDITA and HDITA—are designed for full compatibility with each other as well as conformance with the OASIS DITA and W3C HTML5 standards. They give HTML5 users a set of standardized mechanisms to access the power and flexibility of DITA's reuse and specialization capabilities and give DITA users a way to integrate and interact with HTML5-based content systems without complex mapping or cleanup steps.

XDITA, still in an experimental stage, is outside the scope of this paper and will be implemented and evaluated in the future. The feasibility study reported in this manuscript applies exclusively to HDITA, because HTML5 and DITA are now close enough to achieve a reasonable and semantic mapping with the application of a few simple constraints. Figure 1 compares a basic topic in DITA XML to a simplified topic in HDITA.

Taking advantage of HTML5's custom data attributes, HDITA allows topic specialization without XML tags or advanced metadata. The resulting topics are easier to author and read by writers with basic HTML knowledge and can be parsed through a browser immediately in simple Web versions. We hypothesize that a structured-authoring language based on HTML5 can make DITA more accessible and easier to adopt by technical writers and Web professionals who work in HTML but are not familiar with XML. Figure 2 shows the specialized DITA task topic and the same simplified task topic in HDITA.

Summarizing, HDITA is not presented as a potential replacement for DITA XML. HDITA's main characteristics and advantages are as follows:

- Simplified authoring by using a small set of semantic tags and attributes in HTML5.
- Instant presentation layer for basic deliverables without the need of a transformation process.
- Compatibility with XDITA or full DITA XML for advanced processing and filtering.
- Possibility of using a wide variety of commercial and open source editors for HTML5 instead of specialized tools.

Figure 1: DITA topic compared to HDITA simplified topic

DITA Topic	HDITA Topic
<topic> <title>The point of it all</title> <shortdesc>I can sum it up here</shortdesc> <body> I can say some more stuff <section> <title>Stuff</title> And so on - This - Is - A List </section> </body> </topic>	<article> The point of it all I can sum it up here I can say some more stuff <section> Stuff And so on - This - Is - A List </section> </article>

Figure 2: DITA task compared to HDITA task

DITA Topic	HDITA Topic
<task> <title>How to do something</title> <shortdesc>Introduction to this specific task</shortdesc> <taskbody> <context>Use only when ready</context> <steps> <step> <cmd>Plan something</cmd> </step> <step> <cmd>Do something</cmd> </step> <step> <cmd>Evaluate something</cmd> </step> </steps> <example>Like this</example> </taskbody> </task>	<article data-hd-class="task"> How to do something Introduction to this specific task <section data-hd-class="task/context"> Use only when ready </section> <section data-hd-class="task/steps-informal"> - Plan something - Do something - Evaluate something </section> <section data-hd-class="topic/example"> Like this </section> </article>

- Potential for involving professional communities of Web developers and programmers, who do not necessarily work with XML but most likely are proficient in HTML.

The next section focuses on the benefits of HDITA for technical authors, and it presents the conceptual model for content creation we evaluated in this study.

The HDITA Author Experience

To minimize the potential complexity of authoring and displaying technical content, HDITA aims to simplify the learning curve of DITA XML while preserving its key benefits of modularity and reuse. In the introduction to their edited collection *Content Management: Bridging the Gap between Theory and Practice*, George Pullman and Baotong Gu talk about new demands for technical communicators. When discussing the effects of content management systems on the balance of creation and delivery of information, Pullman and Gu address the "decontextualization at the input stage and the recontextualization at the output stage" of content delivered through a CMS (2009, p. 8). In an authoring workflow with DITA XML, for example, a team of writers could create separate topics with related steps that, when transformed into deliverables, can appear together in one single task as a PDF or a concept as an HTML file. With HDITA, authors can still create content in such an assembly-like environment. However, they can also display content directly in the language (HTML5) in which it was originally written.

That unpacking and repacking of content is at the core of Rick Yagodich's *Author Experience: Bridging the Gap between People and Technology in Content Management*. Yagodich introduces a content communication process that flows from communicator to audience (and vice versa). Yagodich's model includes three parts: input, storage, and output. In the example DITA authoring system from the previous paragraph, *output1* could be a PDF filtering information for a specific type of audience, and *output2* could be a website with information for all audiences. An author in such a process could be disconnected from the final output of her content, and her input work could be limited to what the storage tags and constraints expect from her. Yagodich suggests the creation of a translation layer between the storage and the various output channels and yet another "layer of translation so that the input logic and paradigms make sense to the authors, rather than simply mirroring the storage model" (2014, p. 4).

Yagodich's model, however, resembles Shannon and Weaver's linear process of communication (1949), which has been criticized for treating communication as a mechanical exchange without focusing on the production of meaning (Fiske, 1990) or reducing meaning to content delivered without "allowance for the importance of social contexts and codes" (Chandler, 2002, p. 176). Rebekka Andersen (2014) presents a more semiotic and humanistic model of a similar workflow (a content management system and the content strategy framework that supports it) with content authors (users can be included in this category) and a reception stage with deliverables for end-users that is beyond the transmission's input and output. The middle storage point is represented by a more complex combination of an XML repository, an automated assembly and publishing server, and a delivery engine.

Combining the work of Yagodich and Andersen, we created the conceptual model for an authoring experience with HDITA:

- Authoring/input: Content authors will be directly connected to the repository by using HTML5 for both processes instead of XML.
- Process/storage: Content assembly and delivery sections will be more transparent when authors have direct control over simple text maps and filtering capabilities.
- Reception/output: The delivery of end-user products with customized content, which can be HTML5 or any transformation provided by DITA XML tools.

We investigated the feasibility of this model in a college-level technical instructions assignment, which is described in the following section.

HDITA feasibility study

In order to evaluate the feasibility of authoring and publishing technical content with HDITA, we modified the Instructions assignment of an introductory technical writing course at a major research university in the United States. The technical writing service course, "which helps future engineers, scientists, and managers succeed in their careers, works with the student's disciplinary knowledge to mediate technology, science, or business for users" (Coppola, 1999, p. 259). The particular section we modified was conducted online and was not for students majoring in Professional and Technical Writing. The students, with subject matter knowledge in technical and scientific concepts, were new to technical writing; therefore, we perceived them as novice technical writers with the potential to become casual or formal practitioners after graduation.

Our research questions for the feasibility study were the following:

1. How do novice technical writers describe and evaluate the complexity and difficulty of the different stages of the HDITA authoring process?
2. Can novice technical writers author effective topic-based technical content in HTML5 (HDITA) without full knowledge of XML (DITA)?

The Technical Writing course's learning objectives and assignments traditionally include a basic HTML writing activity for online presentation of content. They also include a technical instructions project, which focuses on developing procedural information to solve users' problems. Our revised project combined those assignments under the following description:

> You need to write Web-based instructions for a real-life situation. Virginia Tech asked you to develop documentation showing students how to use LibreOffice as an alternative to Microsoft Office or Apple iWork. Your job is to write instructions guiding a first-year college student on how to do the following:
>
> - Download and install LibreOffice
> - Write a letter in LibreOffice Writer
> - Create a simple spreadsheet in LibreOffice Calc
> - Create a presentation in LibreOffice Impress.
>
> You should include examples and pictures, but don't make a whole tutorial on all the LibreOffice features. Instead, focus on specific user tasks.

By combining the introductory HTML project and the Instructions module already included in the syllabus, we did not have to make major modifications to the course. The new content to introduce DITA and HDITA only added a couple of hours to the existing lesson plans. Furthermore, the assignment's purpose was not solely to teach students about HDITA. The module containing this project started with discussions

about rhetorical principles for developing user-oriented tasks. The students read relevant chapters from their course's primary textbook, Markel's *Practical Strategies for Technical Communication* (2013), and from Pringle and O'Keefe's *Technical Writing 101* (2009). When advocating for content about XML in technical writing courses, McShane proposed to combine "the theory (single sourcing), methodology (modular writing), and technology (content management) to support, apply, and guide it" (2009, p. 83). In our project, HDITA influenced all three aspects of that formula, but the assignment was not just about writing HTML5 tags.

The online Technical Writing class had eighteen students (12 male and 6 female; 5 non-native English speakers) whose ages ranged between 19 and 28 years old. Academic majors represented in the course were Mathematics, Computer Science, Electrical Engineering, General Engineering, Biology, Animal and Poultry Sciences, Dairy Science, Environmental Science, Theatre Arts, and Construction Engineering.

All data collection was conducted with approval from the Virginia Tech Institutional Review Board under protocol #14-745.

Prior to this assignment, the students had not been exposed to XML as an authoring tool. Some of them, majoring in computing-related fields, had interacted with XML as a layer in database-driven computing projects. The students had no previous knowledge of DITA. We informed them of the project's experimental nature and the extra lessons they received contained information about DITA's topic structure (including concepts, tasks, and references) and HDITA's syntax. However, the course content did not cover DITA XML. For the layers of automated assembly and delivery, because HDITA is still not integrated into the DITA Open Toolkit or other DITA-aware tools, we introduced the students to Jekyll (http://jekyllrb.com) via GitHub Pages (http://pages.github.com) to take advantage of a template for Web deliverables created for a plugin of the DITA Open Toolkit.

Once the students authored and coded their HDITA topics, we asked them to create maps in YAML syntax (http://yaml.org), which then connected to a Jekyll template we provided for deployment in GitHub Pages. The Jekyll template generated a responsive, mobile-ready website following Andersen's observation about content management leaders, commenting on how Web-enabled mobile devices are "revolutionizing content consumption—and thereby production" (Andersen, 2014, p. 126).

Figure 3 shows a student's sample HDITA code for a task and the transformed version of that topic when seen as the GitHub Pages output on a mobile device.

Figure 3: Student HDITA code and responsive Web deliverable

To answer our research questions, we asked the Technical Writing students to compose reflection blog posts documenting their progress, problems, and accomplishments as they worked on the assignment. Once their projects were submitted, students also had to take a survey about their experiences working with the authoring, processing, and output stages of the assignment. The next section presents results and findings of the feasibility study as we revisit our research questions.

Results and Findings

Describing HDITA's levels of difficulty

Our first measure to evaluate the Technical Writing students' perceptions about complexity in HDITA and the overall difficulty of the Lightweight DITA authoring experience was an optional survey they took after completing the Instructions project. We received twelve usable responses. The following questions asked students about their level of confidence with specific stages of the authoring process and gave them Likert scale-like possible answers (Very confident, Somewhat confident, Neutral, Not very confident, and Not confident at all):

"How confident do you feel authoring topics in HDITA?" Nine students selected "Very confident" (2) or "Somewhat confident" (7), and only two answered "Not very confident" (1) or "Not confident at all" (1).

"How confident do you feel writing task-oriented instructions?" Eight students answered "Very confident," two answered "Somewhat confident," one answered "Neutral," and one responded "Not confident at all."

"Would you use HDITA if you had to write Web-based instructions in the future?" Ten students selected "Very likely" (6) or "Somewhat likely" (4), and two answered "Not very likely" (1) or "Not likely at all" (1).

"Would you recommend HDITA to colleagues or peers who have to write instructions?" Ten students selected "Very likely" (6) or "Somewhat likely" (4), and two answered "Not very likely" (1) or "Not likely at all" (1).

The last question asked student authors to rank the following activities, from easiest to most difficult, in the process of creating the Instructions assignment:

- Authoring/input stage:
  - Analyzing the audience's needs for your project
  - Conducting research for your instructions
  - Writing tasks, concepts, and references
  - Making maps with topics
- Storage/process stage:
  - Learning and using the HDITA tags
  - Uploading your topics to GitHub
- Reception/output stage
  - Presenting Web deliverables in GitHub Pages
  - Editing your deliverables
  - Evaluating the usability of your deliverables (this stage was completed, with modifications, by the first author after the course ended).

"Learning and using the HDITA tags" and "Uploading your topics to GitHub Pages" were consistently ranked as difficult with the latter marked as the most difficult. All other steps were ranked as easy to complete.

We obtained additional, qualitative information to evaluate students' perception of complexity in an HDITA authoring workflow through reflection posts they had to write while working on the Instructions assignment. We collected a total of 38 reflection posts and coded their comments in a stakeholders' relationship table. We documented the positive, neutral, and negative statements made about each stage of our model of content management communication (Table 1).

Table 1: Stakeholders' relationship table with students' qualitative feedback

	Positive	Neutral	Negative
Authoring/input	9	1	1
Process/storage	12	3	6
Reception/output	7	2	1

The survey and reflection posts show a mostly positive student reaction for the authoring/input stage and more neutral to negative statements for the process/storage, with comments like the following (all students' comments are presented as they were written, including any errors or typos):

> Overall, I think HDITA and GitHub are awesome ways to create files and webpages. I really like the format and structure of the files that HDITA allows. Once the initial basics are mastered, I feel like it allows you to help yourself along in a way that most programs do not. (Female student majoring in Mathematics with a minor in Computer Science.)

> So far, HDITA seems fairly straightforward to me, and I like the basic layout of the instructions assignment. However, I am wondering if that is a bad thing - for all I know, I might be doing everything completely wrong. Hopefully not, but if I encounter trouble, I will post and ask my questions! (Female student majoring in Theatre Arts.)

> Being a Computer Science major and dealing with HTML/XML before, getting a grip on HDITA wasn't too challenging for me. I think this is a good paradigm to use for making instructions, because of the formatting capabilities with HDITA. (Male Computer Science student.)

> I have used HTML in the past for various simple projects, but this method makes creating clean, well

> formatted pages a lot less of a headache. Getting everything started has been relatively easy so far. (Electrical Engineering male student.)
>
> I do not have any experience with html at all, in fact this is actually my first time doing anything like this. Once I put everything together, the design looked really professional and I was pleased with the outcome. I felt like I was able to create something nice even as a beginner. It made me feel good that I made it on my own. It was very useful in this way. (Female student in Animal and Poultry Sciences.)

A few comments were neutral to negative about the authoring experience:

> This class is the first experience I have had with HTML, but I have had previous programming experience in other languages so it hasn't been very difficult. However, learning the tags and formatting has been a minor challenge. Initially I thought HDITA was more complicated than it actually is, and I have spent a lot of time doing things 3-4 times when I really only needed to do it once. (Female Computer Science student.)
>
> I largely enjoyed this project, however i was uncomfortable with the reference portion of the repository. I still am not sure if i did this section how it was suppose to be done (, , ,)Overall though, this project was a nice combination of everything else we have done previously. (Male General Engineering student.)

Some students actually wanted to dig deeper into the project and requested an API or a way to customize the template:

> I've worked with HTML and CSS before, as well as a handful of other languages. I only have a single remark about HDITA : could i get a simple API of sorts? Something along the lines of 'this tag does this, this tag does that, etc.' would have helped me out a bunch. Besides that, it was a unique experience. (Male student majoring in General Engineering.)
>
> I think an API would be really helpful for inexperienced HTML users (like myself), because I have spent extra time looking up commands only to later find out they don't even work correctly the way I'm trying to use them (, , ,) I do really like the end result that HDITA produces as it is very professional for amount of time needed for production, and am still having fun customizing my websites. (Female student majoring in Mathematics.)
>
> Upon completing the assignment, I can't say I had any problems with the syntax or understanding of HDITA. I've also used Github in the past, so picking that up was not a challenge for me either. In the future, I'll probably learn how to change how the structure of the page looks to make it prettier. I guess a suggestion would be including how to do this in part of a lesson as something optional for our assignment if we have extra time. Other than that I can say that this assignment was pretty interesting and I had a good time doing it. (Male Computer Science student.)

Negative comments were, as in the survey results, grounded in the translation from storage to output/reception, as students struggled to generate their Jekyll sites in GitHub Pages.

> I have used HTML in the past, and I feel HDITA will be a great tool to help programmers create clean, straight-forward instructional webpages. Once users get used to HDITA's tags they will quickly be able to create websites. So far, the only difficulty I have had was Github related. (Male General Engineering student.)

Some negative comments, however, evolved into positive comments or included troubleshooting recommendations for peers:

> No matter how many time I redo the uploading and setup, github just keeps sending me emails saying page build failure. It took me a whole day to figure this out. I checked that file again, and you know what? There is a space in front of a topic, which it is not supposed to be there. Well, I think that tells us when dealing with things you don't know much about, be really, really, really careful. (Male Electrical Engineering student.)
>
> Well, honestly, so far it's been disastrous. I've gotten a page setup (after about 45 minutes of mocking

> about in virtual mud), and now all that's showing is the readme.md... How can I fix this? any tips are welcome. (Male General Engineering student.)

A couple of minutes later, that same student posted the following comment: "Nevermind, i made a stupid mistake. all fixed now! "

> I do not really understand how to do any of this HTML stuff because I have never used it before. I am not a computer person and I am studying biology, so I don't really get how this will help me in the future. I sort of understand how to do basic html, the kind we did (in an earlier assignment) but doing this HDITA is very confusing. (Female student majoring in Biological Sciences.)

Eight minutes later, the student added the following: "Writing the code was easier than I thought it would be but it took me forever to get it to sync with github."

Creating effective HDITA documentation

In order to evaluate the effectiveness of projects created in HDITA by novice technical writers, we first graded them based on a rubric developed using Markel's guidelines for creating instructions (2013) and Pringle & O'Keefe's recommendations for writing task-oriented information (2009). The average grade on the project was 16.4 on a scale of 0 to 20 points. Students' grades were not affected by their opinions of HDITA as an authoring platform, and appropriate use of HDITA tags was just one item in the grading rubric. Two students dropped the course before this assignment, so we collected a total of 16 HDITA submissions. Common errors included those regularly seen in the first draft of an instructions assignment in similar courses, but errors related to HDITA tags were also frequent. Table 2 shows some frequent and relevant errors found in the different stages of the HDITA authoring process.

A former graduate teaching assistant in the Professional and Technical Writing program at Virginia Tech conducted a second round of grading once the course ended. The average grade on this round was 17.2 on a scale of 0 to 20 points.

For end-user evaluation of quality, we conducted sessions with 27 students enrolled in courses in computer science, statistics, and English. We recruited students in several introductory summer courses, and participants received extra credit (one numerical point on their final term grade) for completing a quality review of online instructions for installing LibreOffice written in HDITA. The evaluators were 12 male and 15 female students whose ages ranged from 18 to 34 years old. These students were not enrolled in the Technical Writing course at the time of the evaluation, and they were assigned randomly selected HDITA projects from Technical Writing students who gave us written permission to share their work. The student evaluators were asked to conduct the following activities:

1. Answer a questionnaire about their previous experience with the subject matter of the HDITA deliverables (installing and using office software).
2. Follow the instructions to (a) download and install LibreOffice, and (b) complete at least one of the following tasks: write a letter in LibreOffice Writer, create a simple spreadsheet in LibreOffice Calc, or create a presentation in LibreOffice Impress (evaluators self-reported on this step).
3. Complete a survey based on the IBM quality checklist for evaluating technical documentation (Carey et al., 2014).
4. Provide optional information about positive and negative aspects of the HDITA deliverables.

Table 2: Frequent and relevant errors found in HDITA deliverables

Stage	Error	Frequency (in 16 student projects)
Authoring/input	Some steps are not written in imperative mood or not properly numbered	9
Storage/process	Code displays improper use of HDITA tags	5
Authoring/input	The deliverable presents serious spelling or grammar problems	4
Authoring/input	Content is too technical for the target audience	4
Authoring/input	Topics lack images to explain concepts or tasks	2
Reception/output	Web deliverable not displaying in GitHub Pages	1

To determine the evaluators' experiences with the HDITA deliverables' subject matter, we asked them if they had ever interacted with LibreOffice or similar office suites and how comfortable they were with their skills installing new software on their computers. No student evaluator had previous experience with LibreOffice, but 93% of them had worked with Microsoft Office, 81% with Google Docs, 33% with Apple iWork, and 11% with IMB Collaboration/Lotus. Fifteen percent of the student evaluators said they were very good at downloading and installing software, 37% described themselves as good, 33% as fair, 7% as poor, and 7%, as very poor.

When the quality evaluation sessions concluded, 5 of the 27 student evaluators said they could not complete the tasks as written. Three of those five evaluators were using Mac computers and received randomly assigned instructors for Windows. Another evaluator said she needed more details about the tasks, and the fifth said he couldn't find what he was looking for on the website's menu.

Once the student evaluators conducted tasks with the HDITA deliverables, they completed a survey based on the IBM quality checklist for evaluating technical documentation (Carey et al., 2014). On a scale of 1 to 10, where 1 was "among the worst" and 10 was "among the best, could be used as a model," student evaluators had to rank the randomly assigned HDITA projects on the following characteristics:

- Easy to use (task orientation, accuracy, completeness).
- Easy to understand (clarity, concreteness, style).
- Easy to find (organization, retrievability, visual effectiveness).

Table 3 aggregates the scores received for the randomly assigned HDITA deliverables. The scores emphasize the overall average and minimum value assigned to the projects.

Table 3: Mean and minimum score received on quality documentation characteristics

Quality characteristic	Mean score	Minimum score received
Easy to use	7.26	4
Easy to understand	7.44	4
Easy to find	7.31	3

When asked to provide additional information on the elements they found difficult to follow in the online instructions created with HDITA, the student evaluators commented on the placement of images, the limitations of a side menu bar as the only available navigation tool, and the fact that the instructions they received after the random selection were written for a different operating system than the one installed on their laptops. When asked about elements that they found easy to follow, student evaluators commented on the detailed "step-by-step nature" of the instructions, the use of screen captures with examples, and the overall professional look of the websites ("It reminded me of something I would see from a major software company," said one of the evaluators).

Next, we present the study's conclusions and offer recommendations for further research with Lightweight DITA.

Conclusion and Recommendations

Acknowledging the small number of student authors involved in this study, we can still conclude that creating technical documentation with HDITA was not perceived as difficult or highly complex by novice technical writers. The deliverables produced in HDITA were similar in content to those produced in other introductory technical writing courses with tools like Microsoft Word. However, the students involved in this project created responsive Web-based instructions that they could filter and manipulate using text-based maps. The students also learned about the benefits of structured authoring and topic-based writing without the additional layer of complexity of XML.

Students struggled with the translational component that assembled their topics into websites via Jekyll and GitHub Pages. Nevertheless, most of their comments reflected success after a few attempts. This problem was unique to the feasibility study, because HDITA is still not integrated with DITA-aware tools. In future work, we do not foresee using Jekyll as a translational layer in Lightweight DITA.

Most student evaluators were able to complete the assigned tasks following the instructions created in HDITA. Evaluators reported problems related to the deliverables' navigation and image layout, which can be attributed to the Jekyll template and not to the quality of content and structure in the HDITA source authored by novice technical writers.

We recognize that a classroom environment, compared with the workplace, has great time constraints, and faculty worry about using class time to teach new tools or technology at the expense of course content. Our results show that, combining the already existing modules in HTML and instructions from the syllabus, the HDITA elements included in the course supported the content goals and enhanced the students' technical writing experience. We hope these findings also help practitioners, as HTML is a skill more frequently desired than XML in technical writing and content authoring positions. HDITA represents a step forward in combining the advantages of DITA with the popularity and easiness of HTML.

Benefits of generating Web documentation in HDITA instead of plain HTML5, or even a combination of HTML5 and Jekyll, include access to the transformations from the DITA Open Toolkit and community plug-ins. Once the online course ended, we selected three student projects (with proper permission from the authors) and converted them to DITA XML through Jotsom (http://jotsom.com), an experimental online authoring and transformation environment developed by Don Day, who is also a member of the OASIS Lightweight DITA subcommittee.

The HDITA topics created by the Technical Writing students generated valid DITA XML files in which, for example, an HDITA *<article data-hd-class="task">* tag was an actual DITA *<task>* task, and an HDITA *<section data-hd-class="task/steps-informal/ul/li">* tag was an actual DITA *<step>* tag. This consistency in structure cannot be achieved by converting plain HTML5 to XML. Figure 4 shows the DITA XML generated from a student project and a PDF transformation of that topic created with the DITA Open Toolkit.

A major limitation of this feasibility study was the small size of the class. However, this was the first-ever hands-on evaluation of Lightweight DITA. Exploratory studies like the one reported in this manuscript are essential as the Lightweight DITA subcommittee continues working on approaches to minimize complexity for authors of a widely used, international, open documentation standard such as DITA. Another limitation is that HDITA is not yet included in the DITA Open Toolkit, and the conceptual model we used in this study did not allow for advanced DITA features like conditional processing and advanced content reuse. However, student projects created in HDITA can be transformed to DITA XML and can take advantage of those features.

Figure 4: HDITA topic converted to DITA XML via Jotsom (top) and its PDF transformation (bottom) via the DITA Open Toolkit

Although Lightweight DITA does not replace the functionality of DITA XML, an authoring model similar to the one presented in this paper could be beneficial for practitioners currently using DITA or considering adopting the standard. For current users, HDITA provides an easy way to invite collaborators who are not proficient in XML but work in HTML, including Web developers and authors or programmers and application developers. The students who were comfortable writing HTML or who had programming experience quickly embraced the concept of structured authoring with HDITA. For potential DITA adopters, HDITA represents an entry to the standard without specialized tools. Students authored effective HDITA topics in editors ranging from Windows Notepad to Sublime Edit, and the HTML5 foundation allowed instant basic reader view on a Web browser.

Next steps in this work include obtaining and implementing feedback from practitioners and testing HDITA with larger groups, most likely in small companies or non-profits needing documentation. Furthermore, the subcommittee is working on XDITA, a lightweight XML model, and MarkDITA, which generates DITA-like deliverables from content created in Markdown.

Acknowledgements

Carolyn Rude and Russell Willerton provided helpful feedback on drafts of this article. We also acknowledge the contributions of our colleagues from the Lightweight

DITA subcommittee at OASIS. The Jekyll template used for the class experiment was based on a DITA Open Toolkit plugin created by Jarno Elovirta and modified by students sponsored by a seed grant from the Virginia Tech Institute for Society, Culture and Environment.

References

Abel, S. (2007, March 19). Larry Kollar's XML heresy: Structure can take care of itself (Can't it?). Retrieved from http://thecontentwrangler.com/2007/03/19/larry_kollars_xml_heresy_structure_can_take_care_of_itself_cant_it/

Andersen, R. (2014). Rhetorical work in the age of content management: Implications for the field of technical communication. *Journal of Business and Technical Communication*, *28*(2), 115–157. doi: 10.1177/1050651913513904

Baker, M. (2014a, November 12). What kind of "easy" authoring are you looking for? Retrieved from http://everypageispageone.com/2014/11/12/what-kind-of-easy-authoring-are-you-looking-for/

Baker, M. (2014b, January 27). XML is not the answer. Retrieved from http://everypageispageone.com/2014/01/27/xml-is-not-the-answer/

Bellamy, L., Carey, M., & Schlotfeldt, J. (2012). *DITA best practices: A roadmap for writing, editing, and architecting in DITA*. Boston, MA: IBM Press/Pearson Education.

Carey, M., McFadden Lanyi, M., Longo, D., Radzinski, E., Rouiller, S., & Wilde, E. (2014). *Developing quality technical information: A handbook for writers and editors, third edition* (3rd ed.). Upper Saddle River, NJ: IBM Press.

Chandler, D. (2002). *Semiotics: The basics*. London: Routledge.

Coppola, N. (1999). Setting the discourse community: Tasks and assessment for the new technical communication service course. *Technical Communication Quarterly*, *8*(3), 249–267.

Fiske, J. (1990). *Introduction to communication studies* (2nd ed.). London: Routledge.

Gesteland McShane, B. (2009). Why we should teach XML: An argument for technical acuity. In G. Pullman & B. Gu (Eds.), *Content management bridging the gap between theory and practice* (pp. 73–85). Amityville, NY: Baywood.

Hackos, J. (1994). *Managing your documentation projects*. New York, NY: Wiley.

Hackos, J. (2007). *Information development managing your documentation projects, portfolio, and people.* Indianapolis, IN: Wiley Technology.

Hackos, J. (2011). *Introduction to DITA: A user guide to the Darwin Information Typing Architecture including DITA 1.2* (2nd ed.). Denver: Comtech Services.

Johnson, T. (2015, January 28). 10 reasons for moving away from DITA. Retrieved from http://idratherbewriting.com/2015/01/28/10-reasons-for-moving-away-from-dita/

Kaplan, N. (2014, May 3). The death of technical writing, part 1. Retrieved from http://customersandcontent.com/2014/05/03/the-death-of-technical-writing-part-1/

Kimber, E. (2012). *DITA for practitioners. Volume 1: Architecture and technology*. Laguna Hills, CA: XML Press.

Markel, M. (2013). *Practical strategies for technical communication*. Boston, MA: Bedford/St. Martin's.

McDaniel, R. (2009). Experiences with building a narrative web content management system: Best practices for developing specialized content management systems (and lessons learned from the classroom) In G. Pullman & B. Gu (Eds.), *Content management bridging the gap between theory and practice* (pp. 15–42). Amityville, NY: Baywood.

O'Keefe, S. (2010). XML: The death of creativity in technical writing? *Intercom*, *57*(2), 36–37.

Priestley, M., Hargis, G., & Carpenter, S. (2001). DITA: An XML-based technical documentation authoring and publishing architecture. *Technical Communication*, *48*(3), 352–367.

Pringle, A., & O'Keefe, S. (2003). *Technical writing 101: A real-world guide to planning and writing technical documentation* (2nd ed.). Research Triangle Park, NC: Scriptorium Press.

Pringle, A., & O'Keefe, S. (2009). *Technical writing 101: A real-world guide to planning and writing technical content* (3rd ed.). Research Triangle Park, NC: Scriptorium Press.

Pullman, G., & Gu, B. (2009). Introduction: Mapping out the key parameters of content management. In G. Pullman & B. Gu (Eds.), *Content management bridging the gap between theory and practice* (pp. 1–12). Amityville, NY: Baywood.

Pullman, G., & Gu, B. (Eds.). (2009). *Content management bridging the gap between theory and practice.* Amityville, NY: Baywood.

Rockley, A. (2003). *Managing enterprise content: A unified content strategy*. Indianapolis, IN: New Riders.

Rockley, A., & Cooper, C. (2012). *Managing enterprise content: A unified content strategy* (2nd ed.). Berkeley, CA: New Riders.

Samuels, J. (2014, February 3). What Is DITA? Retrieved from http://techwhirl.com/what-is-dita/

Schengili-Roberts, K. (2015a, April 29). Companies using DITA. Retrieved from http://www.ditawriter.com/companies-using-dita/

Schengili-Roberts, K. (2015b, February 3). How widespread is DITA usage? Retrieved from http://www.ixiasoft.com/en/news-and-events/blog/2015/how-widespread-dita-usage/

Self, T. (2011). *The DITA style guide: Best practices for authors*. Research Triangle Park, NC: Scriptorium Press.

Shannon, C., & Weaver, W. (1949). *The mathematical theory of communication*. Urbana, IL: University of Illinois Press.

Swarts, J. (2010). Recycled writing: Assembling actor networks from reusable content. *Journal of Business and Technical Communication*, *24*(2), 127–163. doi: 10.1177/1050651909353307

Vazquez, J. (2009). *Practical DITA* (2nd ed.). Durham, NC: SDI Global Solutions.

White, L. (2013). *DITA for print: A DITA open toolkit workbook*. Laguna Hills, CA: XML Press.

Yagodich, R. (2014). *Author experience: Bridging the gap between people and technology in content management*. Laguna Hills, CA: XML Press.

About the Authors

Carlos Evia is an associate professor and director of Professional and Technical Writing at Virginia Tech. He conducts research about technology-enabled workplace communication for the Virginia Tech Centers for Human-Computer Interaction and Occupational Safety and Health. He is a member of the DITA Technical Committee, with emphasis on the Lightweight DITA subcommittee. He holds a PhD in Technical Communication and Rhetoric from Texas Tech University. He is available at carlos.evia@vt.edu.

Michael Priestley is IBM's Enterprise Content Technology Strategist. He works with teams across IBM to coordinate standards and technologies that can enable findable, usable, and reusable content across the enterprise content ecosystem. He was the co-editor of the DITA 1.0 and DITA 1.1 specifications, and is currently working on the specification for Lightweight DITA. He is available at mpriestl@ca.ibm.com.

Manuscript received 21 July 2015, revised 24 August 2015; accepted 10 October 2015.