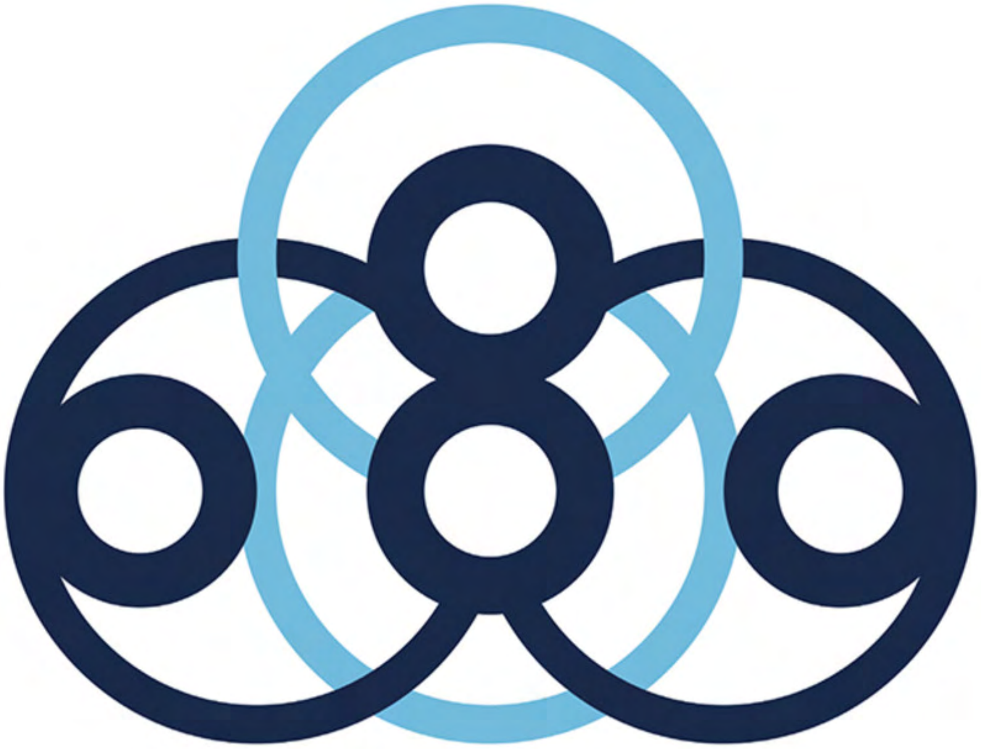


CREATING INTELLIGENT CONTENT WITH LIGHTWEIGHT DITA

Carlos Evia

ATTW
ROUTLEDGE

CREATING INTELLIGENT CONTENT WITH LIGHTWEIGHT DITA

Creating Intelligent Content with Lightweight DITA documents the evolution of the Darwin Information Typing Architecture (DITA) – a widely used open standard for structuring technical content. DITA has grown in popularity and features since its origins as an internal grammar for structuring technical documentation at IBM. This book introduces Lightweight DITA (LwDITA, which should be read as "Lightweight DITA") as a proposed version of the DITA standard that reduces its dependence on complex Extensible Markup Language (XML) structures and simplifies its authoring experience. This volume aims to reconcile discrepancies and similarities in methods for authoring content in industry and academia and does so by reporting on DITA's evolution through the lens of computational thinking, which has been connected in scholarship and media to initiatives for learning to code and programming.

Evia's core argument is that if technical communicators are trained with principles of rhetorical problem solving and computational thinking, they can create structured content in lightweight workflows with XML, HTML5, and Markdown designed to reduce the learning curve associated with DITA and similar authoring methodologies. At the same time, this book has the goal of making concepts of structured authoring and intelligent content easier to learn and teach in humanities-based writing and communication programs. This book is intended for practitioners and students interested in structured authoring or the DITA standard.

Carlos Evia is an associate professor in the Department of Communication at Virginia Tech. He teaches courses in professional communication and content management strategies. Carlos is also a voting member of the DITA technical committee and co-chair of the Lightweight DITA subcommittee at the Organization for the Advancement of Structured Information Standards.

CREATING INTELLIGENT CONTENT WITH LIGHTWEIGHT DITA

Carlos Evia

Routledge
Taylor & Francis Group
NEW YORK AND LONDON

First published 2019
by Routledge
711 Third Avenue, New York, NY 10017

and by Routledge
2 Park Square, Milton Park, Abingdon, Oxon, OX14 4RN

Routledge is an imprint of the Taylor & Francis Group, an informa business

Library of Congress Cataloging in Publication Data
Names: Evia, Carlos, 1971- author.
Title: Creating intelligent content with lightweight DITA / Carlos Evia.
Description: [New York, New York] : Routledge, [2019] | Series: ATTW book series in technical and professional communication | Includes bibliographical references.
Identifiers: LCCN 2018034997| ISBN 9780815393825 (hardback) | ISBN 9780815393818 (paperback) | ISBN 9781351187510 (ebook)
Subjects: LCSH: Darwin Information Typing Architecture. | Technical writing—Data processing. | Writing—Automation. | Information organization—Computer programs.
Classification: LCC QA76.76.H94 E97 2019 | DDC 025.00285—dc23
LC record available at https://lccn.loc.gov/2018034997

ISBN: 9780815393818 (hbk)
ISBN: 9780815393825 (pbk)
ISBN: 9781351187510 (ebk)

Typeset in Minion
by Swales & Willis Ltd, Exeter, Devon, UK

To Jane and Sofia for being everything.
A mi papá (el original Dr. Carlos Evia), a mi mamá (que no es doctora pero de todo sabe), y a mis hermanos.

CONTENTS

FIGURES

PREFACE

In the summer of 2004, I was on my way to Blacksburg, Virginia, to start my new job as assistant professor of technical communication at Virginia Tech. Fresh out of graduate school at Texas Tech University, I was still in that period in which I thought it was my job to read and absorb everything and anything that was published or presented about technical communication. My new office only had a laptop computer on an otherwise empty desk when I started reading some of the just-published proceedings from the 2004 conference (now called the Summit) of the Society for Technical Communication.

The proceedings included a presentation titled "The Future of Technical Communication According to Those Who Teach It," by David Dayton – a fellow graduate (a couple of years ahead of me) of the PhD in Technical Communication and Rhetoric at Texas Tech University. Under the title of "Plate Tectonics of T-COM," Dayton's presentation included a slide that probably changed my academic life. Dayton's slide identified "two major fault lines (that) create tension, release continuous discourse" in the field of technical communication:

- Between academics and practitioners
- Among academics: literary/discourse focused versus social-science/technology focused.

As a somewhat naïve new PhD graduate, I was unaware of those fault lines. Probably it was because I came to the academic side of technical communication from industry (I was a practicing technical writer and rather clumsy interface designer when I decided to get a PhD). Or maybe it was because I was coming from a social sciences and computing systems background in another country. In Mexico, English departments (if they exist) focus on English as a second language and technical writing does not have a strong (any?) presence in literature or letters departments.

I remember calling my father and telling him about Dayton's presentation. I told him I was unsure about my affiliation on those divisions. As a new college professor, did I need to side more with academics? As a new professor in an English department, did I need to learn more about literary approaches and discourse?

My father's advice now sounds simple, but at the time was eye-opening and somehow unexpected. Why not do both? Why not build bridges and connect the seemingly separate camps?

Fifteen years later, this book represents my contribution to patching those fault lines. *Creating Intelligent Content with Lightweight DITA* is the work of an academic who has tried to keep an active presence in industry. It also attempts to connect discourse and writing principles to technological approaches for managing content.

Getting here has been a long trip, and this book was, at different points, a very different work. Initially, it was going to be a history of the computer manual. In that version, the book claimed that the obsolescence of the manual as the main genre of technical communication was one of the causes of the professional fault lines identified in Dayton's presentation. Some of that work survived and is present in Chapter 2. At another point, the book was going to be a guide for incorporating principles of computational thinking in technical writing courses aimed at students of computer science. That work influenced Chapters 7 and 8.

Nevertheless, my experience with the Darwin Information Typing Architecture (DITA) originally as a user, then as a professor teaching DITA in my classes, and eventually as a committee member and spec author, took the book in its final direction. Both original ideas for this book (the one about the history of the manual and the one about computational thinking in technical writing courses) somehow ended up in the same place: they proposed Lightweight DITA (LwDITA) as a solution to more than one problem affecting technical communication and content development. LwDITA represents the evolution of the computer manual and it provides a way to introduce computational thinking in writing courses.

When Taylor & Francis/Routledge circulated the prospectus and sample chapters for this book to anonymous reviewers, their feedback was on point: the book should be about Lightweight DITA. Since I had been breathing LwDITA in my teaching and research work at Virginia Tech, and in my committee contributions to OASIS since 2014, that sounded like an excellent idea. At the same time, it presented a very risky challenge: I had to write a whole book about a proposed standard for structuring information that had not been approved and will not yet be approved by the time the book is published. Therefore, this is not the ultimate LwDITA user guide, and the proposed content components and syntax details included in this book might change as the standard goes through the rounds of committee and general public approval enforced by OASIS. The history behind LwDITA, however, will not change. The need for a process or lifecycle that relies on simple markup structures and values the work of human authors to produce intelligent content will not change either. The evolution of DITA in a path that incorporates feedback from users, practices in writing instruction and communication that represent the bulk of this book will stay the same even if the LwDITA authoring formats look slightly different when the standard is finally approved.

Written from the hypocenter of the fault line between academia and industry in technical communication, I hope this book reaches its target audiences in both camps.

Reference

Dayton, D. (2004). The future of technical communication according to those who teach it. *Paper presented at the conference of the Society for Technical Communication*. May 10, 2004.

ACKNOWLEDGMENTS

This book reflects the hard work of the Lightweight DITA subcommittee at OASIS. We have been in conference calls for many years designing, evaluating, and revising the recommendations included in this book. I want to thank all active members of the subcommittee, but I need to specifically acknowledge the contributions and support of Rahel Anne Bailie, Bill Burns, Stan Doherty, Mark Giffin, Tim Grantham, Rob Hanna, Scott Hudson, and Keith Schengili-Roberts. In the early days of the subcommittee, Don Day, John Hunt, and Amber Swope made invaluable contributions to shape the proposed standard.

Kris Eberlein and Robert Anderson are also members of the subcommittee, but their guidance, leadership, and friendship as DITA spec authors deserve a special mention.

LwDITA would only be a theoretical experiment without the work of adventurous and generous developers like Jarno Elovirta, George Bina, and Radu Coravu. Bernard Aschwanden's interest in the proposed standard should take it to the next level in adoption and promotion. Sarah O'Keefe provided challenging and relevant feedback. JoAnn Hackos and Dawn Stevens made (and still make) room for LwDITA-related sessions in events of the Center for Information-Development Management that have promoted the proposed standard beyond the work of the dedicated subcommittee at OASIS.

Thank you to James Mathewson, Susan Carpenter, John Carroll, Jenifer Schlotfeldt, and all the former and current IBMers who shared with me their experiences about the past, present, and future of DITA and LwDITA.

My mentor and friend Carolyn Rude read all prototypes of this book, from the history of the manual to the LwDITA reference guide. Her feedback and vision as a pioneer explorer and bridge-builder of the technical communication fault lines has inspired me for decades. Rebekka Andersen, Tim Lockridge, Kelly Pender (who actually knows a lot about LwDITA but probably won't admit it), and Russell Willerton helped finding sources, reading chapters, and keeping my disciplinary identity healthy. Allison Hutchison and Kelly Scarff brought their priceless graduate student eyes to the project. Allison's feedback, full of "I get it now!" moments, was a joy to read. Thank you to the dozens of undergraduate and graduate students in my courses who have been participating in my LwDITA experiments since 2014. Bernice Hausman and Joe Eska supported my DITA habit for many years.

Manuel Pérez-Quiñones, Ed Fox, Steve Sheetz, Patrick Fan, Chris Zobel, Deborah Tatar, and Steve Harrison have been at one point or another involved in my explorations for disrupting tradition and adapting principles of computational thinking to "traditionally non-computing" fields like writing and communication.

Tharon Howard believed in this project and encouraged me to write it in its current shape. "We have a good number of proposals in the series now," he said. "But nothing like yours."

Thank you to my brother Pedro Evia (the real Chef Pedro) and his business partner Eduardo Rukos for the *Sensei Sushi* materials.

Finally, if this book needed a co-author, that would be Michael Priestley – the original *ditaguy*. As the father of DITA and LwDITA, Michael has been an innovator and visionary in the world of content development. But above all things he has been a friendly collaborator, leader, and dad-joke generator. And if this book needed a substitute author, that would be my fact-checker, cheerleader, critic, and friend Alan Houser. An undisclosed percentage of this book's royalties might (or might not) go to a fund to buy a Vespa-themed Rolex for Alan.

Thank you to my wife and constant source of motivation Dr. Jane Robertson Evia. Thank you to Sofia Evia for being an awesome bebé. . . I mean, niña grande. Here's a link to Sofia's response to the question of "what is papá's book about?" https://youtu.be/MvPuw8XqOYs.

FOREWORD

Carlos Evia's book *Creating Intelligent Content with Lightweight DITA* is a most welcome addition to the ATTW Book Series in Technical and Professional Communication (TPC) because, as its title suggests, it is a book about building and maintaining content which is appropriate for users using an exciting new standard for Extensible Markup Language (XML) called "Lightweight DITA" or LwDITA. Like all the other books in the ATTW Book Series, *Creating Intelligent Content with Lightweight DITA* is solidly based on its author's comprehensive knowledge of the literature in the field and years of teaching TPC students to create content in DITA for industry clients. Balancing between his mastery of the academy and his practical industry experience as a voting member of the DITA technical committee and as co-chair of the Lightweight DITA subcommittee at the Organization for the Advancement of Structured Information Standards, Evia's book provides TPC students and practitioners with two valuable outcomes: 1) an introduction to a development model for creating digital content adapted for users, and 2) an accessible introduction to the new LwDITA standard.

Evia's work is firmly situated in what is starting to be called the shift from the "craftsman model" of Technical Communication to the "Component Content Management model." This shift has come about as a result of the ways that new technologies like XML DITA, single-source authoring environments, and Content Management Systems have changed the ways that technical writers deliver information to audiences (or what we would now call "users"). As Carlos explains in Chapter 2, the "craftsman" model made technical communicators responsible for the development and delivery of *entire* documents. Workplace practitioners wrote complete documentation sets, recommendation reports, marketing brochures, etc. And our curricula and textbooks are still based on the assumption that our students are being prepared as craft persons who will do this same sort of work in their careers. The Component Content Management paradigm shift—which Carlos describes in Chapter 2, which Tatiana Batova and Rebekka Andersen outline in their 2017 *IEEE Transactions* article on skills needed for Component Content Management, and which JoAnn Hackos describes in her 2015 ISO/IEC/IEEE 26531 standard on content management—has helped our field understand that in the digital era technical writers in the workplace don't "write" so much as they develop, delineate, and manage small content modules which they collect into information architectures. Today's technical communicators still have to write well, but now they also need a skill set which Carlos calls "computational thinking." They have to pull on their knowledge of content strategy, information architectures, and Rhetoric in order to produce usable documentation sets in digital environments.

The Component Content Management shift has tremendous implications for TPC pedagogies and curriculum development. As Filipp Sapienza observed in the *Journal of Technical Writing and Communication* back in 2002, it suggests that "technical communicators will probably face a day when all organizational documents are saved in XML format." As educators, we should prepare our students for that experience, but as Carlos observes in this book, the problem is that learning XML and its complicated DITA standard is really hard. Even academics familiar with XHTML and CSS can find DITA intimidating. But because the new Lightweight DITA standard Carlos and his colleagues have been developing is much less dependent on both XML and DITA, it's far more accessible to TPC students and practitioners new to coding. LwDITA is exciting because it enjoys the benefits of the semantic web and single-source authoring without the incredibly steep learning curve required for XML and DITA.

This book is an extremely timely and much needed introduction to the Component Content Management and computational thinking movement in TPC. As such, it's a pleasure to have it in the ATTW Book Series in Technical and Professional Communication.

Dr. Tharon W. Howard
Editor, ATTW Book Series in Technical
and Professional Communication
July 21, 2018

1

REVISITING THE FUTURE OF TECHNICAL COMMUNICATION

James Mathewson knows a thing or two about technical communication. In his role as Distinguished Technical Marketer at IBM, James, a graduate of the Master of Science in Scientific and Technical Communication from the University of Minnesota-Twin Cities, was looking for interns who could work in a structured authoring environment. James took to Twitter to express his frustration after coming back empty-handed from his search for interns.

> Yes. Writing reusable content is a rare skill. Would that they taught it in college. Essays from whole cloth is more the thing.
>
> (Mathewson, 2016)

When he talks about "reusable content," Mathewson is referring to the "practice of using content components in multiple information products" (Eberlein, 2016 p. 54). In many technical communication practices, "efficient content reuse does not involve copy-and-pasting; instead it uses *transclusion*, whereby content is authored in one location and used by reference in other locations" (Eberlein, 2016 p. 55). The "essays from whole cloth" reference is a nod to the book-oriented kind of writing using a word processor that most academic programs in technical communication and content development teach as standard practice.

It was not supposed to be like this.

In 2012, the Adobe Technical Communication Suite (TCS) team distributed on several social media channels a video titled "The Future of Technical Communication." 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 (Figure 1.1), the video's narrator describes that "for some, (the future of technical

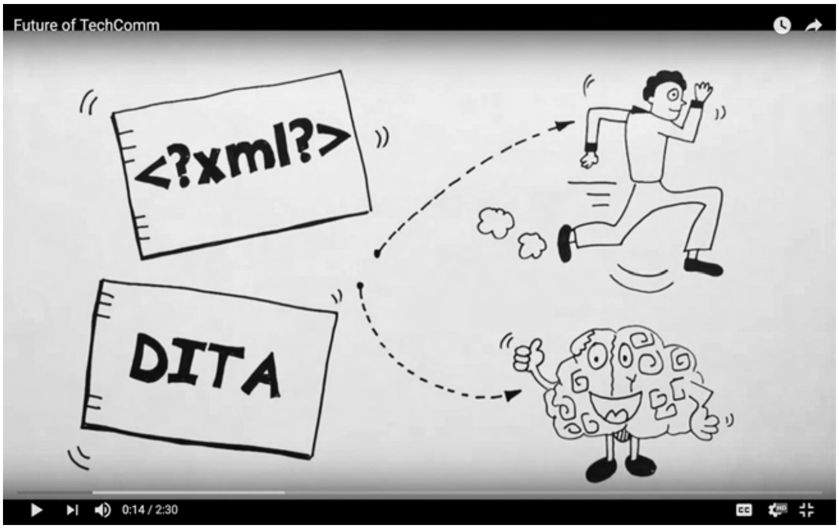


FIGURE 1.1 Screen capture from the video "The Future of Technical Communication," by Adobe Technical Communication Suite. This specific frame displays the future of our profession as "more and more structured content and the ability to work faster and smarter with XML and DITA constructs."

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" (AdobeTCS, 2012).

The "future" of technical communication described in the Adobe video focuses on what Rebekka Andersen describes as "structured content that is highly adaptable and portable and can be configured on the fly in response to specific user requests" (2014, p. 116). Andersen adds that this type of content supported by topic-based information design "has been given various names, including intelligent content, nimble content, smart content, portable content, and future-ready content." Rockley, Cooper, and Abel describe intelligent content as "designed to be modular, structured, reusable, format free, and semantically rich and, as a consequence, discoverable, reconfigurable, and adaptable" (Rockley et al., 2015, p. 1). In academia, scholars have praised Extensible Markup Language (XML) as the foundation for information reuse, single sourcing, and, particularly, content management for more than a decade. Publications from the 2000s heralded that "technical communicators will probably face a day when all organizational documents are saved in XML format" (Sapienza, 2002, p. 156) and argued that "technical communicators should be able to write, edit, and manage XML documentation, including XML tags, document type definitions (DTDs), XML schemas, and Darwin Information Typing Architecture (DITA)" (Gesteland McShane, 2009, p. 74).

The future of technical communication, as seen from industry and academia, was supposed to cover the type of structured writing for reuse that James Mathewson was looking for.

It is not as though the academic side of technical communication is stagnant. On the contrary, scholars in the field have expanded the work of technical communication into domains and scenarios like healthcare, policy-making, and social justice, among others, that were closed to writing researchers in the recent past. James's tweet started a discussion about the many topics covered in courses and programs in technical communication. Researchers work with practitioners that include community partners, government officials, and medics. However, "many of the pedagogical concerns of academic instructors in professional and technical communication have changed little over the past decade, even as practitioner discourse has continued to spin off in the direction of content strategy and similar areas" (Clark, 2016, p. 19). Academic discussion related to the implementation and innovation of intelligent content has been slow, as evidenced by the titles and abstracts of presentations and publications from 2012 to 2018 in relevant conferences and journals, respectively.

Traditionally, and as mentioned in the Adobe video, intelligent content workflows for technical communication rely on structure provided by Extensible Markup Language (XML) and some of its specific grammars like DocBook, S1000D, and DITA (Cowan, 2010; Glushko & McGrath, 2008; Hackos, 2011, among others). Although the benefits of a well-planned and implemented intelligent content workflow built on XML or DITA are undeniable, this type of solution presents significant challenges that have repercussions in the academic and workplace facets of the profession.

Technical communication consultants and practitioners frequently report on success stories and challenge narratives from implementing and managing XML-based content projects in conferences like DITA/Content Management Strategies and LavaCon. Furthermore, blogs and social media postings from content professionals create and maintain an active online community of discussion and recommendations. Because of proprietary practices, it would be impractical to poll private companies, government agencies, and non-profit institutions worldwide to tally a number of DITA and XML users. Nevertheless, some attempts at quantifying and documenting usage figures do exist. For example, the website "DITA Writer" maintains a list of companies using DITA based on reports from social media channels. As of summer 2018, the list included 724 companies, excluding consulting and training firms (DITAWriter, n.d.).

Adoption numbers and success stories should not hide that the evolution of intelligent content takes place on a slightly rocky path; even in practitioner circles, there is 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" (Wachter-Boettcher, 2012, 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:

> 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)

Those are valid concerns. DITA, as it evolves as an open standard, needs to address them and learn from its users. This chapter presents an overview of the evolution of DITA – an XML-based "end-to-end architecture for creating and delivering modular technical information" (Priestley et al., 2001, p. 354). The future of technical communication still involves XML; therefore, the following sections include brief introductions to Extensible Markup Language and the Darwin Information Typing Architecture, providing examples of the main content and collection types included in the latter. Then, the chapter explains the need for a simplified version of DITA that allows authors to contribute to intelligent content ecosystems writing in markup languages that do not require complex XML structures. Lastly, the chapter focuses on strategies for adopting a DITA workflow in three simplified authoring formats, which are connected to principles of computational thinking and emphasize the importance of human abstraction before machine automation.

An Introduction to XML in Technical Communication

Although an author does not even need to be familiar with XML in order to create information products in the workflows introduced in this book, the ideas presented in *Creating Intelligent Content with Lightweight DITA* are rooted in DITA and XML. As a result, I cannot leave out working definitions and background for context when introducing the standard and its evolution as a problem-solving methodology for technical communication. This section will be especially useful to readers who have not ventured into standards and markup languages for intelligent content.

First, I define XML for the specific context of *Creating Intelligent Content with Lightweight DITA*. Definitions and descriptions of the Extensible Markup Language abound in academic and practitioner publications. A search for "XML" on amazon.com reveals more than 1,100 results under the "Books: Computers & Technology: Programming Languages" category. In the realm of rhetoric studies, Applen and McDaniel wrote an important text about XML and its implications for rhetorical work. In one of the contrasting moves of their definition, they posit XML as different from Hypertext Markup Language (HTML) and other typesetting and formatting syntaxes because it enables the processes of "identifying, separating, and recombining" content for different purposes (Applen & McDaniel, 2009, p. 42). Content in XML, for example, can be tagged as a product name, a procedure, a works cited entry, or any other part of a technical or professional document. Once those parts or components are tagged, they can be "assembled" in different ways for use in a variety of deliverables and media. New documents need not be written for each purpose, and one single change can be reflected accurately across all the information materials produced by a given company.

Let me introduce you here to Pedro, a chef and restaurateur who is going to be the main user-author in this book's examples. Chef Pedro has a background in marketing and culinary arts. He is comfortable with technology and social media, but he has never worked with any type of computer code. He is looking for a way to standardize the recipes used by his staff in the several kitchens he manages. As a franchise owner, he wants to ensure that the food in his restaurants is consistent, and his kitchen staff needs up-to-date documentation with recipes and techniques adopted in all of his restaurants. At the same time, the menus in his restaurants need to reflect some of that content, which is also featured on a website, mobile app, and an electronic book he sells describing the history and process of his restaurants. Chef Pedro, like many content owners in this world, has been using Microsoft Word for most of his information products, and the publications department in his company takes some of those Word files and, through long sessions of copy and pasting, produces menus and flyers in Adobe InDesign and maintains the restaurant's website using WordPress. When a change needs to be reflected in those information products (e.g., the kitchen introduces a new special or the office's telephone number should be updated), Chef Pedro makes the changes in his original Word files, and then the publications team has to manually update the menus, flyers, and website that borrow content from the Word master source.

Someone mentioned structured authoring with XML and that caught Chef Pedro's attention. The promise: a centralized content repository structured in XML can be the source for many user deliverables. And the production and update of those deliverables can be automated. No more copy and paste! Now, if Pedro were to open a text editor in his computer and start typing XML tags to structure his content, he probably would not know where to go after typing for a few minutes. To create information deliverables ready for human consumption, XML files need to go through a process of publishing that, according to O'Keefe, can involve the following roles, which "in a small group, one person may hold any or all" (2009, p. 14):

- Document architect, who defines and implements document structure
- Template designer, who establishes the look and feel of content deliverables
- Writer, who creates content
- Technical editors, who can focus on word choice, grammar, and overall organization
- Production editors, who will get involved in defining the transformation files that assign formatting based on structure.

For someone like Chef Pedro (and even for many of my technical writing students), creating the tags to structure content in XML can be an easy-to-learn task. On the other hand, producing information deliverables based on that structure (say, a webpage with a specific recipe) is not necessarily simple when implementing a custom XML content type. That's when a standard like DITA can help. Pedro could buy a software package to structure his recipes and create an online cookbook; however, in a previous stage of his career he was involved in a marketing project that depended on a commercial application. When the developer stopped updating the application and it became obsolete, Pedro's content was locked and required an expensive and time-consuming process of conversion to be rescued in another program. A chef might not be familiar with scholarship in writing studies, but authors like Karl Stolley, in his influential "The Lo-Fi Manifesto," have argued for the adoption of open standards over software packages, identifying it as the only way to ensure that "digital works should long outlast the software that played a role in their creation" (Stolley, 2008).

This kind of data tagging and manipulation that XML allows is at the heart of intelligent content. From the practitioner side of technical communication, O'Keefe points out that XML "defines a standard for storing structured content in text files" (2009, p. 15). O'Keefe's extended definition includes the following features of XML:

- It is a markup language, which means that content is enclosed by tags.
- Its element tags are enclosed in angle brackets (<element>This is element text</element>).
- It does not provide a set of predefined tags. Instead, authors define their own tags and their relationships. (O'Keefe, 2009, p. 15)

Additionally, O'Keefe provides a sample XML file that presents a recipe for making marinara sauce. (O'Keefe, 2009, p. 16) (Figure 1.2).

This process of tagging, commenting, and nesting data will look familiar to readers who have used HTML or another markup language. O'Keefe's recipe[1] features a main container element (<**Recipe**>), which includes several sub-elements (<**Name**>, <**IngredientList**>, and <**Instructions**>). Some of these sub-elements

```
<Recipe Cuisine="Italian" Author="Unknown">
    <Name>Marinara Sauce</Name>
    <IngredientList>
        <Ingredient>
            <Quantity>2 tbsp.</Quantity>
            <Item>olive oil</Item>
        </Ingredient>
        <Ingredient>
            <Quantity>2 cloves</Quantity>
            <Item>garlic</Item>
            <Preparation>minced</Preparation>
        </Ingredient>
        <Ingredient>
            <Quantity>1/2 tsp.</Quantity>
            <Item>hot red pepper</Item>
        </Ingredient>
        <Ingredient>
            <Quantity>28 oz.</Quantity>
            <Item>canned tomatoes, preferably San Marzano</Item>
        </Ingredient>
        <Ingredient>
            <Quantity>2 tbsp.</Quantity>
            <Item>parsley</Item>
            <Preparation>chopped</Preparation>
        </Ingredient>
    </IngredientList>
    <Instructions>
        <Para>Heat olive oil in a large saucepan on medium. Add garlic and hot
  red pepper and sweat until fragrant. Add tomatoes, breaking up into smaller
 pieces. Simmer on medium-
low heat for at least 20 minutes. Add parsley, simmer for another five minutes.
   Serve over long pasta.</Para>
    </Instructions>
</Recipe>
```

FIGURE 1.2 Sarah O'Keefe's sample recipe for marinara sauce in XML. The structured recipe keeps content organized and should help computers understand and manipulate elements and attributes. Human readers, however, still need to wait for a publishing process that will produce content deliverables that do not look like computer code.

hold actual content (e.g., **<Name>**), and others nest additional levels of elements and, eventually, content (e.g. the "parsley" **<Item>** inside **<Ingredient>** inside **<IngredientList>**). XML also uses attributes to present additional information about data and content (see the **@Cuisine** and **@Author** attributes in the main **<Recipe>** element). If you're unfamiliar with XML structures, don't worry; the relevant XML elements for this book will be discussed in detail in subsequent chapters. For now, just focus on the fact that XML uses tags to identify the types of rhetorical "moves" that the content on the page represents.

Just like Chef Pedro, who wants to create a structured template for all entries in his recipe guide, information developers at IBM started looking at XML in the late 1990s to organize technical content. One of the solutions that came out of those explorations at IBM was the Darwin Information Typing Architecture.

Much Ado About DITA

In an interview for the website "DITAWriter," Don Day, one of the original developers of the DITA standard, 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.
>
> (Day, quoted in DitaWriter, 2016)

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 and product support portals on the Web. (Day et al., 2001). 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 (Day, quoted in DITAWriter, 2016).

IBM eventually donated DITA as an open standard, which is currently maintained by the non-profit consortium OASIS. DITA, however, "has evolved substantially since that initial donation to encompass a very wide scope of requirements indeed" (Kimber, 2012, 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" (OASIS Darwin Information Typing Architecture TC, n.d.). Hackos summarizes the key benefits of DITA for technical communicators as follows:

- A fully tested DTD or schema for XML-based authoring
- A community of developers investing in improvements to the DITA model
- An open source toolkit you can use to produce your own output in multiple media without having to invest in proprietary tools
- A thoroughly developed approach to information development originating with OASIS and now encompassing many other companies, large and small, that find value in a standards-based approach (2011, p. 9).

Additionally, from a perspective addressing the needs of managers and supervisors, Hackos presents a list of DITA's business advantages, which suggest that the standard 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 (Hackos, 2011, p. 10).

For Chef Pedro, adopting a standard like DITA simplifies the process of structuring content and producing information deliverables for human users. Instead of designing custom tags and depending on a publishing team to design templates and validation tools, he can use DITA's content types and take advantage of the benefits listed by Hackos. For an author, the main benefit from Hackos's list is in the "fully tested DTD or schema for XML-based authoring," which I discuss in the next section.

DITA Content Types: More than Templates

DITA's "fully tested DTD or schema for XML-based authoring" comes from the *Information Typing* part of its name. Content in DITA is presented as units or individual XML files that conform to pre-established types or models. Those pre-established content types are enforced by files that are commonly referred to as DTDs, although DTD is only one of the markup languages used to verify the structure of those files. XML, and as a consequence DITA, files can also be validated by XML Schema and RELAX NG (Regular Language for XML Next Generation) files.

If that validation process sounds a little too complicated, as an author all you need to know is that the content types enforced by the DITA standard create information topics. A topic is "a self-contained unit of information. An effective topic covers only one subject. Each topic is long enough to make sense on its own, but short enough to stick to one point without expanding into other subjects" (Bellamy et al. 2012, p. 8), and can be defined as "small independent piece of information on a single subject" (Baker, 2013, p. 71). Topic-based writing is described as "authoring an information set as a collection of discrete units called *topics*, rather than as a whole book or help system" (Baker, 2016 p. 52). This kind of writing is the basis of several technical communication and intelligent content practices and techniques, including component content management systems (CCMS), which are "a centralized system that helps organizations capture, manage, store, preserve, and deliver topic-based content (components)" (Kerzreho, 2016 p. 60), and single sourcing, which can be defined as the practice of "creating content once, planning for its reuse in multiple places, contexts, and output channels" (White, 2016 p. 56).

While authors certainly can work on a topic-based environment without DITA (see Baker, 2013), the term *topic* is frequently associated with this open standard, as explained in the following quote from Andersen & Batova:

> Content components are the building blocks of information products. While terms such as granules, modules, and units are commonly used to describe these blocks, the term topic has gained the most traction in the past few years, particularly in the trade literature. Topic derives from the widely adopted open content standard known as Darwin Information Typing Architecture (DITA), which defines a common structure for content that promotes the consistent creation, sharing, and reuse of content.
> (Andersen & Batova, 2015, p. 255)

The literature focuses on three topic types that "represent the vast majority of content produced to support users of technical information" (Hackos, 2011, 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 (Pringle & O'Keefe, 2009, p. 235).

For authors of technical content, 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 the sauces section in Chef Pedro's recipe book, while tasks can provide recipes for specific salsas and condiments (he can write a step about dicing tomatoes once and reuse in all the recipes that need it!), and a reference topic can list common techniques and tools for preparing ingredients.

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 (as we saw in the marinara sauce example) does not provide a defined set of tags, but DITA does offer a catalog of elements and attributes relevant for technical communicators. Although in Chapters 5 and 6 I will further analyze the DITA tags and attributes, for the scope of this introductory section, I structured and tagged O'Keefe's recipe for marinara sauce as a DITA task (Figure 1.3). You can download these code samples from the *Creating Intelligent Content with Lightweight DITA* GitHub repository (https://github.com/carlosevia/lwdita-book).

The recipe has a strong structure with visible sections. The **<task>** element opens the topic announcing "this is a task," and it contains elements like the following:

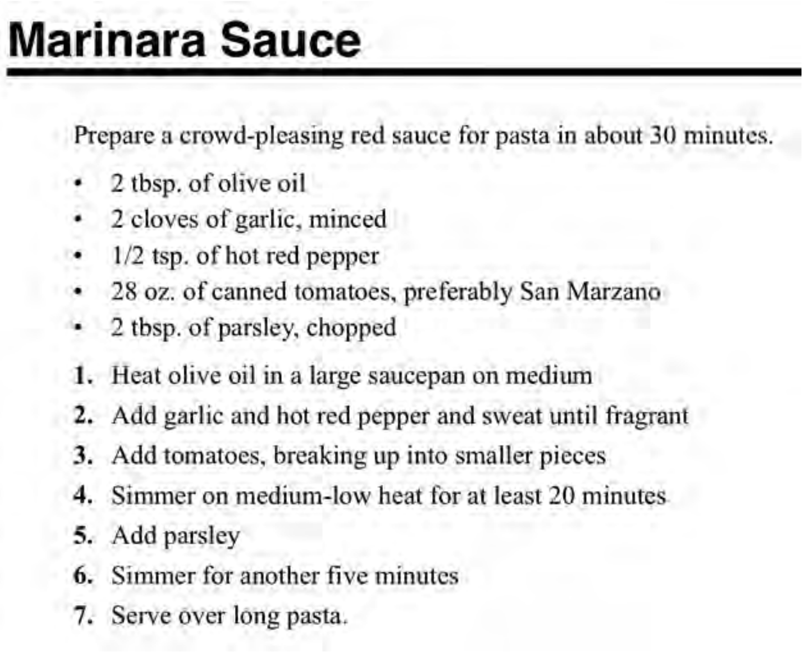
- **<shortdesc>** with a summary of the topic's contents
- **<prolog>** with some information about the recipe's author and category
- **<prereq>** with a list of the ingredients needed for the recipe
- **<steps>** with a collection of individual **<step>** elements containing a command (**<cmd>**) with an action verb.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE task PUBLIC "-//OASIS//DTD DITA Task//EN" "task.dtd">
<task id="t-marinara">
    <title>Marinara Sauce</title>
    <shortdesc>Prepare a crowd-
pleasing red sauce for pasta in about 30 minutes.</shortdesc>
    <prolog>
        <author>Unknown</author>
        <metadata>
            <category>Italian</category>
        </metadata>
    </prolog>
    <taskbody>
        <prereq>
            <ul>
                <li>2 tbsp. of olive oil</li>
                <li>2 cloves of garlic, minced</li>
                <li>1/2 tsp. of hot red pepper</li>
                <li>28 oz. of canned tomatoes, preferably San Marzano</li>
                <li>2 tbsp. of parsley, chopped</li>
            </ul>
        </prereq>
        <steps>
            <step>
                <cmd>Heat olive oil in a large saucepan on medium</cmd>
            </step>
            <step>
                <cmd>Add garlic and hot red pepper and sweat until fragrant
</cmd>
            </step>
            <step>
                <cmd>Add tomatoes, breaking up into smaller pieces</cmd>
            </step>
            <step>
                <cmd>Simmer on medium-low heat for at least 20 minutes</cmd>
            </step>
            <step>
                <cmd>Add parsley</cmd>
            </step>
            <step>
                <cmd>Simmer for another five minutes</cmd>
            </step>
            <step>
                <cmd>Serve over long pasta.</cmd>
            </step>
        </steps>
    </taskbody>
    </task>
```

FIGURE 1.3 Marinara sauce recipe as a DITA task. The standard's "fully tested DTD or schema for XML-based authoring" includes commonly used elements or moves in technical publication. Thus, the recipe collector does not have to invent custom tags.

Hackos mentioned "an open source toolkit you can use to produce your own output in multiple media without having to invest in proprietary tools" (2011, p. 9). Using that toolkit, known as the DITA Open Toolkit (OT), or a software tool that uses the Open Toolkit, Chef Pedro can produce quick information deliverables. If a sous-chef needs a printable PDF version of the recipe, Pedro can produce it without hiring a template designer (Figure 1.4). And if a different member of the kitchen team requires a web version of the recipe, the DITA-OT also provides that option and allows the author to link to a Cascading Style Sheet (CSS) file for formatting and design (Figure 1.5). In this introductory chapter I will only present the results of transformations using the DITA-OT, but in Chapter 5 I will focus on specific steps for conducting those transformations.

Concept, task, and reference are, for many authors and their managers, essential to DITA. Yet, the big-picture ideas of topic-based information and component content management go beyond the actual topic types enforced by the DITA standard and DITA-aware tools. Although concept, task, and reference are still defined types in the DITA standard, the official specification for DITA 1.3 also includes topic types for troubleshooting, which "provides markup for corrective action information such as troubleshooting and alarm clearing" (2.7.1.6 Troubleshooting topic, 2016) and glossary, which "defines a single sense of one term" (2.7.1.7 Glossary entry topic, 2016). Actually, the all-inclusive edition of the DITA 1.3 standard has 26 document types (predefined document, or even genre, templates) and 621 element types (placeholders or structures for specific content moves). Even in its base edition, DITA 1.3 has 4 document types and 189 element types. And authors should not underestimate



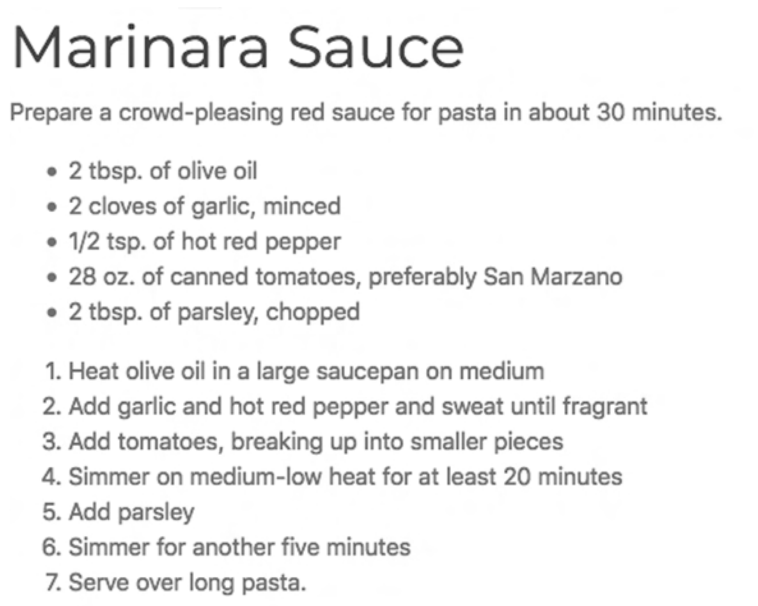
Marinara Sauce

Prepare a crowd-pleasing red sauce for pasta in about 30 minutes.

- 2 tbsp. of olive oil
- 2 cloves of garlic, minced
- 1/2 tsp. of hot red pepper
- 28 oz. of canned tomatoes, preferably San Marzano
- 2 tbsp. of parsley, chopped

1. Heat olive oil in a large saucepan on medium
2. Add garlic and hot red pepper and sweat until fragrant
3. Add tomatoes, breaking up into smaller pieces
4. Simmer on medium-low heat for at least 20 minutes
5. Add parsley
6. Simmer for another five minutes
7. Serve over long pasta.

FIGURE 1.4 Marinara sauce DITA task transformed to a PDF deliverable. This simple output did not require a dedicated stylesheet or choices about pagination and typography.



Marinara Sauce

Prepare a crowd-pleasing red sauce for pasta in about 30 minutes.

- 2 tbsp. of olive oil
- 2 cloves of garlic, minced
- 1/2 tsp. of hot red pepper
- 28 oz. of canned tomatoes, preferably San Marzano
- 2 tbsp. of parsley, chopped

1. Heat olive oil in a large saucepan on medium
2. Add garlic and hot red pepper and sweat until fragrant
3. Add tomatoes, breaking up into smaller pieces
4. Simmer on medium-low heat for at least 20 minutes
5. Add parsley
6. Simmer for another five minutes
7. Serve over long pasta.

FIGURE 1.5 Marinara sauce DITA task transformed to an HTML5 deliverable. This sample output includes a link to an external CSS file for formatting.

the importance of a generic topic, defined as "the basic unit of authoring and reuse" (2.2.1.1 The topic as the basic unit of information, 2016), which is the only information type included in the base edition of the DITA standard. The generic topic, with its simple tags for paragraphs and lists, can mark up a marketing blog post. A basic topic can also include a flexible procedure, or even provide pointers for a web-based product tour with some JavaScript processing, among many other applications. Additionally, the versatile task topic in DITA 1.3 can take the shape of a general task topic, a strict task, or a machinery task topic, depending on the context and purpose of usage. DITA 1.3 also includes topic types designed for learning and training projects: learning plan, learning overview, learning content, learning summary, and learning assessment.

If those pre-established topic types were not enough for a particular writer and context, DITA topics can be *specialized* to create information types unique to any intelligent content domain. At the topic level, and without getting too deep into the process of DITA specialization, just as <**concept**> and <**task**> are specialized from the original generic <**topic**>, <**task**> could specialize into <**recipe**>. For example, Chef Pedro is going to need to provide ingredients for his recipes, so he could create an ingredients specialization to include this specific element (i.e., <**ingredients**> as a specialization of <**prereq**> for pre-requisites). This exercise in markup flexibility is a direct application of both the extensible part of XML and the Darwin element in DITA: XML elements can be extended and DITA information types can evolve to accommodate diverse content and processing needs.

DITA Maps

Properly tagged DITA topics can create or join content collections ready for reuse of material, single-sourcing of whole topics or their elements, and filters as determined by a deliverable's audience or context. The process of assembling topic collections depends on DITA maps, which are defined as "the glue that binds your topics together, the driver for producing your output, and the information path for your users to follow" (Bellamy et al., 2012, p. 91).

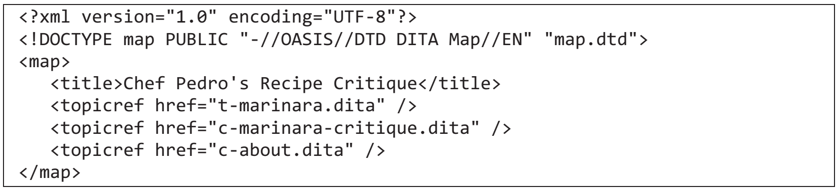
In their textbook *DITA Best Practices*, Bellamy et al. recommend using DITA maps to create an information set that specify which topics should be included in a user deliverable produced from the map, define an information architecture with the navigation for a set of topics, and create relationships between topics (Bellamy et al., 2012).

The following scenario combines the examples I have presented so far: Chef Pedro has a website in which he critiques recipes. For this week's entry, he will critique the marinara sauce recipe from Figure 1.3. The web deliverable that Chef Pedro needs for this project should include the following sections:

- The marinara sauce recipe (structured as a DITA task)
- Chef Pedro's critique of the recipe (structured as a DITA concept)
- Chef Pedro's "about" page, with biographical and professional information (structured as a DITA concept).

Figure 1.6 shows a very basic DITA map for Chef Pedro's recipe critique website.

The DITA map has a **<title>** element for the whole project. Then, the map includes links (**<topicref>**) for the individual files needed for the website. Using the DITA-OT, which I will cover in more detail in Chapters 5 and 6, I transformed



```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE map PUBLIC "-//OASIS//DTD DITA Map//EN" "map.dtd">
<map>
   <title>Chef Pedro's Recipe Critique</title>
   <topicref href="t-marinara.dita" />
   <topicref href="c-marinara-critique.dita" />
   <topicref href="c-about.dita" />
</map>
```

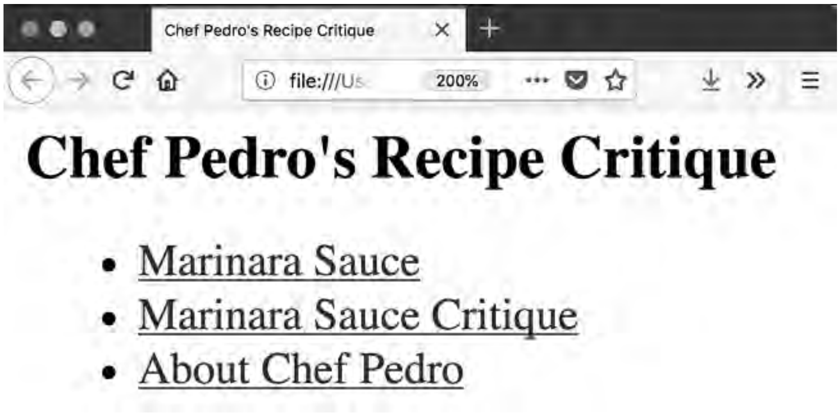
FIGURE 1.6 DITA map for Chef Pedro's recipe critique website. The map includes links (or topic references) to the marinara sauce recipe, Chef Pedro's critique of the recipe, and an "about" page with information about the author.

the map to a web deliverable. The DITA-OT automatically generated a navigation menu (figure 1.7) and took care of basic layout.

These features and benefits contribute to DITA's popularity and adaptability as an intelligent content solution for technical communication. Without a doubt, these features and benefits can also be intimidating for an author new to DITA. Although no author is expected to become an expert on all topic types and their relations, criticism in industry and, perhaps, the timid interest from academia stem from the long, and getting longer, list of types, tags, and their functions included in the DITA standard.

Let's go back to Chef Pedro, who is attempting now to structure some of his own recipes as tasks following specifications from the DITA 1.3 standard. The process sounds simple, and the content components in the DITA *task* document type seem just right for structuring a recipe. There can be some instances of the **<step>** element with a **<cmd>** for a command in each step, but things get quite complicated when Pedro looks at all the available elements in a task topic[2], which include the following: **<taskbody>**, **<prereq>**, **<context>**, **<steps>**, **<steps-informal>**, **<steps-unordered>**, **<step>**, **<stepsection>**, **<cmd>**, **<info>**, **<substeps>**, **<substep>**, **<stepxmp>**, **<choicetable>**, **<chhead>**, **<choptionhd>**, **<chdeschd>**, **<chrow>**, **<choption>**, **<chdesc>**, **<choices>**, **<steptroubleshooting>**, **<stepresult>**, **<tutorialinfo>**, **<tasktroubleshooting>**, **<result>**, and **<postreq>**. And that's just the task document type! Keep in mind that DITA 1.3 has 26 document types.

The next section introduces a DITA-based approach to intelligent content that has the potential of minimizing the standard's learning curve and promoting its adoption in small and medium scale information authoring and processing environments as well as classrooms.



Chef Pedro's Recipe Critique

- Marinara Sauce
- Marinara Sauce Critique
- About Chef Pedro

FIGURE 1.7 Navigation menu for a web transformation of Chef Pedro's recipe critique project. The DITA-OT generated the menu and took care of basic layout for this web deliverable.

DITA, Why Go Lightweight?

Michael Priestley, Senior Technical Staff Member and Enterprise Content Technology Strategist at IBM and known as one of DITA's "founding fathers" (Etheridge, 2016), defines Lightweight DITA (LwDITA, which should be read as "Lightweight DITA") as a simplified schema for structuring content, with fewer elements, tighter content models and a simplified specialization architecture to define new types compared to those of DITA XML. According to Priestley, "if there are three ways of doing things with full DITA, there will be only one way to do it with Lightweight DITA."

> That simplification makes it possible to implement DITA without XML – for example using Markdown, or HTML5. This brings the advantages of structured authoring to where people are already creating content, rather than trying to get every author onto one content platform. Lightweight DITA can be the glue that ties together many different authoring platforms across a company. It can also be an on-boarding ramp for full DITA. Lightweight DITA is particularly attractive to companies who need a faster ROI [return on investment] and an easier learning curve.
>
> (Priestley, quoted in Etheridge, 2016)

Creating Intelligent Content with Lightweight DITA introduces and analyzes LwDITA as an approach for developing intelligent content. It aims to address concerns and doubts about the adoption and evolution of DITA as a major standard for technical publications. Those concerns and doubts revolve around a major point of discrepancy in the world of technical communication: in industry, XML and DITA are widely used and some critics even see them as outdated and complex; in academia, "faculty are simply ignoring the subject, even though it has played a central role in the practitioner literature" (Clark, 2016, p. 19). In conversations at academic conferences and in social media exchanges, colleagues from other universities tell me they would like to try DITA in the future and make plans to contact me when they are working on their technical communication syllabus. I rarely get a second call and the conversation stays in the "someday/maybe" list. If XML and DITA are still seen as new tools by academics when some practitioners are already labeling them as arcane, then we are just writing a new chapter in the documented gap between research and practice in the field (Andersen, 2013; Rude, 2015; Lauren & Pigg, 2016, and others).

For an audience of curious but hesitant academics, this book demystifies the process of authoring and processing intelligent content. *Creating Intelligent Content with Lightweight DITA* shows structured authoring in practice without the need for sophisticated software tools. It also connects the rhetorical structures that drive the evolution of DITA as a standard for technical information to foundational concepts from the field's academic background and to common structures in computing education. For interested practitioners, this book presents the logical and computational essence of changes and improvements on a major documentation standard. For any reader, *Creating Intelligent Content with Lightweight DITA* places human authors at the center of intelligent content workflows, proposing that adopting LwDITA as a thought process and authoring schema will provide writers with some of the benefits associated with other information standards (DITA included) while **emphasizing thinking and abstraction over automation and machine processing**. This book is not about a specific tool or software platform: as an open standard in development, LwDITA does not depend on a vendor or "app." Hackos explains the importance of information standards for technical communicators with the following scenario, which also applies to LwDITA:

> What that means to information developers is that you can author a DocBook or DITA topic in one tool and open the topic in another tool without the loss of information or invalid markup. If the tool developers respect the standard, they allow for interoperability among tools.
>
> (Hackos, 2016, p. 29)

Based on the LwDITA proposed standard, the principles and strategies included in this book can be effective in a variety of authoring environments. I am aware, though, that some large content repositories with diverse user groups do need a robust environment built on DITA XML, and that strong component of the standard is not going away. I am a voting member of the DITA Technical Committee with the Organization for the Adoption of Structured Information Standards (OASIS), which is currently working on version 2.0 of the standard; DITA's present and future are necessary for the lightweight approaches discussed in this book. Therefore, this is unequivocally a book about DITA, but it is not a detailed introduction and how-to to the widely adopted XML-based DITA standard (for comprehensive introductory texts to DITA XML, see Hackos, 2011; Bellamy et al., 2012).

Creating Intelligent Content with Lightweight DITA does not have a reader prerequisite of experience with the DITA standard. However, I do assume that readers from industry and academia will have a technological curiosity about content development in workflows that go beyond a word processor or a "What You See Is What You Get" (WYSIWYG) web editor. Some experience with HTML or Markdown would also help, but the book provides enough context and information about those languages to understand and use the LwDITA proposed standard.

Computational Thinking and the Evolution of DITA

The theoretical axis of the evolution of DITA analyzed in this book is based on the concept of computational thinking, which is defined by their main proponents as follows:

> Computational thinking is the thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent.
>
> (Cuny et al. 2010, quoted by Wing, 2011)

Computational thinking (CT) has been connected in scholarship and media to initiatives for learning to code and programming. In her seminal essay on this topic, Wing explains that computational thinking takes an approach to "solving problems, designing systems and understanding human behavior by drawing on concepts fundamental to computer science" (Wing, 2006, p. 33), and that it is more about conceptualizing than programming. She adds that "thinking like a computer scientist means more than being able to program a computer. It requires thinking at multiple levels of abstraction" (2006, p. 35). She succinctly defines abstraction as "the essence of computational thinking" (2008, p. 3717).

In the technical communication literature, the concept of abstraction has been linked to Johndan Johnson-Eilola's work to establish "common ground between academic and corporate models" of the profession. Johnson-Eilola's skills for rearticulating technical communication included abstraction, which "requires students not merely to memorize information but also to learn to discern patterns, relationships, and hierarchies in large masses of information" (Johnson-Eilola, 1996, p. 260). Johnson-Eilola adds that a "paradigmatic example of this skill can be found in one of the most common tasks in software documentation: rethinking a series of system commands so that it coincides with a user's task representation and context" (1996, p. 260). More than two decades after Johnson-Eilola's model was published, abstraction continues to be a desired skill in the training and work of technical communicators. Furthermore, his example is still relevant when thinking about LwDITA and computational thinking: an author needs to separate the layers of abstraction when creating a task topic for a specific audience and context. The tools might be different, but the principles are the same.

Abstraction allows authors of intelligent content to separate the "layers" of a particular problem, and work on each one individually without concern for the others. Then, as the layers are recombined, they work together to solve the problem at hand, much like an algorithm. In the case of a computational solution for intelligent content, abstraction is a combination of two elements: information representation and separation of concerns (layers of abstraction). Computing is concerned with automating these abstractions. In order for that automation to be successful, computational thinking requires understanding not only of the concepts that each layer of abstraction represents, but also of the relationships between the multiple layers behind a specific problem.

Some computing professionals will argue that the abstractions in computational thinking are mainly related to algorithms. Some technical communication practitioners share that opinion (Baker, 2016). In Wing's model, however, computing abstractions go beyond numerical abstractions and cover symbolic, algorithmic, and representational abstractions (Wing, 2008). Others will emphasize the role of automation even in Wing's definition of computational thinking. *Creating Intelligent Content with Lightweight DITA* does not have the purpose of minimizing the importance of computing automations, and by all means technical communicators should learn computing programming languages and workflows if given the chance. This book treats ambitious promises of computational thinking with caution, as "there is little evidence to believe that students are learning higher-order thinking skills by learning programming" (Guzdial, 2016, p. 50). Nonetheless, thinking in and planning abstractions are really the everyday tasks of technical authors in an intelligent content environment. Wing (2008) describes computing as a combination of "mental" tools (abstractions) and "metal" tools (automation). Authors are in charge of the abstractions behind the code and content that provides the backbone for intelligent content. Automation is then provided by software applications, be it a commercial product like Adobe FrameMaker or an open source implementation of DITA. Authors do not need special software tools to create content in DITA; however, they will need a software processor to transform DITA topics into deliverables for human users. I will describe those processing applications later in this chapter (and throughout the book). Authors can participate in automation by writing a script or application to filter content. However, that step is not necessary for applying principles of computational thinking in technical communication from an author's perspective. I have seen colleagues in technical communication who value computational thinking and literacy as core skills of a college education and, as a result, send their students to take an introductory class in programming in a Computer Science department. Before (or in addition to) taking, say, a Python course out of their disciplinary context, why not send technical communication students to a course covering the *mental* and *metal* tools that practitioners in their field value?

A core argument in this book is based on Wing's definition of computational thinking, previous research about abstractions in technical communication, the work I have conducted as co-chair of the Lightweight DITA subcommittee at OASIS, and my experience teaching DITA at the college level since 2006 and LwDITA since 2015. I argue that **if technical communicators are trained with principles of rhetorical problem solving and computational thinking, they can work in lightweight environments without the need of a robust XML solution**. This type of training will primarily enhance an author's understanding of the layers of abstraction and common argumentation moves behind an intelligent content solution. These layers of abstraction are not necessarily new to the field of technical communication (they include long-discussed concepts and principles like separating content from design, and planning for single-sourcing and content reuse, among others), but are essential building blocks of intelligent content solutions that are still not fully embraced in academia. Before LwDITA existed, I presented a preliminary list of those abstractions as they applied to my teaching of DITA XML (Evia et al., 2015). Chapters 7 and 8 present an analysis of the abstractions behind the computational thinking and rhetorical problem solving processes that generate intelligent content with LwDITA, but the next section focuses on the potential benefits of LwDITA for content developers who do not need the features of full DITA XML.

Structuring Intelligent Content with LwDITA

LwDITA is a topic-based architecture for tagging and structuring intelligent content using flexible markup options. 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[3] format with a subset of XDITA elements that can be used for maximizing input readability while maintaining structure in content.

You do not need to use all three "flavors" at the same time to adopt LwDITA. You can work in HDITA all the time and you would still be using LwDITA. You can live in an MDITA environment without XML or HTML tags and you 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, Pedro can hire a technical writer to create recipe topics in XDITA (based on XML) while a marketing professional writes a description of the cookbook's features in HDITA (based on HTML5), and an engineer uses MDITA (based on Markdown) to create a reference for a specific command from the kitchen's laboratory. All their topics are treated as DITA and can take advantage of the standard's reuse, filtering, and single-sourcing capabilities.

All code examples in this book will focus on the open standards for DITA and LwDITA, and automation-related discussions will be based on features and affordances from the open source DITA-OT and "raw" XML, HTML5, and Markdown code. Professional authoring tools can hide code in What You See

Is What You Get (WYSIWYG) options, but the rhetorical, pedagogical, and computational principles of *Creating Intelligent Content with Lightweight DITA* view and perceive LwDITA topics as code.

The idea of simplified DITA code that would reduce its learning curve has been a topic of discussion on the standard's technical committee with OASIS for a few years. In 2011, the technical committee planned to release a "limited DITA profile," which was still XML-based, but depended heavily on HTML tags (such as <**p**> for paragraph and <**li**> for list item) to simplify many semantic structures of DITA XML. As the concept of Lightweight DITA developed further, at one point it became an XML subset of DITA that included, for example, 27 possible elements inside a topic, whereas DITA XML 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 LwDITA is not to replace DITA XML. Instead, LwDITA provides basic access to authors who do not need all the DITA standard's features but whose deliverables should be compatible with DITA XML.

Michael Priestley, from IBM, created the OASIS Lightweight DITA subcommittee in 2014 with the purpose of releasing LwDITA as an open standard related to but independent from DITA XML. I co-chair the Lightweight DITA subcommittee with Priestley, and I have published, alone and with Priestley, about the development of LwDITA (e.g., Evia & Priestley, 2016; Evia, 2017). While LwDITA is under development and not an approved OASIS standard at the time of this writing, feedback on our talks and papers about the new standard provides support for its development based on positive reactions and interest from partners in industry and academia. As lead editor of the LwDITA technical specification, I have attempted to combine the needs and resources of academic and industry content professionals, testing and implementing applied computational principles built on common concepts of genre and rhetorical theory.

As of this writing, LwDITA is a work in progress. This book reflects the structure of this proposed standard as it was presented in its initial introductory committee note (Evia et al., 2018). LwDITA details might change between the publication of this book and the actual release of the Lightweight DITA standard.

For a quick example of LwDITA in action (and I will analyze *more thoroughly* its formats later in the book), I coded O'Keefe's recipe for marinara sauce in LwDITA's different authoring formats. Figure 1.8 shows the recipe authored in XDITA – the LwDITA authoring format based on a simplified version of DITA XML.

The first major change is the topic type. In DITA XML (see Figure 1.3), the recipe was structured as a task and had predetermined elements that the DITA standard associates with a task (like <**steps**> and <**prereq**>**).** The simplified authoring experience of LwDITA, however, is based on a single topic type with elements common to most information units, including the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE topic PUBLIC "-//OASIS//DTD LIGHTWEIGHT DITA Topic//EN"
"lw-topic.dtd">
<topic id="t-marinara">
    <title>Marinara sauce</title>
    <shortdesc>Prepare a crowd-
pleasing red sauce for pasta in about 30 minutes.</shortdesc>
    <prolog>
        <data name="author" value="Unknown"/>
        <data name="category" value="Italian"/>
    </prolog>
    <body>
        <section>
            <title>Ingredients</title>
            <ul>
                <li>
                    <p>2 tbsp. of olive oil</p>
                </li>
                <li>
                    <p>2 cloves of garlic, minced</p>
                </li>
                <li>
                    <p>1/2 tsp. of hot red pepper</p>
                </li>
                <li>
                    <p>28 oz. of canned tomatoes, preferably San Marzano</p>
                </li>
                <li>
                    <p>2 tbsp. of parsley, chopped</p>
                </li>
            </ul>
        </section>
        <section>
            <title>Preparation</title>
            <ol>
                <li>
                    <p>Heat olive oil in a large saucepan on medium</p>
                </li>
                <li>
                    <p>Add garlic and hot red pepper and sweat until fragrant
</p>
                </li>
                <li>
                    <p>Add tomatoes, breaking up into smaller pieces</p>
                </li>
                <li>
                    <p>Simmer on medium-low heat for at least 20 minutes</p>
                </li>
                <li>
                    <p>Add parsley</p>
                 </li>
                 <li>
                   <p>Simmer for another five minutes</p>
                 </li>
                 <li>
                    <p>Serve over long pasta.</p>
                 </li>
             </ol>
        </section>
    </body>
</topic>
```

FIGURE 1.8 Marinara sauce recipe as an XDITA topic. The XML tags are still visible and the recipe looks like a DITA topic. However, it is no longer a task, since LwDITA's initial specification only includes one topic type.

- Title: A label that connotes the purpose of the content that is associated with it
- Short description: A brief depiction of the purpose or theme of a topic
- Prolog: A container for metadata about a topic (for example, author information or subject category)
- Body: A container for the main content of a topic. It might include several sections
- Section: An organizational division within a topic. It can have an optional title.

These common elements can be represented, with modifications to accommodate different authoring languages, in XDITA, HDITA, and MDITA. Figure 1.9 shows the same recipe as a topic tagged in HDITA, the LwDITA authoring format that uses HTML5.

The topic has commonly-used HTML5 elements like headings and lists, but an additional benefit of HDITA is that topics authored in this LwDITA format do not require a transformation process to generate a publishable outcome (Figure 1.10). Because the topics are HTML5 files, they can be rendered in any web browser, and publishers can customize the rendered format with a standard CSS stylesheet.

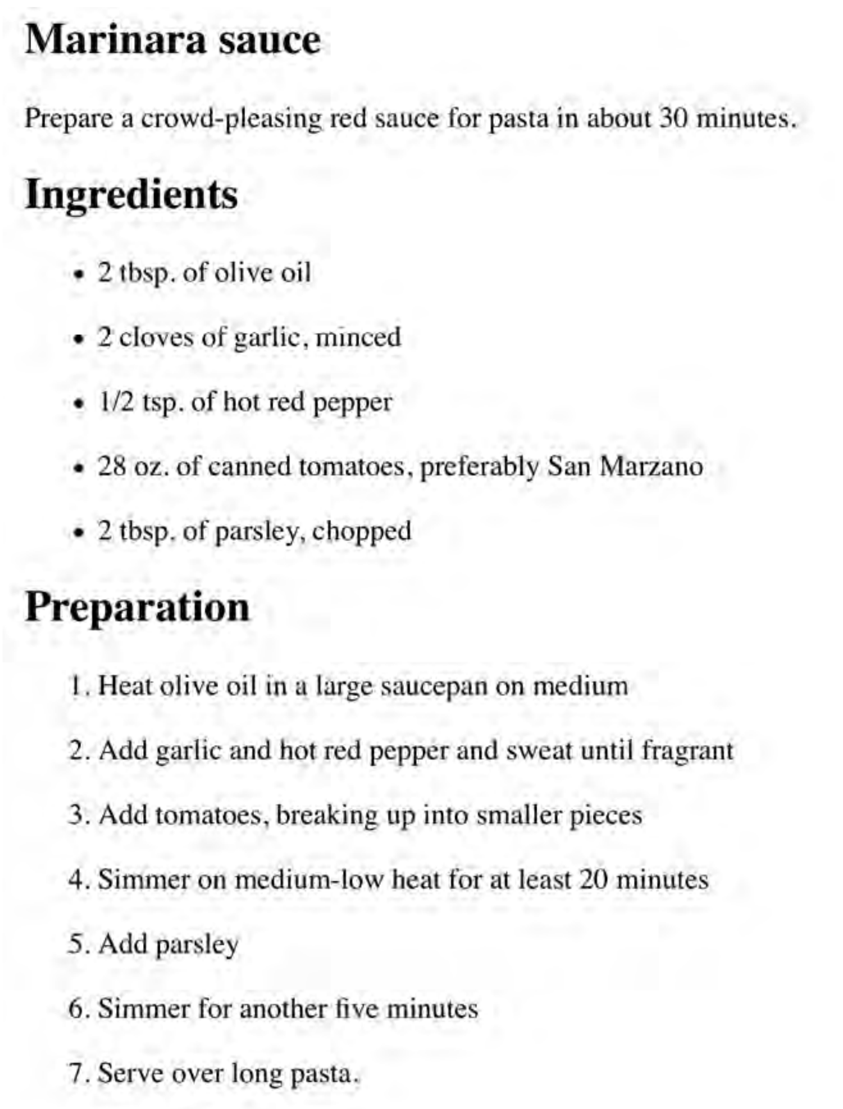
MDITA, the LwDITA authoring format that uses Markdown, can also be used to structure the recipe for marinara sauce (figure 1.11). In its core profile, MDITA provides structure for major elements present in DITA XML and Markdown (title, sections, lists, etc.). In its extended profile, MDITA allows a header authored in YAML[4] (the recursive acronym for *YAML Ain't Markup Language*) with metadata about the topic's author and some categories that we have been carrying since the original XML sample.

Regardless of its authoring format, when transformed with LwDITA-aware tools into information deliverables for human users, the topic for the marinara sauce recipe would pretty much look the same. That is a key feature of LwDITA: end users will not know the author's process and will just receive information products with consistent structure. Figure 1.12 shows a PDF version of the XDITA topic created with the DITA-OT.

An author trained in principles of structured authoring could use XDITA and take advantage of its DITA-like sections, elements, and constraints. XDITA provides some of the DITA mechanisms for reuse and single sourcing that could be essential for a technical communicator but probably distracting or confusing for a casual content contributor. HDITA can be less intimidating for collaborators with experience creating content in HTML5, while still including reuse and filtering options. Both XDITA and HDITA can be authored in WYSIWYG editors that keep code and tags hidden (permanently or temporarily, depending on the specific software tool) from the content creator. MDITA is a plain text variant for developers and authors who do not need advanced content reuse capabilities (but they still can use them with raw HDITA code fragments). All three formats, however, are compatible with each other and also with topics created according to the DITA XML standard. All three formats also incorporate fundamental actions of content authoring, like staging, coaching, and describing, which are essential moves of

```
<!DOCTYPE html>
<meta name="author" content="Unknown">
<meta name="keywords" content="Italian">
       <title>Marinara sauce</title>
      <body>
        <article id="t-marinara">
            <h1>Marinara sauce</h1>
            <p>Prepare a crowd-
pleasing red sauce for pasta in about 30 minutes.</p>
            <h2>Ingredients</h2>
            <ul>
                <li>
                    <p>2 tbsp. of olive oil</p>
                </li>
                <li>
                    <p>2 cloves of garlic, minced</p>
                </li>
                <li>
                    <p>1/2 tsp. of hot red pepper</p>
                </li>
                <li>
                    <p>28 oz. of canned tomatoes, preferably San Marzano</p>
                </li>
                <li>
                    <p>2 tbsp. of parsley, chopped</p>
                </li>
            </ul>
            <h2>Preparation</h2>
            <ol>
                <li>
                    <p>Heat olive oil in a large saucepan on medium</p>
                </li>
                <li>
                    <p>Add garlic and hot red pepper and sweat until fragrant
</p>
                </li>
                <li>
                    <p>Add tomatoes, breaking up into smaller pieces</p>
                </li>
                <li>
                    <p>Simmer on medium-low heat for at least 20 minutes</p>
                </li>
                <li>
                    <p>Add parsley</p>
                </li>
                <li>
                    <p>Simmer for another five minutes</p>
                </li>
                <li>
                    <p>Serve over long pasta.</p>
                </li>
            </ol>
         </article>
   </body>
```

FIGURE 1.9 Marinara sauce recipe as an HDITA topic. The topic is now an HTML5 article, and the elements' structure looks very similar to what can be accomplished with XML.



Marinara sauce

Prepare a crowd-pleasing red sauce for pasta in about 30 minutes.

Ingredients

- 2 tbsp. of olive oil
- 2 cloves of garlic, minced
- 1/2 tsp. of hot red pepper
- 28 oz. of canned tomatoes, preferably San Marzano
- 2 tbsp. of parsley, chopped

Preparation

1. Heat olive oil in a large saucepan on medium
2. Add garlic and hot red pepper and sweat until fragrant
3. Add tomatoes, breaking up into smaller pieces
4. Simmer on medium-low heat for at least 20 minutes
5. Add parsley
6. Simmer for another five minutes
7. Serve over long pasta.

FIGURE 1.10 HDITA recipe for marinara sauce seen on a web browser. An added benefit of HDITA is an instant presentation view that does not require processing or transformation to generate a basic publishable outcome.

```
---
id: t-marinara
author: Unknown
category: Italian
---
# Marinara Sauce
Prepare a crowd-pleasing red sauce for pasta in about 30 minutes.

## Ingredients
- 2 tbsp. of olive oil
- 2 cloves of garlic, minced
- 1/2 tsp. of hot red pepper
- 28 oz. of canned tomatoes, preferably San Marzano
- 2 tbsp. of parsley, chopped.

## Preparation
1. Heat olive oil in a large saucepan on medium
2. Add garlic and hot red pepper and sweat until fragrant
3. Add tomatoes, breaking up into smaller pieces
4. Simmer on medium-low heat for at least 20 minutes
5. Add parsley
6. Simmer for another five minutes
7. Serve over long pasta.
```

FIGURE 1.11 Marinara sauce recipe as an MDITA topic. The major change in this version is the absence of tags to represent elements.

technical communication (Eli Review, n.d.) that act as commonplace elements in the repertoire of an author. Chapter 3 will look at how those common moves of technical communication relate to content structures in DITA and LwDITA.

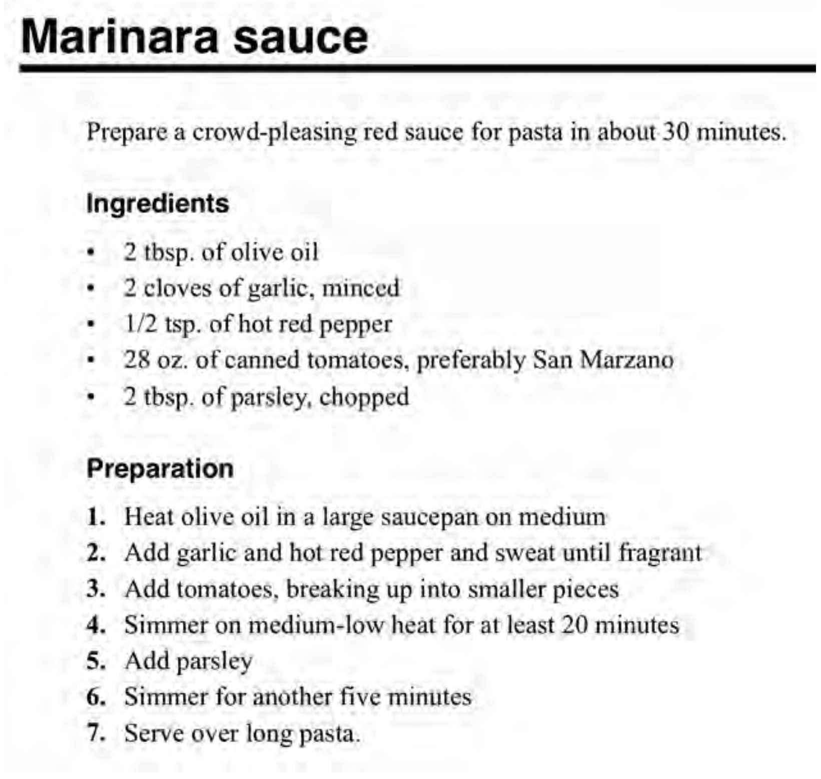
For processing purposes, a single DITA map can combine topics created in different LwDITA formats (Figure 1.13).

Deliverables created from the sample map in Figure 1.13 can include a print cookbook or an online recipe guide, based on the automation tools used by Chef Pedro and his team in a specific publishing scenario.

Computer Code for Human Authors

DITA and all three LwDITA formats are undeniably code. Calling them "computer code," however, could offend programmers and developers. Particularly for technical communication students and practitioners with backgrounds in writing and the Humanities, *this is* their computer code. They will probably not use advanced programming languages, but they work with XML, HTML5, and even Markdown code that for them requires a different kind of thinking than desktop publishing workflows involving long document files with a word processor. These are the skills that someone like IBM's James Mathewson expects from a technical communication graduate.

Authors using any combination of LwDITA formats do not need new technological skills. They will continue "the move away from a document-based to a



Marinara sauce

Prepare a crowd-pleasing red sauce for pasta in about 30 minutes.

Ingredients

- 2 tbsp. of olive oil
- 2 cloves of garlic, minced
- 1/2 tsp. of hot red pepper
- 28 oz. of canned tomatoes, preferably San Marzano
- 2 tbsp. of parsley, chopped

Preparation

1. Heat olive oil in a large saucepan on medium
2. Add garlic and hot red pepper and sweat until fragrant
3. Add tomatoes, breaking up into smaller pieces
4. Simmer on medium-low heat for at least 20 minutes
5. Add parsley
6. Simmer for another five minutes
7. Serve over long pasta.

FIGURE 1.12 PDF version of the XDITA topic for the marinara sauce recipe. Compare to the deliverable produced from a DITA XML task in Figure 1.4. The topic gained sub-headings because the <section> element replaced more specific moves associated with a task.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE map PUBLIC "-//OASIS//DTD DITA Map//EN" "map.dtd">
<map>
    <title>Fantastic Cookbook</title>
    <topichead>
    <topicmeta><navtitle>Sauces and condiments</navtitle></topicmeta>
    <topicref href="t-tikkamasala.dita" format="dita" />
    <topicref href="t-mole.html" format="hdita" />
    <topicref href="t-marinara.md" format="mdita" />
    </topichead>
</map>
```

FIGURE 1.13 DITA map aggregating different LwDITA formats. The end users will see all the recipes with the same structure regardless of authoring process.

topic-based approach to developing, managing, and publishing content" (Andersen, 2014, p. 116) widely adopted in the field of technical communication. New knowledge will actually come in the form of abstractions that allow authors to identify rhetorical moves (e.g., staging with a **<shortdesc>**, guiding with **<steps>**, showing with **<example>**) that were strictly enforced in DITA XML but are not required in LwDITA. In DITA, for example, an author has strict tags for a short description element and hazard statements, whereas in some LwDITA formats those sections are just paragraphs. These thought processes follow the foundations of the skill set labeled as computational thinking in recent literature about computing education.

LwDITA is not for everyone. In a large organization with limited resources to train authors, the content integrity and structural consistency provided by DITA XML might be the best solution. Fortunately, DITA is still evolving and its technical committee at OASIS is at the time of this writing planning version 2.0 of this content standard. LwDITA does not have the objective of replacing DITA, which is still available for authors and teams who need its full capabilities. The abstraction tasks from this framework, however, will enable critical users to apply computational thinking and technical communication principles in a series of layers that reveal intelligent content structure beyond what a software tool allows.

The 2012 Adobe video described a future of technical communication that now is more like its present, or even its past. For some, technical communication is still about structured information and intelligent content that adapts to users' needs. However, complex XML code is only one way (and maybe approaching obsolescence) to get there. Simplified lightweight markup (or *markdown*) cannot be ignored because authors are creating topics directly on the online environments where they will be read; after all, not every publishing project needs advanced reuse and filtering. Furthermore, coding and automating content delivery is but a "metal" element in the complicated process behind authoring and publishing intelligent content. Human beings in charge of content creation need to move their "mental" focus to the abstraction thinking behind the rhetorical decisions that make content intelligent.

Before we move on to specific how-to and examples of creating intelligent content with LwDITA following principles of computational thinking, we need to revisit the origins of DITA XML and analyze how its content structures and discourse conventions (based on the archetypal computer manual) evolved into LwDITA, and those are the main themes of the next chapter.

Notes

1 I present the recipe in the source's original Pascal Case; all other code examples in this book will use lowercase, which is more commonly associated with DITA best practices.

2 http://docs.oasis-open.org/dita/dita/v1.3/errata01/os/complete/part2-tech-content/langRef/containers/task-elements.html#task2

3 "a plain text format for writing structured documents, based on formatting conventions from email and usenet," http://commonmark.org

4 http://yaml.org

References

2.2.1.1 The topic as the basic unit of information. (2016, October 25). Retrieved from http://docs.oasis-open.org/dita/dita/v1.3/os/part1-base/archSpec/base/topicdefined.html#topicdefined

2.7.1.6 Troubleshooting topic. (2016, October 25). Retrieved from http://docs.oasis-open.org/dita/dita/v1.3/errata01/os/complete/part3-all-inclusive/archSpec/technicalContent/dita-troubleshooting-topic.html#dita-troubleshooting-topic

2.7.1.7 Glossary entry topic. (2016, October 25). Retrieved from http://docs.oasis-open.org/dita/dita/v1.3/errata01/os/complete/part3-all-inclusive/archSpec/technicalContent/dita-glossary-topic.html#glossaryArch

AdobeTCS. (2012, July 16). *Future of TechComm*. [Video File]. Retrieved from https://youtu.be/dSdhnyDF0YY

Andersen, R. (2013). The value of a reciprocal relationship between research and practice. *Information Management News*. Retrieved from http://www.infomanagementcenter.com/publications/e-newsletter/may-2013/the-value-of-a-reciprocal-relationship-between-research-and-practice/

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.

Andersen, R., & Batova, T. (2015). The current state of component content management: An integrative literature review. *IEEE Transactions on Professional Communication*, 58(3), 247–270.

Applen, J. D., & McDaniel, R. (2009). *The rhetorical nature of XML: Constructing knowledge in networked environments*. New York: Routledge.

Bacha, J. (2009). Single sourcing and the return to positivism: The threat of plain-style, arhetorical technical communication practices. In G. Pullman & B. Gu (Eds.) *Content management: Bridging the gap between theory and practice* (pp. 143–159). Amityville, NY: Baywood.

Baker, M. (2013). Every page is page one: Topic-based writing for technical communication and the web. Laguna Hills, CA: XML Press.

Baker, M. (2016, March 1). Algorithms: Separating content from formatting. Retrieved from https://techwhirl.com/algorithms-separating-content-from-formatting/

Bellamy, L., Carey, M., & Schlotfeldt, J. (2012). *DITA best practices: A roadmap for writing, editing, and architecting in DITA*. Upper Saddle River, NJ: IBM Press.

Clark, D. (2016). Content strategy: An integrative literature review. *IEEE Transactions on Professional Communication*, 59(1), 7–23.

Cowan, C. (2010). *XML in technical communication* (2nd ed.). Peterborough: Institute of Scientific and Technical Communication.

Day, D., Priestley, M., & Schell, D. (2001, March 1). *Introduction to the Darwin Information Typing Architecture*. Retrieved from http://www.ibm.com/developerworks/library/x-dita1/

DITAWriter (n.d.). Companies Using DITA. Retrieved June 19, 2018, from http://www.ditawriter.com/companies-using-dita/

DITAWriter. (2016, July 15). *Don Day and Michael Priestly [sic]* on the beginnings of DITA: Part 2. Retrieved from http://www.ditawriter.com/don-day-and-michael-priestly-on-the-beginnings-of-dita-part-2/

Eberlein, K.J. (2016). Content reuse. In R. Gallon (Ed.) *The Language of Technical Communication* (pp. 54–55). Laguna Hills, CA: XML Press.

Eli Review. (n.d.). The Essential moves of technical communication. Retrieved from http://elireview.com/content/curriculum/techcom/

Etheridge, A. (2016, May 17). Experts talk DITA & localization – Michael Priestley. Retrieved from http://www.whp.net/en/experts-talk-dita-localization-michael-priestley/

Evia, C. (2017). Authoring standards-based intelligent content the easy way with Lightweight DITA. *Proceedings of the 35th ACM International Conference on the Design of Communication*.

Evia, C., Sharp, M. R., & Perez-Quiñones, M. A. (2015). Teaching structured authoring and DITA through rhetorical and computational thinking. *IEEE Transactions on Professional Communication*, 58(3), 328–343.

Evia, C., & Priestley, M. (2016). Structured authoring without XML: Evaluating lightweight DITA for technical documentation. *Technical Communication*, 63(1), 23–37.

Evia, C., Eberlein, K., & Houser, A. (2018). *Lightweight DITA: An introduction*. Version 1.0. OASIS.

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 Pub.

Glushko, R. J., & McGrath, T. (2008). Document engineering: Analyzing and designing documents for business informatics and web services. Cambridge, MA: The MIT Press.

Guzdial, M. (2016). *Learner-centered design of computing education: Research on computing for everyone*. San Rafael, CA: Morgan & Claypool.

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

Hackos, J. T. (2016). International standards for information development and content management. *IEEE Transactions on Professional Communication*, 59(1), 24–36.

Johnson-Eilola, J. (1996). Relocating the value of work: Technical communication in a post-industrial age. *Technical Communication Quarterly*, 5(3), 245–270.

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

Kerzreho, N. (2016). Component content management system. In R. Gallon (Ed.) *The language of technical communication* (pp. 60–61). Laguna Hills, CA: XML Press.

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

Lauren, B., & Pigg, S. (2016). Toward multidirectional knowledge flows: Lessons from research and publication practices of technical communication entrepreneurs. *Technical Communication*, 63(4), 299–313.

Mathewson, J. [James_Mathewson]. (2016, September 8). Yes. But writing reusable content is a rare skill. Would that they taught it in college. Essays from whole cloth is more the thing. [Tweet]. Retrieved from https://twitter.com/James_Mathewson/status/774044623840870400

O'Keefe, S. (2009). Structured authoring and XML. In O'Keefe, S. (Ed.) *The compass: Essential reading about XML, DITA, and Web 2.0*. Durham, NC: Scriptorium Publishing Services, Inc.

OASIS Darwin Information Typing Architecture (DITA) TC. (n.d.). Retrieved from https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=dita

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. (2009). Technical writing 101: A real-world guide to planning and writing technical content. Durham, NC: Scriptorium Press.

Rockley, A., Cooper, C., & Abel, S. (2015). *Intelligent Content: A Primer*. Laguna Hills, CA XML Press.

Rude, C. D. (2015). Building identity and community through research. *Journal of Technical Writing and Communication*, 45(4), 366–380.

Sapienza, F. (2002). Does being technical matter? XML, single source, and technical communication. *Journal of Technical Writing and Communication*, 32(2), 155–170.

Stolley, K. (2008). The lo-fi manifesto. *Kairos: A Journal of Rhetoric, Technology, and Pedagogy* 12(3). Retrieved from http://kairos.technorhetoric.net/12.3/

Wachter-Boettcher, S. (2012). Content everywhere: Strategy and structure for future-ready content. Brooklyn, N.Y: Rosenfeld Media.

White, L.W. (2016). Single sourcing. In R. Gallon (Ed.) *The language of technical communication* (pp. 56–57). Laguna Hills, CA: XML Press.

Wing, J. (2006). Computational thinking. *Communications of the ACM*, 49(3), 33–35.

Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 366(1881), 3717–3725. https://doi.org/10.1098/rsta.2008.0118

Wing, J.M. (2011). Research notebook: Computational thinking – what and why? Retrieved from https://www.cs.cmu.edu/link/research-notebook-computational-thinking-what-and-why

2.2.1.1 The topic as the basic unit of information . (2016, October 25). Retrieved from http://docs.oasis-open.org/dita/dita/v1.3/os/part1-base/archSpec/base/topicdefined.html#topicdefined
2.7.1.6 Troubleshooting topic . (2016, October 25). Retrieved from http://docs.oasis-open.org/dita/dita/v1.3/errata01/os/complete/part3-all-inclusive/archSpec/technicalContent/dita-troubleshooting-topic.html#dita-troubleshooting-topic
2.7.1.7 Glossary entry topic . (2016, October 25). Retrieved from http://docs.oasis-open.org/dita/dita/v1.3/errata01/os/complete/part3-all-inclusive/archSpec/technicalContent/dita-glossary-topic.html#glossaryArch
AdobeTCS . (2012, July 16). Future of TechComm. [Video File]. Retrieved from https://youtu.be/dSdhnyDF0YY
Andersen, R. (2013). The value of a reciprocal relationship between research and practice. Information Management News. Retrieved from http://www.infomanagementcenter.com/publications/e-newsletter/may-2013/the-value-of-a-reciprocal-relationship-between-research-and-practice/
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), 115157.
Andersen, R. , & Batova, T. (2015). The current state of component content management: An integrative literature review. IEEE Transactions on Professional Communication, 58(3), 247270.
Applen, J. D. , & McDaniel, R. (2009). The rhetorical nature of XML: Constructing knowledge in networked environments. New York: Routledge.
Bacha, J. (2009). Single sourcing and the return to positivism: The threat of plain-style, arhetorical technical communication practices. In G. Pullman & B. Gu (Eds.) Content management: Bridging the gap between theory and practice (pp. 143159). Amityville, NY: Baywood.
Baker, M. (2013). Every page is page one: Topic-based writing for technical communication and the web. Laguna Hills, CA: XML Press.
Baker, M. (2016, March 1). Algorithms: Separating content from formatting. Retrieved from https://techwhirl.com/algorithms-separating-content-from-formatting/
Bellamy, L. , Carey, M. , & Schlotfeldt, J. (2012). DITA best practices: A roadmap for writing, editing, and architecting in DITA. Upper Saddle River, NJ: IBM Press.
Clark, D. (2016). Content strategy: An integrative literature review. IEEE Transactions on Professional Communication, 59(1), 723.
Cowan, C. (2010). XML in technical communication (2nd ed.). Peterborough: Institute of Scientific and Technical Communication.
Day, D. , Priestley, M. , & Schell, D. (2001, March 1). Introduction to the Darwin Information Typing Architecture. Retrieved from http://www.ibm.com/developerworks/library/x-dita1/
DITAWriter (n.d.). Companies Using DITA. Retrieved June 19, 2018, from http://www.ditawriter.com/companies-using-dita/
DITAWriter . (2016, July 15). Don Day and Michael Priestly [sic] on the beginnings of DITA: Part 2. Retrieved from http://www.ditawriter.com/don-day-and-michael-priestly-on-the-beginnings-of-dita-part-2/
Eberlein, K.J. (2016). Content reuse. In R. Gallon (Ed.) The Language of Technical Communication (pp. 5455). Laguna Hills, CA: XML Press.
Eli Review . (n.d.). The Essential moves of technical communication. Retrieved from http://elireview.com/content/curriculum/techcom/ 30
Etheridge, A. (2016, May 17). Experts talk DITA & localization Michael Priestley. Retrieved from http://www.whp.net/en/experts-talk-dita-localization-michael-priestley/
Evia, C. (2017). Authoring standards-based intelligent content the easy way with Lightweight DITA. Proceedings of the 35th ACM International Conference on the Design of Communication.
Evia, C. , Sharp, M. R. , & Perez-Quiones, M. A. (2015). Teaching structured authoring and DITA through rhetorical and computational thinking. IEEE Transactions on Professional Communication, 58(3), 328343.
Evia, C. , & Priestley, M. (2016). Structured authoring without XML: Evaluating lightweight DITA for technical documentation. Technical Communication, 63(1), 2337.

Evia, C. , Eberlein, K. , & Houser, A. (2018). Lightweight DITA: An introduction. Version 1.0. OASIS.
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. 7385). Amityville, NY: Baywood Pub.
Glushko, R. J. , & McGrath, T. (2008). Document engineering: Analyzing and designing documents for business informatics and web services. Cambridge, MA: The MIT Press.
Guzdial, M. (2016). Learner-centered design of computing education: Research on computing for everyone. San Rafael, CA: Morgan & Claypool.
Hackos, J. T. (2011). Introduction to DITA: A user guide to the Darwin Information Typing Architecture including DITA 1.2 (2nd edition). Comtech Services, Inc.
Hackos, J. T. (2016). International standards for information development and content management. IEEE Transactions on Professional Communication, 59(1), 2436.
Johnson-Eilola, J. (1996). Relocating the value of work: Technical communication in a post-industrial age. Technical Communication Quarterly, 5(3), 245270.
Kaplan, N. (2014, May 3). The death of technical writing, part 1. Retrieved from https://customersandcontent.com/2014/05/03/the-death-of-technical-writing-part-1/
Kerzreho, N. (2016). Component content management system. In R. Gallon (Ed.) The language of technical communication (pp. 6061). Laguna Hills, CA: XML Press.
Kimber, E. (2012). DITA for practitioners, volume 1: Architecture and technology. Laguna Hills, CA: XML Press.
Lauren, B. , & Pigg, S. (2016). Toward multidirectional knowledge flows: Lessons from research and publication practices of technical communication entrepreneurs. Technical Communication, 63(4), 299313.
Mathewson, J. [James_Mathewson] . (2016, September 8). Yes. But writing reusable content is a rare skill. Would that they taught it in college. Essays from whole cloth is more the thing. [Tweet]. Retrieved from https://twitter.com/James_Mathewson/status/774044623840870400
OKeefe, S. (2009). Structured authoring and XML. In OKeefe, S. (Ed.) The compass: Essential reading about XML, DITA, and Web 2.0. Durham, NC: Scriptorium Publishing Services, Inc.
OASIS Darwin Information Typing Architecture (DITA) TC . (n.d.). Retrieved from https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=dita
Priestley, M. , Hargis, G. , & Carpenter, S. (2001). DITA: An XML-based technical documentation authoring and publishing architecture. Technical Communication, 48(3), 352367.31
Pringle, A. , & OKeefe, S. (2009). Technical writing 101: A real-world guide to planning and writing technical content. Durham, NC: Scriptorium Press.
Rockley, A. , Cooper, C. , & Abel, S. (2015). Intelligent Content: A Primer. Laguna Hills, CA XML Press.
Rude, C. D. (2015). Building identity and community through research. Journal of Technical Writing and Communication, 45(4), 366380.
Sapienza, F. (2002). Does being technical matter? XML, single source, and technical communication. Journal of Technical Writing and Communication, 32(2), 155170.
Stolley, K. (2008). The lo-fi manifesto. Kairos: A Journal of Rhetoric, Technology, and Pedagogy 12(3). Retrieved from http://kairos.technorhetoric.net/12.3/
Wachter-Boettcher, S. (2012). Content everywhere: Strategy and structure for future-ready content. Brooklyn, N.Y: Rosenfeld Media.
White, L.W. (2016). Single sourcing. In R. Gallon (Ed.) The language of technical communication (pp. 5657). Laguna Hills, CA: XML Press.
Wing, J. (2006). Computational thinking. Communications of the ACM, 49(3), 3335.
Wing, J. M. (2008). Computational thinking and thinking about computing. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 366(1881), 37173725. https://doi.org/10.1098/rsta.2008.0118
Wing, J.M. (2011). Research notebook: Computational thinking what and why? Retrieved from https://www.cs.cmu.edu/link/research-notebook-computational-thinking-what-and-why

Before Intelligent Content, there was the Computer Manual

2.2.1.1 The topic as the basic unit of information . (2016, October 25). Retrieved from http://docs.oasis-open.org/dita/dita/v1.3/os/part1-base/archSpec/base/topicdefined.html#topicdefined
Adorno, T.W. (1991). The culture industry: Selected essays on mass culture. London: Routledge.
Baker, M. (2013). Every page is page one: Topic-based writing for technical communication and the web. XML Press.
Campbell, K.S. & Naidoo, J.S. (2017). Rhetorical move structure in high-tech marketing white papers. Journal of Business and Technical Communication. 31(1) 94118.
Carey, M. . (2014). Developing quality technical information: A handbook for writers and editors. 3rd Ed. Upper Saddle River, NJ: IBM Press.
Carroll, J.M. (1990). The Nurnberg funnel: Designing minimalist instruction for practical computer skill. Cambridge, MA: M.I.T. Press.
Closs, S. (2016). DITA the topic-based XML standard: A quick start. Switzerland: Springer.
Cohen, G. & Cunningham, D.H. (1984). Creating technical manuals: A step-by-step approach to writing user-friendly instructions. New York, NY: McGraw-Hill.
DITAWriter . (2016, July 13). Don Day and Michael Priestley on the beginnings of DITA: Part 1. Retrieved from http://www.ditawriter.com/don-day-and-michael-priestley-on-the-beginnings-of-dita-part-1/
Eberlein, K.J. , Harrison, N. , Hunt. J. , & Swope, A. (2015). DITA 1.3: Why three editions? OASIS.
Eli Review . (n.d.). The Essential moves of technical communication. Retrieved from http://elireview.com/content/curriculum/techcom/
Flower, L. (1989). Rhetorical problem solving: Cognition and professional writing. In M. Kogen (Ed.), Writing in the business professions (pp.336). Urbana, IL: NCTE.
Hargis, G. (1998). Developing quality technical information: A handbook for writers and editors. Upper Saddle River, NJ: Prentice Hall.
Hargis, G. . (2004). Developing quality technical information: A handbook for writers and editors. 2nd Ed. Upper Saddle River, NJ: Prentice Hall.
Hart-Davidson, W. & Omizo, R. (2017). Genre signals in textual topologies. In L. Walsh & C. Boyle (Eds.), Topologies as techniques for a post-critical rhetoric (pp. 99123). New York, NY: Palgrave Macmillan.
Johnson. T. (2015, June 29). Slides, notes, and lessons learned at the STC summit 2015 in Columbus, Ohio. [Blog post]. Retrieved from http://idratherbewriting.com/2015/06/29/lessons-learned-at-the-stc-summit-2015/
Licklider, J.C.R. (1965). Libraries of the future. Cambridge, MA: M.I.T. Press.
Miller, C. R. (2015). Genre change and evolution. In N. Artemeva & A. Freedman (Eds.), Genre studies around the globe: Beyond the three traditions (pp. 154185). Lexington, KY: Trafford.
Miller, C.R. & Selzer, J. (1985). Special topics of argument in engineering reports. In L. Odell & D. Goswami (Eds.), Writing in non-academic settings. New York: Guilford.
Dean, M. . (1983). Producing quality technical information. San Jose, CA: IBM.59
OKeefe. S. (2010). XML: The death of creativity in technical writing? Intercom, (February), 3637.
Price, J. (1984). How to write a computer manual: A handbook of software documentation. Menlo Park, CA: Benjamin Cummings.
Ross, D. (Ed.). (2017). Topic-driven environmental rhetoric. New York: Routledge.
Svenvold, M. (2015, January 26). The disappearance of the instruction manual. Popular Science. Retrieved from https://www.popsci.com/instructions-not-included
Patterson, D. (1975). Technical writing: User manuals. Asterisk, 2(5), 89.
Prelli, L. (1989). A rhetoric of science: Inventing scientific discourse. Columbia, SC: University of South Carolina Press.
Priestley, M. (2001). DITA XML: A reuse by reference architecture for technical documentation. Proceedings of the 19th annual international conference on computer documentation, 152156.
Priestley, M. (2001a). XML and the Darwin Information Typing Architecture (DITA). Communication Design Quarterly Review, 2(2), 34.
Priestley, M. , Hargis, G. , & Carpenter, S. (2001). DITA: An XML-based technical documentation authoring and publishing architecture. Technical Communication, 48(3), 352367.

Rigo, J. (1976). User manual outline. Asterisk, 2(8), 710.

How DITA Evolved Into LwDITA

About . (n.d.). Retrieved from https://www.informationmapping.com/en/information-mapping
CommonMark . (n.d.). CommonMark. Retrieved from http://commonmark.org/
DITA Technical Committee . (2013, October 8). October 8, 2013 Phone meeting. Archived at https://markmail.org/message/rzexlutrvecucxty
DITA Technical Committee . (2014, May 10). May 10, 2014 Phone meeting. Archived at https://markmail.org/message/zanoxhojqycwbbkn 80
Evia, C. , & Priestley, M. (2016). Structured Authoring without XML: Evaluating Lightweight DITA for Technical Documentation. Technical Communication, 63(1), 2337.
Evia, C. , Eberlein, K. , & Houser, A. (Eds.) (2018). Lightweight DITA: An introduction. Version 1.0. OASIS.
Gruber, J. (2004, December 17). Markdown. Retrieved from https://daringfireball.net/projects/markdown/
Hackos, J. T. (2016). International standards for information development and content management. IEEE Transactions on Professional Communication, 59(1), 2436.
Horn, R.E. . (1969). Information mapping for learning and reference. United States Air Force.
Information Mapping . (2011). Information Mapping and DITA: Two worlds, one solution [White paper]. Retrieved from http://www.writec.com/Media_public/WP-dita-white-paper.pdf
LimitedDitaProfile . (2011, April 4). In OASIS DITA Wiki. Retrieved from https://wiki.oasis-open.org/dita/LimitedDitaProfile
Miller, C. R. (2015). Genre change and evolution. In N. Artemeva & A. Freedman (Eds.), Genre studies around the globe: Beyond the three traditions (pp. 154185). Lexington, KY: Trafford.
Priestley, M. (2012, March 20). Lightweight DITA proposal - 13076. Message archived at https://markmail.org/message/rydwna45z4yln7xe
Priestley, M. (2012a, October 23). DITA starter sets proposal 13051. Message archived at https://markmail.org/message/lpmjxf5zigre2qf5
Priestley, M. (2013, April 16). Lightweight DITA: A preview of the DITA 1.3 feature. Paper presented at the Content Management Strategies/DITA North America Conference . Providence, RI.
Priestley, M. (2013a, December 10). A lightweight DITA update. Retrieved from https://www.slideshare.net/mpriestley/a-lightweight-dita-update
Priestley, M. (2014, April 11). Overview of Lightweight DITA (XDITA and HDITA). Retrieved from http://dita-archive.xml.org/blog/overview-of-lightweight-dita-xdita-and-hdita
Priestley, M. (2014a, May 27). Proposal for Lightweight DITA subcommittee. Message archived at https://markmail.org/message/cutgxe627wodz2ca
Priestley, M. (2014b, September 15). Strawman deliverables list. Message archived at https://markmail.org/message/jjb6z2n5dkgia2bd
Priestley, M. , Hargis, G. , & Carpenter, S. (2001). DITA: An XML-based technical documentation authoring and publishing architecture. Technical Communication, 48(3), 352367.
Priestley, M. , Evia, C. , & Ai, L. (2015, April 20). Does DITA need tags? Paper presented at the Content Management Strategies/DITA North America Conference . Chicago, IL.
Rockley, A. , Cooper, C. , & Abel, S. (2015). Intelligent Content: A Primer. Laguna Hills, CA: XML Press.
The Information Mapping Method . (n.d.). Retrieved from https://www.informationmapping.com/en/?option=com_content&view=article&id=50&Itemid=400
Veen, J. (Host). (2017, August 8). Everything you know about web design changed last march [Audio podcast]. Retrieved from https://www.relay.fm/presentable/28

The LwDITA Initial Authoring Formats

Evia, C. , Sharp, M. R. , & Perez-Quiones, M. A. (2015). Teaching structured authoring and DITA through rhetorical and computational thinking. IEEE Transactions on Professional Communication, 58(3), 328343.
Evia, C. , Eberlein, K. , & Houser, A. (Eds.) (2018). Lightweight DITA: An introduction. Version 1.0. OASIS.
Hackos, J. T. (2011). Introduction to DITA: A user guide to the Darwin Information Typing Architecture including DITA 1.2 (2nd edition). Comtech Services, Inc.
Software Carpentry (2018). Introducing the shell. Retrieved from http://swcarpentry.github.io/shell-novice/01-intro/index.html
W3C (2008, November 26). Extensible Markup Language (XML) 1.0 (Fifth Edition) . Retrieved from https://www.w3.org/TR/xml/

The LwDITA Map Components

Bellamy, L. , Carey, M. , & Schlotfeldt, J. (2012). DITA best practices: A roadmap for writing, editing, and architecting in DITA. Upper Saddle River, NJ: IBM Press.
Evia, C. , Eberlein, K. , & Houser, A. (2018). Lightweight DITA: An introduction. Version 1.0. OASIS.

The LwDITA Topic Components

Evia, C. , Sharp, M. R. , & Perez-Quiones, M. A. (2015). Teaching structured authoring and DITA through rhetorical and computational thinking. IEEE Transactions on Professional Communication, 58(3), 328343.
Evia, C. , Eberlein, K. , & Houser, A. (Eds.) (2018). Lightweight DITA: An introduction. Version 1.0. OASIS.
GitHub Flavored Markdown Spec . (2017, August 1). Retrieved from https://github.github.com/gfm/
Robidoux, C. (2008). Rhetorically structured content: Developing a collaborative single-sourcing curriculum, Technical Communication Quarterly, 17(1), 110135.
Rockley, A. (2002). 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.
Rockley, A. , Cooper, C. , & Abel, S. (2015). Intelligent Content: A Primer. Laguna Hills, CA: XML Press.
W3C . (2017, December 14). HTML 5.2: W3C recommendation. Retrieved from https://www.w3.org/TR/html52

The Abstractions Behind Intelligent Content

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), 115157.
Andersen, R. (2014a). Toward a more integrated view of technical communication. Communication Design Quarterly Review. 2(2), 1016.

Bailie, R.A. & Urbina, N. (2013). Content strategy: Connecting the dots between business, brand, and benefits. Laguna Hills, CA: XML Press.

Barba, L.A. (2016, March 5). Computational thinking: I do not think it means what you think it means [Blog post]. Retrieved from http://lorenabarba.com/blog/computational-thinking-i-do-not-think-it-means-what-you-think-it-means/

College Board . (2017). AP computer science principles. Retrieved from https://apcentral.collegeboard.org/pdf/ap-computer-science-principles-course-and-exam-description.pdf

Dayton, D. (2004). The future of technical communication according to those who teach it. Paper presented at the conference of the Society for Technical Communication.

Emig, J. (1977). Writing as a mode of learning. College Composition and Communication, 28(2), 122128.

Evia, C. , Sharp, M. R. , & Perez-Quiones, M. A. (2015). Teaching structured authoring and DITA through rhetorical and computational thinking. IEEE Transactions on Professional Communication, 58(3), 328343.

Flower, L. (1989). Rhetorical problem solving: Cognition and professional writing. In M. Kogen (Ed.), Writing in the business professions (pp.336). Urbana, IL: NCTE.

Gu, B. & Pullman G. (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. 112). Amityville, NY: Baywood.

Guzdial, M. (2008). Education: Paving the way for computational thinking. Communications of the ACM. 51(8), 2527.

Guzdial, M. (2016). Learner-centered design of computing education: Research on computing for everyone. New York, NY: Morgan & Claypool.

Guzdial, M. (2016a, January 13). What does it mean to assess computational thinking? [Blog post]. Retrieved from https://computinged.wordpress.com/2016/01/13/what-does-it-mean-to-assess-computational-thinking/

Hart-Davidson, W. (2001). Reviewing and rebuilding technical communication theory: Considering the value of theory for informing change in practice and curriculum. Paper presented at the Society for Technical Communication conference.

Hart-Davidson, W. (2010). Content management: Beyond single-sourcing. In R. Spilka (Ed.) Digital literacy for technical communication: 21st centuty theory and practice (pp. 128143). New York, NY: Routledge.

Hayakawa, S. I. & Hayakawa, A.R. (1990). Language in thought and action. Orlando, FL: Harcourt.

ISTE . (2018). ISTE standards for students. Retrieved from http://www.iste.org/standards/for-students

Musaeus, P. , Tatar, D. , & Rosen, M. (2017) Medical computational thinking: Computer scientific reasoning in the medical curriculum. In P.J. Rich & C.B. Hodges (Eds.) Emerging research, practice, and policy on computational thinking (pp. 8598), Springer.163

Pappano, L. (2017, April 4). Learning to think like a computer. The New York Times, Retrieved from https://www.nytimes.com/2017/04/04/education/edlife/teaching-students-computer-code.html

Perkovi, L. , Settle, A. , Hwang, S. , & Jones, J. (2010). A framework for computational thinking across the curriculum. Proceedings of the fifteenth annual conference on innovation and technology in computer science education, 123-127.

Rockley A. & Gollner, J. (2011, January 10). An intelligent content strategy for the enterprise. Bulletin of the American Society for Information Science and Technology. Retrieved from https://onlinelibrary.wiley.com/doi/full/10.1002/bult.2011.1720370211

Rockley, A. (2016, July 14). Why automation is the future of content creation. Retrieved from https://contentmarketinginstitute.com/2016/07/automation-future-content-creation/

Rockley, A. , Cooper, C. , & Abel, S. (2015). Intelligent Content: A Primer. Laguna Hills, CA: XML Press.

Rude, C. D. (2015). Building identity and community through research. Journal of Technical Writing and Communication, 45(4), 366380.

Senske, N. (2017). Evaluation and impact of a required computational thinking course for architecture students. Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education, 525530.

Tanenbaum, A.S. (2006). Structured computer organization (5th Ed.). Upper Saddle River, NJ: Pearson Prentice Hall.

Abstractions in the LwDITA Content Lifecycle

2.2.4.2.1 Conditional processing attributes (2018, June 19). Retrieved from http://docs.oasis-open.org/dita/dita/v1.3/errata02/os/complete/part3-all-inclusive/archSpec/base/conditional-processing-attributes.html#conditional-processing-attributes
Albers, M. J. (2003). Single sourcing and the technical communication career path. Technical Communication, 50(3), 335344.
Ament, K. (2003). Single sourcing: Building modular documentation. Norwich, NY: William Andrew.
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), 115157.
Bacha, J. (2009). Single sourcing and the return to positivism: The threat of plain-style, arhetorical technical communication practices. In G. Pullman & B. Gu (Eds.) Content management: Bridging the gap between theory and practice (pp. 143159). Amityville, NY: Baywood.196
Bailie, R.A. (2014). Content strategy. In S. Abel & R.A. Bailie (Eds.), The language of content strategy (pp. 1415). Laguna Hills, CA: XML Press.
Bailie, R.A. & Urbina, N. (2013). Content strategy: Connecting the dots between business, brand, and benefits. Laguna Hills, CA: XML Press.
Baker, M. (2013). Every page is page one: Topic-based writing for technical communication and the web. Laguna Hills, CA: XML Press.
Batova, T. (2014). Component content management and quality of information products for global audiences: An integrative literature review. IEEE Transactions on Professional Communication. 57(4), 325339.
Bellamy, L. , Carey, M. , & Schlotfeldt, J. (2012). DITA best practices: A roadmap for writing, editing, and architecting in DITA. Upper Saddle River, NJ: IBM Press.
Carliner, S. (2010). Computers and technical communication in the 21st century. In R. Spilka (Ed.), Digital literacy for technical communication: 21st century theory and practice (pp. 2150). New York, NY: Routledge.
Carter, L. (2003). The implications of single sourcing on writers and writing. Technical Communication, 50(3), 14.
Clark, D. (2007). Content management and the separation of presentation and content. Technical Communication Quarterly, 17(1), 3560.
Clark, D. (2016). Content strategy: An integrative literature review. IEEE Transactions on Professional Communication, 59(1), 723.
Creekmore, L. (2014). Metadata. In S. Abel & R.A. Bailie (Eds.), The language of content strategy (pp. 2829). Laguna Hills, CA: XML Press.
Flower, L. (1989). Rhetorical problem solving: Cognition and professional writing. In M. Kogen (Ed.), Writing in the business professions (pp.336). Urbana, IL: NCTE.
Gallagher, J.R. (2017). Writing for algorithmic audiences. Computers and composition. 45, 2535.
Getto, G. & St. Amant, K. (2014). Designing globally, working locally: Using personas to develop online communication products for international users. Communication Design Quarterly. 3(1), 2446.
GitHub Flavored Markdown Spec . (2017, August 1). Retrieved from https://github.github.com/gfm/
Hackos, J. T. (2011). Introduction to DITA: A user guide to the Darwin Information Typing Architecture including DITA 1.2 (2nd edition). Denver, CO: Comtech Services, Inc.
Halvorson, K. & Rach, M. (2012). Content strategy for the web (2nd Ed.). Berkeley, CA: New Riders.
Harrison, N. (2016). Content variables. In R. Gallon (Ed.), The language of technical communication (pp. 6667). Laguna Hills, CA: XML Press.
Hart-Davidson, W. . (2007). Coming to content management: Inventing infrastructure for organizational knowledge work. Technical Communication Quarterly. 17(1), 1034.
ISTE . (2018). ISTE standards for students. Retrieved from http://www.iste.org/standards/for-students
Pringle, A. , & OKeefe, S. (2009). Technical writing 101: A real-world guide to planning and writing technical content. Durham, NC: Scriptorium Press.
Rothrock, L. & Narayanan, S. (2011). Human-in-the-loop simulations: Methods and practice. London: Springer-Verlag.197

Swarts, J. (2010). Recycled writing: Assembling actor networks from reusable content. Journal of Business and Technical Communication. 24(2), 127163.

Swisher, V. (2014). Global content strategy: A primer. Laguna Hills, CA: XML Press.

Vazquez, J. (2016). Conditional content. In R. Gallon (Ed.), The language of technical communication (pp. 6465). Laguna Hills, CA: XML Press.

Wachter-Boettcher, S. (2012). Content everywhere: Strategy and structure for future-ready content. Brooklyn, N.Y: Rosenfeld Media.

Willerton, R. (2015). Plain language and ethical action: A dialogic approach to technical content in the 21st century. New York, NY: Routledge.

Conclusion

Jones, N. N. , Moore, K. R. , & Walton, R. (2016). Disrupting the past to disrupt the future: An antenarrative of technical communication. Technical Communication Quarterly, 25(4), 211229. http://dx.doi.org/10.1080/10572252.2016.1224655

Mehlenbacher, B. (2013). What is the future of technical communication? In J. Johnson-Eilola & S.A. Selber (Eds.), Solving problems in technical communication (pp. 187208). Chicago, IL: The University of Chicago Press.