Teaching Case

Teaching Structured Authoring and DITA Through Rhetorical and Computational Thinking

—Carlos Evia, Matthew R. Sharp, and Manuel A. Pérez-Quiñones

***Abstract*—Background:** *The diffusion of component content management and structured authoring workflows and technologies in technical communication requires that instructors of documentation courses determine effective ways to teach component content management to students who may initially be intimidated by authoring environments and structures, such as the Darwin Information Typing Architecture (DITA). This teaching case describes how component content management and DITA were integrated into the Creating User Documentation course of an undergraduate professional writing program.* **Research questions:** *How can instructors of technical and professional writing best teach English and humanities students to operate within a structured authoring workflow? How can computational abstraction be combined with students' previously acquired genre knowledge to ease their adoption of the DITA to create technical documentation?* **Situating the case:** *The development of this course was informed by literature from a variety of scholarly and industry sources, which reveal connections between DITA, computational thinking, and Rhetorical Genre theory. Specifically, the concept of "layers of abstraction" guides the development of the course's structure, allowing students to separate and independently process the various aspects of a structured authoring workflow.* **How the case was studied:** *The case was studied informally through the experience of the authors as they developed and taught the course, through informal discussions and structured interviews with industry professionals, and through student reflections from discussion forum posts from Fall 2012 through Fall 2013.* **About the case:** *Initially developed with a focus on print manuals and online help, the course began teaching topic-based authoring in the mid-2000s; however, most enterprise-level editors and tools were cost-prohibitive for students and faculty. Furthermore, many computing concepts associated with structured authoring were intimidating for an audience of students in an English department. An affordable solution was adopting the open-source DITA standard, using free trials or open-source editors. The intimidation factor was minimized by designing the course around five layers of abstraction that draw on students' previous rhetorical knowledge: Layer 1: Developing quality documentation, Layer 2: Separating content from design, Layer 3: Authoring granular content with XML, Layer 4: Authoring and linking Component Content Management modules with DITA, and Layer 5: Single-sourcing and content reuse. This case discusses each layer of abstraction, the associated assignments for each layer, and the results of each layer based on student feedback.* **Results and conclusions:** *Although the course is not universally loved by students, it has seen many successes and provides a much-needed foundation in component content management and structured authoring for students who might become technical communicators. The teaching team has learned to avoid overemphasizing coding and automation in structured authoring, maintain a solid grounding on writing principles and good technical communication requirements, and draw upon students' existing knowledge of genres and their constraints.*

***Index Terms*—** *Abstractions, computer languages, DITA, documentation, rhetoric, standards, writing, XML.*

In "Component Content Management: Shaping the Discourse through Innovation Diffusion Research and Reciprocity," Andersen [1] makes a call to combine efforts from the academia and industry sides of technical communication to create awareness about the importance of component content-management diffusion. Structured authoring is a key component of an effective component content-management environment, and while many technical communicators still work for companies that use unstructured content [2], adoption of a structured authoring workflow is increasing [3]. In addition, reports on industry practices [3] and skills required in job postings [2] frequently cite the Darwin Information Typing Architecture (DITA) as the most popular content structure in technical communication. This steady increase in demand for the skills of structured authoring is not the only reason to teach about DITA in technical communication courses, however. McShane claims that embracing technologies, such as XML, offer "us—and our students—an opportunity to employ the full-range of our rhetorical skills" [4, p. 74]. In other words, it allows our students to critically engage with a new technology, providing them additional

Manuscript received May 26, 2014; revised October 12, 2014; accepted December 16, 2014. Date of current version January 29, 2016.

C. Evia is with the Department of English and Center for Human–Computer Interaction, Virginia Polytechnic Institute and State University, Blacksburg, VA 24061 USA (email: cevia@vt.edu).
M. R. Sharp is with the Humanities and Communication Department, Embry-Riddle Aeronautical University, Daytona Beach, FL 32114 USA (email: matthew.sharp@erau.edu).
M. A. Pérez-Quiñones is with the Department of Software and Information Systems, University of North Carolina at Charlotte, Charlotte, NC 28223 USA (email: perez.quinones@uncc.edu).

IEEE 10.1109/TPC.2016.2516639

insight into how technologies can affect the content they produce and letting them experience the rhetorical affordances and constraints of a particular technological choice. Eble [5] echoes this connection between rhetorical skills and technology by repurposing the traditional canons of memory and delivery as database/archives and interface/distribution, respectively. These are the canons of component content management, where content is stored in the database and repurposed to be distributed across various versions and media. Our students live in this world. They use structured content, and will most likely be required to create and manage it at some point in their careers.

This teaching case is based on experiences covering structured authoring and DITA in the Professional Writing undergraduate program of a major research university in Virginia. The authors have worked to design and teach a course, ENGL 3814: *Creating User Documentation* which, in its current iteration, includes modules and assignments using Extensible Markup Language (XML), Extensible Stylesheet Language Transformations (XSLT), and DITA, acknowledging that these are currently the most common ways to structure technical content. The case includes recommendations for programs with modest budgets for software tools.

During the course redesign, ongoing since 2006, the progression of topics related to XML and component content management has represented a learning challenge for most English majors and minors. Whereas basic XML syntax and tags are relatively easy to teach and learn, the automation of deliverables via XSLT and the DITA Open Toolkit is much more complicated. The issue is compounded by a student population with vastly different computing skills. At this large school, known for its engineering programs, some students in the class have experience with programming languages from other academic disciplines; however, most English and communication students often avoid anything too "technical." Furthermore, students are uncomfortable with the separation of format from content because, as noted by McShane [4], they have always had control over the format of their documents in Microsoft Word and even in many HTML-based web environments.

The solution proposed in this case includes incorporating elements of Computational Thinking, which takes an approach to "solving problems, designing systems and understanding human behavior by drawing on concepts fundamental to computer science" [6, p. 33]. Specifically, our approach focuses on the concept of abstraction, which has been identified as a key component of Computational Thinking [7], [8] and proposes learning progressions that lead to a whole understanding of the methodology, tools, and benefits of a particular system. Abstraction allows us 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. By separating the process of producing documentation through component content management into several layers of abstraction that students can tackle one at a time, we allow them to focus their efforts on each layer before adding another. By the end of the semester, what once seemed daunting and antithetical to their experience, makes sense as part of a larger process of creating documentation.

This case combines these elements of computational thinking with applied rhetoric to create replicable models in response to the critical mass of industry content-management adoption at the heart of this special issue. For readers of the IEEE TRANSACTIONS ON PROFESSIONAL COMMUNICATION, this manuscript offers a model for incorporating a technical standard for documentation (DITA) into teaching practices of technical and professional communication. Particularly, the case is driven by the following research questions:

- How can instructors of technical and professional writing best teach English/humanities students to operate within a structured authoring workflow?
- How can computational abstraction be combined with students' previously acquired genre knowledge to ease their adoption of DITA to create technical documentation?

We begin by situating the case within relevant literature about Rhetorical Genre theory, computational thinking, and DITA. Then, we discuss how the case was studied as a an experience-based needs assessment and evaluation report, followed by an in-depth discussion about the case itself, including how it was developed and executed based on five layers of abstraction. Finally, we end by discussing our general conclusions about the course, the limitations of the case, and suggestions for future research.

SITUATING THE CASE

This section situates the case within the literature of the research and practice of technical communication and computer science. After explaining how the literature was selected, we then discuss the key concepts to the course's development and evolution: Rhetorical Genre theory, DITA, and computational thinking.

How Literature was Selected In order to provide a balanced perspective on the importance of component content management and DITA, the literature informing this teaching case was chosen from a variety of scholarly and industry sources. Scholarly publications in rhetoric, writing, and technical communication that define genres and collections of genres by the actions they perform within larger collectives of activity were used to discuss Rhetorical Genre theory. Publications and textbooks written by industry experts on technical communication and content management inform our discussions of DITA, and scholarly publications from journals in computer science inform our discussions of computational thinking. Specifically, we referenced literature that defined computational thinking as a method for problem solving and discussed how it can be applied to other disciplines, particularly in nontechnical fields.

Rhetorical Genre Theory According to Bitzer's definition, "a work of rhetoric is pragmatic;(. . .)it performs some task" [9, p. 4]. It responds to a certain exigence in order to effect change in the world. Miller [10] claims that a rhetorical understanding of genre, therefore, must understand it as a typified social action in response to a recurring exigence. Russell used the words "routinized" and "operationalized" [11, p. 546]. In other words, genres are a standardized way of taking action through text, and that standardization often includes a unique structure or expected types of content. Genres, however, are defined by the tasks they accomplish, not by their format. Likewise, the granular topics within component content-management workflows are defined by the actions they perform—their purpose—within the documentation [12]. That definition as a specific type of topic, then, includes some expectations for the content. For instance, in DITA, tasks tell the user how to perform a certain action, and they are expected to contain steps in the imperative mood. Concepts, on the other hand, explain ideas and processes and are expected to contain paragraphs made up of declarative statements. As noted by Robidoux [13], writing technical content requires the integration of various types of information into coherent structures that users can predict. Much like different genres, these different information types perform different actions individually, but they work together to accomplish the larger goals of a documentation project.

The combination of various information types within a component content-management workflow is reminiscent of a genre set. Devitt [14] introduced the notion of genre sets, which are collections of genres used to perform a number of related tasks. Her study described the various genres used by tax accountants to perform their work, including various types of letters, memoranda, proposals, reviews, and protests. Bazerman [15] expanded the concept of a genre set to a genre system where multiple genres from multiple genre sets coordinate to accomplish the goals of an entire system of activity. The genre system has a variety of users, all appropriately responding to each other's genre usage with genres from their own genre sets in order to accomplish the larger goals of the activity system. This collaboration of genres and users mirrors the ways in which various topics and authors in a DITA environment collaborate to author content for multiple channels and deliverables.

While undergraduate English students may not have this depth of knowledge about genres, sets, and systems, they do take courses that touch on the rhetorical understanding of genres. For the most part, the students in the course seem to understand the concept of genre and that different genres accomplish different goals, perform different tasks (a layer of abstraction in itself), and have different requirements and constraints. *Creating User Documentation* was designed to capitalize upon that rhetorical knowledge in order to help students separate the various layers of abstraction that comprise the process of creating technical documentation within a component content-management workflow with DITA.

DITA DITA "is an XML-based, end-to-end architecture for authoring, producing, and delivering technical information" [16, p. 1]. As described by some of its main architects, 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. [16, p. 1]

A brief history of DITA starts by explaining it as a standard published by the non-profit Organization

for the Advancement of Structured Information Standards (OASIS). The DITA specification was donated by IBM to OASIS in March 2004 [17], "but it has evolved substantially since that initial donation to encompass a very wide scope of requirements indeed" [18, p. 6]. Nevertheless, DITA's authoring core still resides on its main three information types or topics: concept, task, and reference, which "represent the vast majority of content produced to support users of technical information" [19, p. 7]. Hackos lists the following items as key benefits of DITA for technical communicators:

- 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 [19, p. 9].

These benefits contribute to DITA's popularity as a component content-management solution for technical documentation.

Even though DITA is not the only component content-management solution for technical content, it does dominate industry conferences and publications. In fact, most books about DITA come from industry practitioners or consultants and, in some cases, are self-published: XML Press offers *DITA for Practitioners* [18] and *DITA for Print* [20]; Scriptorium Press published *The DITA Style Guide* [17], SDI Global released *Practical DITA* [21], The Rockley Group published *DITA 101* [22], IBM Press offers *DITA Best Practices* [23], and CIDM has *Introduction to DITA* [19]. However, academic discussions about this standard and its importance to the discipline are scarce, and finding materials or an approach to teach it at the college level required years of experimenting and collaboration between the departments of English and computer science.

Computational Thinking In 2007, this manuscript's first author became coprincipal investigator in the community-building project "LIKES: Living in the KnowlEdge Society," sponsored by the National Science Foundation's Directorate of Computer and Information Science and Engineering's Pathways to Revitalized Undergraduate Computing Education [24]. The LIKES mission was to study the implementation of innovative computing concepts in traditionally noncomputing disciplines, and develop and implement "tools and techniques that enable learning of both computing concepts and the concepts of the disciplines" [24]. One of the key terms that emerged during the LIKES workshops was computational thinking, which was introduced as a problem-solving technique for tasks involving computing automation.

Aho defines computational thinking as "the thought processes involved in formulating problems so their solutions can be represented as computational steps and algorithms" [25, p. 832]. Therefore, if an XML-based component content-management workflow was described in computational thinking terms, it would focus on the computational solution to the problem of documentation chaos created by an abundance of unstructured content: a standardized and automated repository of reusable modules that can adapt to different users' needs. According to Wing, computing is "the automation of our abstractions," and "the essence of computational thinking is abstraction" [7, p. 3717]. The computing literature offers many definitions for abstraction; however, most of the work has been in the context of abstraction in computing programs (see, for example, [26]–[28]) and not in other areas. Wing [7] posits that computing abstractions are richer than in other disciplines, as they go beyond numerical abstractions and cover symbolic abstractions as well as other types, such as algorithmic and representational.

In the case of a computational solution for structured content, abstraction is a combination of two elements: information representation and separation of concerns (layers of abstraction). The use of abstract representations enables focusing on relevant parts of a problem. The multiple layers of abstractions allow processing (computationally) some layers without concerns about the details to be encountered at other layers. Computing is concerned with automating those abstractions, and 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 of abstraction [7].

In the particular example of developing technical content with DITA, one layer of abstraction is in the ability to draw upon authors' existing knowledge of genres to identify unique types of content and apply

the appropriate conventions of that content type. Another is the specific way to mark up content to produce documentation within a structured authoring workflow. This layer allows a technical communicator to separate content from format. In order for the final deliverable(s) to be successfully produced, however, the technical communicator also needs to understand the relationships between DITA and the other layers of abstraction involved in the publishing process, such as the concept of single sourcing and the XSLT files and scripts that automate a transformation to PDF, HTML, or another deliverable method. Programming experiments for novice computing students have found that computational thinking concepts, such as abstraction, have eased the transition into coding [29], which is part of the challenge to English students who are learning to write XML but are so used to working in a desktop publishing environment such as Microsoft Word.

Just as Wing predicts that deeper computational thinking within the domain of computer science

> may enable scientists and engineers to model and analyze their systems on a scale orders of magnitude greater than they are able to handle today. [7, p. 3719]

Likewise, the use of computational thinking and abstraction within technical communication (to create component content-management repositories and deliverables) allows modeling, analyzing, and creating systems and collections of documents that are orders of magnitude more complex and more efficient than ever before. This view is similar to the abstraction ladder [30], in which each level reveals additional characteristics of a concept for more convenient understanding. Thus, the seemingly daunting concept of "authoring content in a Component Content Management workflow with DITA" is divided into easy-to-process levels before introducing any computational automation (to be performed later by the DITA Open Toolkit or specialized software).

For the purposes of this case, we have identified the following layers of abstraction that comprise the process of creating technical documentation within a component content-management workflow with DITA:

> Layer 1: Developing quality documentation
> Layer 2: Separating content from design
> Layer 3: Authoring granular content with XML
> Layer 4: Authoring and linking component content-management modules with DITA
> Layer 5: Single-sourcing and content reuse

This case reports on how an undergraduate course modeled on these layers of abstraction allows students within a Professional Writing program and an English department to capitalize on their existing rhetorical strengths while effectively improving their computing skills and learning to create documentation deliverables using DITA.

How This Case was Studied

This section reports on the methods used to assess the skills that needed to be incorporated into the course and to evaluate the course over three semesters. First, we state the research methodology, and then we discuss the requirements for expert and student participants. Next, we discuss how the data were collected from participants and, finally, how the data were analyzed.

Choice of a Research Methodology This case is primarily an experience-based needs assessment and evaluation report with Institutional Review Board's (IRB's) approval to review, analyze, and report certain data.

Participants The case benefited from interviews with technical communication experts from industry and academia, and analysis of forum reflections from students taking the course (approved by Virginia Tech's IRB with protocols 11–311 and 13–773, respectively). The primary objective for obtaining expert opinions from academia and industry was to conduct a needs assessment that contributed to the instructors' ideas on how to revise the course.

Expert participants from industry and academia were selected based on either their publication record on topics related to documentation, single sourcing, and DITA, or their presence in online social media promoting training for structured authoring in the workplace or adoption of DITA or similar standards. An initial round of email invitations went out to the experts in 2011 and continued after nonresponding and declining participants were discarded. Experts were asked for permission to be quoted by name in publications, and did not receive monetary compensation for their participation.

Student reflection posts on their assignments were mandatory and part of their semester grades. Only students enrolled in the 3000-level course *Creating User Documentation* during the Fall 2012, Spring 2013, and Fall 2013 semesters were included in this

case. Our IRB-approved consent form specified that students taking the course during the Fall 2013 semester could opt out of having their comments included in publications and reports. For students who took the course in the Fall 2012 and Spring 2013 semesters, the IRB approved the analysis and quoting of posts as exempt, acknowledging that students had already been graded for those assignments and tracking them down for approval would be a difficult task. As instructors of record of the course sections included in this case, we ensured that participation did not affect student grades. Students in the sections covered by our IRB documents (a total of 53 students: 29 female and 24 male) were not asked to identify or describe previously acquired knowledge of rhetorical genres or technical writing. Instead, we assumed a baseline of skills based on the course's mandatory prerequisites at the junior level.

How Data Were Collected An important source of direction and content for the course comes from a research-to-practice loop grounded on input from experts on DITA, content management, and technical communication in industry and academia. Some experts include practitioners who have graduated from our program in Professional Writing, but over the years, more structured interviews were conducted with influential names included as follows:

- J.D. Applen, Associate Professor of English, University of Central Florida
- Dave Clark, Associate Dean of the College of Letters & Science, University of Wisconsin-Milwaukee
- Don Day, former Lead DITA Architect, IBM
- Michael Priestley, Enterprise Content Technology Strategist, IBM
- Sarah O'Keefe, President, Scriptorium Publishing
- Julio Vazquez, Senior Content Analyst at Vasont Systems.

Interviews were conducted, according to individual preferences, via telephone or email, with questions related to the relevance of structured authoring in the technical communication curriculum, and ideas and experiences about challenges and solutions of adopting (or teaching about) an authoring workflow with DITA or a similar topic-based writing methodology. Furthermore, Sarah O'Keefe and Julio Vazquez have each visited the course to share first-hand, industry experiences with students, following Don Day's advice to conduct a “brown bag” session with DITA experts.

Documenting student feedback was a key method for answering this case's research questions about the feasibility of implementing a structured authoring workflow with DITA in an English course. Students were required to write reflection blogs or forum posts during and after completing major assignments in *Creating User Documentation*. The posts' requirements were broadly specified in the syllabus as follows:

> These posts are intended to supplement course readings and class discussion, particularly in terms of thinking through the reasons why technical communicators would choose certain genres, authoring techniques, or tools, as well as how those choices affect technical communicators, their companies, and their audiences. You may also use this space to reflect on issues you encounter in the process of completing course assignments and suggest ways for your peers to avoid similar issues. These posts should be a space for your ‘confession’ as a technical communicator in training.

Although the reflection posts have been a tradition in the course since 2005, only those from courses that took place over three semesters in 2012 and 2013, approved by the Virginia Tech IRB, are included in this case as measures of results for each layer of abstraction in the process of creating technical documentation within a component content-management workflow with DITA. The reflection posts were created as a way to document students' success and struggle stories when adopting a DITA-based documentation workflow. The posts were not used as a tool to grade or assess student projects. Each major project, with or without DITA, was graded with a rubric embedded with detailed, qualitative feedback on the mechanical effectiveness of writing, the rhetorical impact of the produced documents, and the quality of technical information included.

How Data Were Analyzed During the process of reading, commenting on, and grading the reflection posts, we looked for examples of success stories, struggles, or frustrations related to the adoption of a component content-management authoring environment with DITA. For the purpose of this course, topic-based authoring and single-sourcing techniques are as relevant to the subject matter as persuasive writing and technical accuracy of instructions; therefore, we did not see DITA as a tool. (Our process was tool-agnostic: most students decided to use Oxygen XML for their

projects and, thus, wrote about problems specific to that editor, but others worked in TextEdit and the command-line interface using the DITA Open Toolkit and had different tool-related experiences.) The reflection posts worked as a virtual gauge to measure the adoption of DITA and understanding of a topic-based writing workflow, and did not have a rigorous analysis component attached to them.

Assuring Credibility and Trustworthiness We were as interested in detecting problems as we were in looking for successful implementations of DITA authoring workflows. Therefore, our analysis of reflection posts attached to each layer of abstraction attempts to cover all facets of the spectrum without bias.

About the Case

This section describes the process of constant revision of syllabus, materials, and assignments in *Creating User Documentation* with the purpose of reflecting trends and the need from technical communicators in industry and academia. The section starts with a description of the problem behind this case, and then it provides a brief description of the course itself. Then, we describe the process involved in developing the course as a solution to the problem, with an emphasis on the layers of abstraction identified to introduce concepts of component content management, structured authoring, and DITA from a computational thinking and Genre theory perspective.

Problem Originally based on the production of print manuals and online help, the course *Creating User Documentation* needed to reflect the needs of a

> field as diverse as technical communication, which includes both academics and industry professionals and an incredibly diverse array of interests, practices, and specialties. [31, p. 43]

Preliminary research in the mid-2000s pointed to topic-based writing and XML-based single sourcing as key topics for a documentation course. Because most enterprise-level editors and tools were cost-prohibitive for students and faculty, an affordable solution was to explore the open-source DITA standard. However, many computing concepts associated with a DITA authoring workflow (working on the command-line, writing XML and XSLT code, customizing transformations, and content references) were too intimidating for an audience of students in an English department.

Brief Description of the Course *Creating User Documentation* is an advanced technical documentation course in a Professional Writing program. Students come to this course with required experience in composition, introductory topics in professional writing, technical editing and, most recently, writing for digital media. Since 2006, the course has experienced frequent changes and adaptions based on industry and academia trends. Table I outlines the skills that students currently learn in the class through a combination of the following documentation assignments:

(1) Standard operating procedure in a word processor
(2) Template-based how-to documentation in XML
(3) Adaptation of the standard operating procedure into DITA
(4) DITA-based user assistance system
(5) Multimodal/hybrid how-to project.

Process for Developing the Course This section discusses the process of determining what to teach in the course as well as how those plans were realized, guided by the layers of abstraction identified as central to the process of developing content in a structured authoring environment.

Determining What to Teach: *Creating User Documentation* has been in the Professional Writing curriculum at Virginia Tech since the early 2000s. Our involvement started in 2004, and from that year until 2006, the main source of ideas on what to teach came from presentations and talks at conferences, such as the Summit of the Society for Technical Communication. Structured authoring and DITA were frequently mentioned at those events. Later on, social media interactions, industry conversations, and job ads continued the conversation on the need for good content-development skills and experience with XML and XSLT as desired characteristics of our graduates. From the more in-depth interviews with experts in industry and academia, however, we learned that structured authoring should be only one of the strong elements of a well-rounded curriculum. As a result, our work also included updating other courses such as *Developing Online Content* (formerly titled *Writing for the Web*) and creating *Writing and Digital Media*, which is a course on multimodal authoring and social media management. At the same time, the topic of abstraction as a key component of a problem-solving approach for learning computing concepts appeared on publications and the LIKES

workshops, sponsored by the National Science Foundation, mentioned earlier in this manuscript.

From those sources of information, we were determined to introduce structured authoring for component content-management environments through a combination of quality writing for technical documentation, a clear separation of content and design to enable single-sourcing and content reuse, a brief introduction to XML and XSLT to understand the concept of granularity, and application of the DITA as a standard for technical content.

Realizing the Plans: A computational thinking approach to the course helped solve the combined problems of creating technical documentation in DITA with nonexpensive or open-source tools in an English course. As a problem-solving method, computational thinking comprises the automation of abstractions and, in this case, the automations were handled by the DITA Open Toolkit and free or inexpensive DITA-aware editors. Dividing the course into five layers of abstraction simplified computing-related tasks for the students while taking advantage of their previously acquired rhetorical skills.

Although all assignments in the course are based on principles of minimalism in documentation and structured authoring, not all of them are about DITA and component content management. For example, the final assignment in the course has been an experiment in creativity and documentation, and in different semesters, students have created how-to videos (with or without elements of entertainment education); procedural comic books and web comics [32]; and optional hybrid DITA topics combining text, audio, and video [33].

Furthermore, teaching *Creating User Documentation* involves a constant search and evaluation of reading materials and software applications that make topics related to DITA, structured authoring, and component content management more accessible for an audience of undergraduate English majors and minors. In the textbook area, the course has moved since 2004 from an original focus on manuals as its default genre to topic-based writing and publication. Whereas the required textbook in 2004 was *The User Manual Manual: How to Research, Write, Test, Edit & Produce a Software Manual*, by Bremer [34], in the fall of 2014, the required texts were *Every Page Is Page One: Topic-Based Writing for Technical Communication and the Web*, by Baker [35], *Developing Quality Technical Information*, by Carey et al. [36], and *Introduction to DITA: A User Guide to the Darwin Information Typing Architecture*, by Hackos [19]. Regarding tools and software, the evolution has been from HTML created in Notepad or TextEdit to Oxygen XML Editor, with trial periods of XMLSpy, Serna XML, XMLmind, and even command-line transformations with the DITA Open Toolkit. These evaluations have been informal and based on student input, budgetary restrictions, and recommendations from social media and trade publications. For each layer of abstraction that guided the course's development, the following sections provide a general description of each layer, its assignments, and its results based on student feedback.

Layer 1: Developing Quality Documentation: The foundational structure of the course, before getting into any XML-related projects, is in the "technical writing process" (with an enhanced emphasis on audience analysis) introduced by Pringle and O'Keefe [37]. In a 10-step process, Pringle and O'Keefe summarize the requirements of a technical content project, with tasks that include identifying deliverables, planning and outlining, creating content, editing, and interacting with subject–matter experts, among others. Furthermore, based on a recommendation from IBM's Michael Priestley, a second foundational model for the course came from the characteristics of quality technical information listed by Carey *et al.* [37] as follows:

- Easy to use:
  - Task orientation
  - Accuracy
  - Completeness.
- Easy to understand:
  - Clarity
  - Concreteness
  - Style.
- Easy to find:
  - Organization
  - Retrievability
  - Visual effectiveness.

The main objective in this layer is to introduce technical documentation as a process and not just a product. Both models provide structure for the planning, authoring, and revising performed by students in the course. Following these models, students learn to see their technical content as an asset and learn about return on investment and interacting with clients and experts. This layer also includes an introduction to user experience,

TABLE I
QUICK FACTS ABOUT THE SOLUTION

Budget:	$289.00 spent on classroom licenses for oXygen XML Editor
Length of time needed to complete the project to this point:	Course has been in continuous redevelopment since 2006, but the study was completed over 3 regular semesters (16 months)
Skills used in the project:	• Planning and developing technical content • Writing XML code • Writing technical topics in DITA • Transforming DITA topics to PDF, HTML, and EPUB • Interviewing users and testing the effectiveness of deliverables
Technology used:	• Learning-management system (Sakai) • Computers and email • Telephone for interviews • XML editors

as students develop user personas and conduct interviews for audience analysis. Tools used in this layer are those with which students are already familiar, including a word processor (usually Microsoft Word) and spreadsheets, which allow students full control over content and format. Therefore, within this layer of abstraction, students are able to focus on creating quality documentation, before adding additional technologies and tools to that process.

Assignments. In this layer, students create a documentation plan following a model by Pringle and O'Keefe [37]. They include detailed personas and usage scenarios as they envision a standard operating procedure (SOP) for a series of tasks affecting their experience as students or residents of their college town. The layer ends with the actual SOP with a memo reporting on a qualitative usability test with three potential users.

Results. Students commonly expressed their surprise at how useful the documentation plan and personas actually became while writing the SOP, which implies that they were beginning to understand the importance of planning in the technical writing process as well as the necessity of a focus on the user and task-orientation—fundamental aspects of quality technical documentation.

> I think that writing the doc plan for the first assignment was a great start to the class because it gave me a good foundation for the rest of the course. Writing outlines and doc plans will be necessary for the rest of the assignments throughout the semester, and learning this skill early will help me develop my technical writing skills further

wrote a student from the fall 2012 semester.

A student from the fall 2013 semester added the following:

> The documentation plan provided a quick and easy checklist of topics that I needed to cover in my SOP. Referring to the plan after I thought I was finished helped me avoid neglecting topics that I initially claimed that I would cover, which shows its purpose and usefulness. I now understand how having a detailed documentation plan can help writers and clients avoid misunderstandings and keep clients from making up unreasonable expectations for the final product.

Layer 2: Separating Content From Design: Pringle and O'Keefe classify "the methodology for developing technical content into four levels:" chaos, page consistency, template-based authoring, and structured authoring [37, p. 40]. In the previous layer, students developed page-consistent documents in a word processor, and this second layer is about the process of abstraction behind the separation of content from design through the use of templates. Baker warns about the difficulties of teaching this layer in the following statement:

> One of the hardest things about moving technical writers from desktop publishing to

> structured writing is persuading them to give up responsibility for how the final output looks. Writers will keep looking for ways to specify layout, even in markup languages specifically designed to remove layout concerns. They understand their jobs in terms of the responsibilities their old tools imposed on them. [35, p. 87]

Tools used in this layer move beyond a word processor and focus on open-source code-friendly editors (including Komodo Edit, Notepad++, TextWrangler, and others) to start writing in HTML and XML. Students remain responsible for writing quality documentation, but this layer requires them to relinquish control of format in order to create content within an XML and template-based environment.

Assignments. Students develop content in XML for presentation on the web. They receive templates already created with CSS and then XSLT, which they cannot modify in this layer. The introduction to XML includes discussions and examples about its everyday uses, from RSS to KML, VoiceXML, and even iTunes. The main project is a flat-file database like a collection of movies or a catalog of books. Their XML "collections" at this point consist of one long content file with many instances of an element in a format similar to the following:

```
<movie mine = "yes">
 <title>Back to the Future</title>
 <director>Robert Zemeckis</director>
 <actor>Michael J. Fox</actor>
 <actress>Lea Thompson</actress>
 <genre>Science fiction</genre>
</movie>
```

The XSLT templates can then be modified to sort or filter movies by student-determined criteria, and they can also customize CSS files to change appearance, but only after they have completed the content for their projects.

Results. While some students found XML more difficult to learn than others, they still understood the importance of using markup languages in professional writing contexts. According to a student from the fall semester of 2012, "coding means the Internet; the Internet means publication, and publication means viewership."

A student from the spring 2013 semester authored the following "confession:"

> And even now, I continue to despise coding—except for in the context of creating user documentation. In the context of this class, for some reason, I don't hate it. In fact I enjoy it. Partially because I am learning a useful skill, but mostly because this course is taking a task that I have always associated with computer engineering and software expertise and presenting it to me in a way that connects code to the concepts I am familiar with: document layout, headings, chunking, and the like. When I was able to start thinking of using XML as simply another medium to create user documentation with, and not a magical formula that makes computers work their power, I was able to near-instantly wrap my head around the styles and languages of XML. I think it's rather amazing how simply shifting the context of coding seemed to snap it to another part of my brain entirely and allow me to overcome a major hangup in a day.

Furthermore, students seemed to understand the benefits of separating content from format. Many forum posts acknowledged the advantages of transforming content into multiple outputs but also acknowledged that it allowed them to focus on the content. A student from fall 2012 said that the separation of content from format "allow(s) us to focus on what we know best—writing." Later, the same student added that "[s]tructured formatting through XML allows our treasured content to be quickly and easily manipulated to suit the needs of the platform and our audience."

Many other students, however, were more accustomed to HTML, which allows more control over the format and presentation of information. In the fall of 2012 and the spring 2013, forum postings reflected a desire to learn more about the ways to customize the transformations of XML files. As a result, the fall 2013 course incorporated more detailed explanations of XSLT and CSS for customization.

Layer 3: Authoring Granular Content With XML: In their model of document engineering, Glushko and McGrath define granularity as "the extent to which a system contains separate components" [38, p. 633]. Component content-management-based systems depend on this granularity for linking and reuse. However, before students are exposed to those advanced topics, they need to master this third layer of abstraction. At this point, students are between simple XML catalogs and specialized

grammars such as DITA. In an interview, Sarah O'Keefe recommended

> that XML should be a core component of (technical writing) curricula. In addition to DITA, there should be some coverage of other standards, such as DocBook, S1000D, NLM, and perhaps TEI. This will give students a better understanding of DITA as *a* standard rather than *the* standard.

Thus, before embracing a specific standard, this layer has the objective of introducing component-based XML authoring and filtering with the purpose of presenting modules created in XML as equivalents of more traditional technical and professional writing genres. Tools used in this layer include XML-specific editors such as XMLSpy and Oxygen, and students are encouraged to modify content and templates as they interact with web-aimed transformations combining XML and JavaScript routines.

Assignments. Students author one or more XML documents with related content that then will be linked in different ways to adapt to users' needs thanks to XSLT filters and JavaScript DOM manipulations. In different semesters, the main project has been a detailed online cookbook with recipes reflecting a specific theme or type of cuisine (the XML files move around the repeating element ⟨recipe⟩ and its many children), a summary of a specific college football game (with instances of the element ⟨quarter⟩following an account of ⟨team⟩ and ⟨player⟩), and quick-start guides for electronics and gadgets (with the DITA-like element of ⟨step⟩ included in a parent ⟨gadget⟩).

Results. As the students advance to more complicated linking and transforming processes with XML, the layers maintain a connecting thread from the rhetorical purpose of each element or unit of content. A step within a recipe or a play within a quarter performs a certain action and, therefore, should be written in a certain way. Students' existing knowledge of genres and their conventions allowed them to understand this layer of abstraction more readily. For example, in the forums across all three semesters, students discussed the differences between different elements in each unit. In the unit on XML, they helped clarify for each other the difference between tips and notes, and in the DITA-specific units, they discussed the conventions of short descriptions, steps, and unordered steps. This genre-based focus on the purpose and conventions of each element naturally led to a focus on the purpose and conventions of each topic within the DITA standard.

This layer was really challenging for some students, as this post from the fall 2013 semester shows:

> The XML project was frustrating and confusing at first (...) It took a while to understand that we had to write the XML code, then also have corresponding XSLT and HTML pages for each module. However, the process wasn't so bad after I got the hang of it. It was convenient that I was able to copy & paste the majority of my code and only had to change a few numbers or words depending on which gadget/module I was writing for. Writing the code itself was definitely the most time-consuming part of the project, but it wasn't difficult.

Similar success-after-struggle stories from that semester include statements like “once I saw my work show up in the browser, I had a boost of confidence and encouragement. That's what makes XML so fun, but frustrating at the same time. It is either right or wrong. There is no in between with XML,” and

> the XML project wasn't exactly fun or enjoyable, but it was rewarding to see the end results. I thought of the project like a puzzle too because it had so many different pieces that had to be put together correctly for the page to work properly.

These stories reveal that while students initially had trouble producing a large amount of code, they eventually began to see its utility in creating the various pieces of granular content that each performs a specific function in the final deliverable.

Layer 4: Authoring and Linking Component Content-Management Modules With DITA: At the end of layer 3, students were satisfied with the results of their homebrewed structured authoring environment. This next layer starts by comparing their XML template to an international standard by asking “what if someone in another school or another country decided to add recipes/games/gadgets to your project?” As an open source, XML-based, international standard for structured authoring, DITA is at the core of layer 4. Building on the concept of genre, the discussion of DITA starts with understanding the usage cases for the topics of concept, task, and reference. In addition, the module explores how students build genre systems through topic-linking and the development of DITA maps to filter and organize topics for specific deliverables. The overall objective

of this layer is to make students comfortable with topic-based writing and the functions and responsibilities of a DITA author. Tools used include open-source and inexpensive DITA-aware editors and the DITA Open Toolkit.

Assignments. An effective introduction to DITA starts with content with which the students are already familiar. In this layer, the main project is to convert the SOP created in a word processor during layer 1 into valid DITA topics. Conversion of legacy documentation to DITA is not a simple copy-and-paste process. Rockley et al. point out that

> the ease of the conversion is only as good as the quality of the content to begin with. If the authors were consistent in how they created content and content used templates there is a good chance that the content can be converted programmatically, but sadly that is usually not the case. [22, p. 59]

Students need to envision their SOP, originally created for print delivery, as a web or mobile system. Sections become topics, and tables of contents transform into maps as students create content inventories to see what's missing or what needs to be cut in the conversion. Final deliverables include a PDF version of the SOP and a corresponding web help system.

Results. Students were able to transfer their knowledge of genre conventions by thinking of each topic type within DITA as its own genre with its own, unique constraints and structure. A fall 2012 student wrote that

> one of the most helpful elements of understanding DITA is grasping how there are different types of markups based on the type of information the document contains. With each different type of document I typed up for both assignments, I put myself in a certain frame of mind for completing each type of markup based on the main three categories.

Students were even able to see how the DITA topics were useful in projects that took place outside of a structured authoring environment (video tutorials, for example), demonstrating that they understood the relationships between multiple layers of abstraction well enough to transfer that abstraction to documentation in other media. A student from the spring of 2013 said, "I found it very interesting that without even realizing it we are using the same concepts from our DITA projects in our tutorial, and how everything seems to match up with a DITA element."

Furthermore, within this layer students began understanding the sequence of assignments for themselves, not necessarily as layers of abstraction, but as a logical sequence that allows them to build on what they learned earlier in the semester. A fall 2013 student wrote, "I agree that the DITA project was much easier than the XML project. I think that this was the case because we already were familiar with XML before we started this project. If we had done this project before doing the XML project, I believe it would have been overwhelming."

Layer 5: Single-Sourcing and Content Reuse: A longstanding definition of single-sourcing of technical documentation is the process of

> writing information once and using it many times. It does not mean writing it and then copying and pasting it into another source, or modifying the information for different needs such that you have multiple sources [39, p. 189]

Additionally, Bellamy et al. mention that "in the world of DITA, reuse is about writing content once and reusing that content whenever it's needed" [23, p. 183]. The purpose of this layer is to introduce students to the concepts of single-sourcing and reuse. In DITA, maps represent an accessible entry point to single-sourcing, and reuse can be achieved easily through content references (conrefs). The conref "is a method of transclusion, which means that you can include a topic or part of a topic in another topic by using a reference" [23, p. 184]. In this layer, students write content once, and then create multiple deliverables for different audiences, using some unique topics and some reused topics. Tools used in this layer include DITA-aware editors like Oxygen with advanced functions, such as conditional processing and filtering. In addition, students learn about customization of DITA deliverables, incorporating personalized CSS files to their web products, and exploring the use of plug-ins for PDF and Open Document Format for Office Applications (ODF).

Assignments. Following a recommendation from Julio Vazquez, formerly of IBM and Systems Documentation, Inc., the main project in this layer is a fictional scenario asking students to prepare breakfast recipes in DITA for different audiences. The assignment introduces "Armando the vegan roommate" and a bacon-loving mother who visits a student. Both need breakfast instructions and

recipes, but Armando requires a web help or mobile-friendly deliverable respecting his dietary restrictions, while "mom" wants a PDF with plenty of meat. The assignment description specifies that each deliverable "should include at least four different recipes, and the point of reuse is that some overlap is expected, or in this case, required: you will need at least two recipes with a conditional ingredient (or ingredients) that could be adapted for vegan and non-vegan users (example: replace tofu for eggs)." A recipe can have audience-based conditional processing instructions as follows:

<step audience="mom">

<cmd>Flip bacon with tongs after 5 minutes of cooking.</cmd>

</step>

<step audience="armando">

<cmd>Flip vegan bacon alternative with tongs after 7 minutes of cooking.</cmd>

</step>

Also, with a conref, a step written once,

<step id="stove"> <cmd>Place pan on stove top.</cmd></step>

can be called in another topic,

<step conref="t-bacon.dita#bacon/stove">.

This assignment still requires students to create quality documentation within a DITA environment, but the added complexity of reuse and conditional processing at this layer requires that they envision ways to reuse and adapt that content across various types of deliverables for different audiences.

Results. After mastering layer 4, most students seemed to appreciate the flexibility and customization options afforded by layer 5. A fall 2012 student, referencing Rockley et al.'s [22] discussion of content collections as a "library of content," said that

> the DITA maps function like the Dewey Decimal system, organizing and calling upon files as they are needed. The structured authoring style of DITA allows for content to easily be reused and the maps are the defining factor in how the topics are organized for use.

In the spring of 2013, a student commented on the following:

> After two separate assignments with DITA, I can see the reason why people may use it. It's an easy way to single-source your information and be able to re-use it. Obviously, that's a great ability to have. I'm currently planning my *Writing for the Web* (a senior-level course in the program) final paper that has to do with responsive designs. I've talked about how it is way more efficient to create one responsive design that works for mobile phones, tablets, and computers. This is sort of a similar situation here. You can create one thing of 'content' and then source it out to a PDF, e-book, or HTML file. Then, whenever you have to make a change, you can make it one time. Then, transform your file to the different formats…and you're good. So I can definitely see the benefit of that.

Talking about the final class project with DITA, which by fall of 2013 incorporated team-based work, a student said

> "I couldn't agree more that this project is helpful. Many professional/technical writers work in teams and collaborate on projects. That is why DITA is so great to use because maps and topics can be linked and re-used with others so easily.

A classmate added that

> after doing the DITA Breakfast project, this project was super easy to do. I applied everything that I learned in the first project, but on a bigger scale. I also got really into the customizations for the WebHelp and the PDF. After customizing the project it seemed like a real document that could actually be used by someone instead of just a project for a class.

These final reflections confirm that separating the various layers of abstraction involved in the process of creating technical documentation with DITA not only allows students to more easily adapt to a DITA environment but also to understand the rhetorical affordances of the technology.

Conclusions, Limitations, and Suggestions for Future Research

This section presents conclusions from the teaching case, introduces suggestions for improvement and adoption in similar professional and technical writing programs, and forecasts ideas for future

research on the production and delivery of technical documentation authored in DITA.

Conclusions *Creating User Documentation* is a difficult course to take... and to teach. From the instructor's perspective, it requires constant updates to reflect trends and preferences in industry and research in academia. Therefore, it involves keeping an eye on the proceedings and reports of conferences and meetings on both sides of the technical communication professional spectrum. From the student perspective, the assignments can be challenging but rewarding. However, the course is not universally loved, and more than once a student's reflection has implied that it looks more like a Computer Science class than an English course. A solution to this problem is to avoid overemphasizing the coding and automation side of DITA and XML and to maintain a solid grounding on writing principles and good technical communication requirements.

The course is a modest attempt to answer the call broadcasted by Andersen [1] for uniting efforts in industry and academia to facilitate the adoption and understanding of component content-management solutions. Some graduates of the program become technical communication practitioners and join the workforce with a solid knowledge of structured authoring and DITA. By introducing an abstraction-oriented computational thinking approach, the layers build over time and give students opportunities to master one level before addressing the next one, and it helps that automation comes on the DITA Open Toolkit and students do not need to learn a programming language.

A student from the fall 2012 semester summarized her experiences during the course in the following post, which was appended to the first assignment's (SOP) forum:

> Okay so this is extremely late coming, but I think it's interesting to compare where we were for this assignment to the place we are in now with DITA. I say this because we are actually using information we learned from this section and the assignments that we did for this part of the class to complete websites now. I can honestly say that I didn't think that I would be able to make half of the things that we have in class and I'm kind of impressed with myself considering how bad I am at technology (my own fault since I don't like to sit around and play with technology and I get frustrated easy with it).

By acknowledging that the skills learned in the course's first layer of abstraction are still relevant to the final layer, this student is confirming that the concept of abstraction that guided course development did, in fact, allow students to build each new skill on top of those learned in previous layers.

Limitations A major limitation of this case is the lack of information about our graduates who become technical writing practitioners. As of April 2014, we do not have a complete database documenting placement and employment for Professional Writing graduates. From informal conversations on social media, we only have three documented cases of graduates who work or worked with structured authoring in the workplace: two of them work with MadCap Flare and one has conducted work with DITA and DocBook.

Also, one semester does not allow for a full exploration of DITA specialization, including the Learning and Training Content Specialization, or an additional layer of abstraction for advanced customization with plug-ins. And budgetary limitations also complicate the adoption of more user-friendly, enterprise-level DITA editors, or even a more robust solution like Adobe FrameMaker. However, the deliverables created by students in this course reflect proper balance of rhetorical, persuasive writing, accurate technical content, and modest computing implementation. Students from *Creating User Documentation* frequently present about the process of creating their DITA assignments at the Virginia Tech undergraduate English research conference, and some have participated in related internships and research projects in their senior semester.

For possible adoption of this case in similar academic programs, a limitation could be the course sequencing at our institution that includes *Creating User Documentation*. The course's prerequisites covering fundamentals of Genre theory, and more advanced offerings on web design and content strategy, create a unique environment for our students. From a technology perspective, possible adoption of this case in similar programs could take some of the abstraction levels and apply them to topic-based authoring in a Wiki-like environment, or even on HTML for the web. DITA is not a software program, and students who learn about topics and single sourcing using DITA can apply those principles to blogs and even word-processing-based documents.

Suggestions for Future Research Reporting from a discussion about research topics that could unite industry and academia in technical communication, Benavente and Clark [40] identified a few lines of inquiry that were deemed as relevant for both sectors. These lines include reception studies, content strategy studies, and metrics/measurement studies, among others. All of those lines could connect to the process of authoring technical documentation in DITA. In addition, academia needs to conduct research to innovate the field and not just reproduce what industry is already doing. It could be interesting to continue exploring the production of DITA deliverables for small screens and mobile devices, the development of hybrid topics with multimodal content, and a combination of how-to videos with principles of structured authoring and component content management.

ACKNOWLEDGMENTS

Katherine Mawyer conducted some of the interviews quoted in this paper as part of an undergraduate research project supervised by the first author.

REFERENCES

[1] R. Andersen, "Component content management: Shaping the discourse through innovation diffusion research and reciprocity," *Tech. Commun. Quart.*, vol. 20, no. 4, pp. 384–411, 2011.
[2] K. Schengili-Roberts. (2013). What size are DITA-using firms and where is DITA going?*DITAWriter.* [Online]. Available: http://www.ditawriter.com/what-size-are-dita-using-firms-and-where-is-dita-going/?utm_medium=twitter&utm_source=twitterfeed
[3] S. O'Keefe and A. Pringle, The state of structure, 2011, Cary, NC, USA, 2011.
[4] B. J. G. McShane, "Why we should teach XML: An argument for technical acuity," in *Content Management: Bridging the Gap Between Theory and Practice*, G. Pullman and B. Gu, Eds. Amityville, NY, USA: Baywood, 2009, pp. 73–85.
[5] M. F. Eble, "Digital delivery and communication technologies: Understanding content management systems through rhetorical theory," in *Content Management: Bridging the Gap between Theory and Practice*, G. Pullman and B. Gu, Eds. Amityville, NY, USA: Baywood, 2009, pp. 87–102.
[6] J. M. Wing, "Computational thinking," *Commun. ACM*, vol. 49, no. 3, pp. 33–35, 2006.
[7] J. M. Wing, "Computational thinking and thinking about computing.," *Philos. Trans. A. Math. Phys. Eng. Sci.*, vol. 366, no. 1881, pp. 3717–3725, Oct. 2008.
[8] College Board, *Computer Science: Principles Computational Thinking Practices Big Ideas, Key Concepts, Supporting Concepts*, 2012.
[9] L. Bitzer, "The rhetorical situation," *Philos. Rhetor.*, vol. 1, no. 1, pp. 1–14, 1968.
[10] C. R. Miller, "Genre as social action," *Q. J. Speech*, vol. 70, no. 2, pp. 151–167, May 1984.
[11] D. R. Russell, "Rethinking genre in school and society," *Writ. Commun.*, vol. 14, no. 4, pp. 504–554, 1997.
[12] J. Hackos, *Content Management for Dynamic Web Delivery*. Hoboken, NJ, USA: Wiley, 2002.
[13] C. Robidoux, "Rhetorically structured content: Developing a collaborative single-sourcing curriculum," *Tech. Commun. Quart.*, vol. 17, no. 1, pp. 110–135, 2008.
[14] A. J. Devitt, *Writing Genres*. Carbondale, IL, USA: Southern Illinois University Press, 2004.
[15] C. Bazerman, "Systems of genres and the enactment of social intentions," in *Genre and the New Rhetoric*, A. Freedman and P. Medway, Eds. New York, USA: Taylor & Francis, 1995, pp. 79–104.
[16] D. Day, M. Priestley, and D. Schell. (2005). Introduction to the Darwin Information Typing Architecture: Toward portable technical information. IBM developerWorks. [Online]. Available: https://www.ibm.com/developerworks/library/x-dita1/
[17] T. Self, *The DITA Style Guide: Best Practices for Authors*. Research Triangle Park, NC, USA: Scriptorium Publishing, 2011.
[18] E. Kimber, *DITA for Practioners Volume 1: Architecture and Technology*. Laguna Hills, CA, USA: XML Press, 2012.
[19] J. Hackos, *Introduction to DITA,* 2nd ed. Denver, CO, USA: Comtech Services, 2011.
[20] L. W. White, *DITA for Print: A DITA Open Toolkit Workbook*. Laguna Hills, CA, USA: XML Press, 2013.
[21] J. Vasquez, *Practical DITA*. Raleigh, NC, USA: Write Spirit, 2009.
[22] A. Rockley, S. Manning, and C. Cooper, *DITA 101: Fundamentals of DITA for Authors and Managers*. Schomberg, ON, Canada: Rockley Group, 2009.
[23] L. Bellamy, M. Carey, and J. Schlotfeldt, *DITA Best Practices: A Roadmap for Writing, Editing, Architecting in DITA*. Upper Saddle River, NJ, USA: IBM Press, 2012.
[24] E. Fox, C. Evia, W. Fan, S. Sheetz, and C. Zobel. (2007). Living In the KnowlEdge Society (LIKES),*NSF Award Abstract.* [Online]. Available: http://www.nsf.gov/awardsearch/showAward?AWD_ID=0722259&HistoricalAwards=false
[25] A. V. Aho, "Computation and computational thinking," *Comput. J.*, vol. 55, no. 7, pp. 832–835, 2012.
[26] J. Bennedsen and M. E. Caspersen, "Abstraction ability as an indicator of success for learning computing science?," in *Proc. 4th Int. Workshop Comput. Educ. Res.*, 2008, pp. 15–26.

[27] P. Frorer, M. Manes, and O. Hazzan, "Revealing the faces of abstraction," *Int. J. Comput. Math. Learn.*, vol. 2, no. 3, pp. 217–228, 1997.
[28] B. Haberman, "High-school students' attitudes regarding procedural abstraction," *Educ. Inf. Technol.*, vol. 9, no. 2, pp. 131–145, 2004.
[29] H. C. Webb and M. B. Rosson, "Using scaffolded examples to teach computational thinking concepts," in *Proc. 44th ACM Tech. Symp. Comput. Sci. Educ.*, 2013, pp. 95–100.
[30] S. I. Hayakawa and A. R. Hayakawa, *Language in Thought and Action,* 5th ed. Orlando, FL, USA: Harcourt, 1990.
[31] R. Andersen, S. Benavente, D. Clark, W. Hart-davidson, and C. Rude, "Open research questions for academics and industry professionals?: Results of a survey," *Commun. Des. Quart.*, vol. 1, no. 4, pp. 42–49, 2013.
[32] C. Evia, M. Stewart, T. Lockridge, S. Scerbo, and M. Perez-Quiñones, "Structured authoring meets technical comics in techcommix," in *Proc. 30th ACM Int. Conf. Design Commun.*, 2012, pp. 353–354.
[33] C. Evia, S. Healy, and T. Lockridge, "Evaluating a workflow for authoring multimodal DITA," in *Proc. 31st ACM Int. Conf. Design Commun.*, 2013, p. 185.
[34] M. Bremer, *The User Manual Manual: How to Research, Write, Test, Edit and Produce a Software Manual.* Concord, CA, USA: UnTechnical Press, 1999.
[35] M. Baker, *Every Page is Page One: Topic-based Writing for Technical Communication and the Web.* Laguna Hills, CA, USA: XML Press, 2013.
[36] M. Carey, D. Longo, M. McFadden Lanyi, E. Radzinski, S. Rouiller, and E. Wilde, *Technical Writing 101: A Real-World Guide to Planning and Writing Technical Content,* 3rd ed. Research Triangle Park, NC, USA: Scriptorium Publishing, 2009.
[37] A. Pringle and S. O'Keefe, *Developing Quality Technical Information: A Handbook for Writers and Editors,* 3rd ed. Upper Saddle River, NJ, USA: IBM Press, 2014.
[38] R. J. Glushko and T. McGrath, *Document Engineering: Analyzing and Designing Documents for Business Informatics and Web Services.* Cambridge, MA, USA: MIT Press, 2005.
[39] A. Rockley, "The impact of single sourcing and technology," *Tech. Commun.*, vol. 48, no. 2, pp. 189–193, 2001.
[40] S. Benavente and D. Clark. (2013). Using research to examine beliefs and habits,*CIDM Inf. Manage. News.* [Online]. Available: http://www.infomanagementcenter.com/enewsletter/2013/201303/second.htm

Carlos Evia is an associate professor of Technical Communication in the Department of English at Virginia Polytechnic Institute and State University, Blacksburg, VA, USA, where he also directs the program in Professional and Technical Writing. He conducts research on multicultural user assistance for the Virginia Tech Centers Human–Computer Interaction and Innovation in Construction Safety and Health. Prof. Evia is a member of the DITA Technical Committee with the Organization for the Advancement of Structured Information Standards.

Matthew R. Sharp is an assistant professor of Communication in the Humanities and Communication Department at Embry-Riddle Aeronautical University, Daytona Beach, FL, USA. With a background in professional writing, web development, fundraising, and enrollment management, his research analyzes organizational activity systems and their mediating technologies from both cultural and rhetorical studies perspectives.

Manuel A. Pérez-Quiñones is an Associate Dean of the College of Computing and Informatics and Professor in the Department of Software and Information Systems at the University of North Carolina at Charlotte, Charlotte, NC, USA. His research interests include personal information management; human–computer interaction; user interface software; as well as educational, cultural, and diversity issues in computing. He is a member of the Coalition to Diversify Computing.