

Yuliya Jalkanen

CASE STUDY ON XML-AUTHORING SOFTWARE

CASE STUDY ON XML-AUTHORING SOFTWARE

Yuliya Jalkanen
Bachelor's Thesis
Spring 2018
Business and Information Technology
Oulu University of Applied Sciences

ABSTRACT

Oulu University of Applied Sciences
Business Information Technology

Author(s): Yuliya Jalkanen

Title of Bachelor's thesis: Case study on XML-Authoring Software

Supervisor(s): Teppo Räisänen

Term and year of completion: Spring 2018

Number of pages: 40

During the past few decades there has been a significant change in the ways that the technical documentation is perceived and created. This change is a result of the current globalized environment where communications and technologies develop and spread as fast as ever, and where the documentation often needs to be mass-produced, customized, tailored and delivered on demand around the globe. Current technical content creation and management approaches are often focused on dynamic, medium independent documentation that can be effectively maintained, continually edited and stay as accurate, clear and structured as possible.

Seen as a part of product development and as an asset, the technical documentation must satisfy the raising expectations and address the current trends in the field of content creation. To explore the modern ways of technical content creation, it is necessary to examine which aspects of content creation have changed and what technologies, approaches and philosophies are trending in the field of content creation now.

The purpose of this research is to examine one of the modern solutions for dynamic technical content creation, a combination of two PTC Arbortext Family products, PTC Arbortext® Editor™ and PTC Arbortext® Styler™.

This case study uses applied research methods, with main aim for practical application of the findings, specifically, to create a demo package that showcases main features of the chosen solutions. To achieve that aim, a theoretical framework had been established and examined. The main concepts that formed the basis for the research include current trends in technical documentation production, information architecture philosophy, advantages of structured information concepts such as descriptive markup, and Darwin Information Type Architecture (DITA) approach to technical content creation.

Results of the research suggest that the practical outcomes of the case study support the concepts covered in the theoretical framework. The demo package contents address the goals that were set in the beginning of the research. As the concepts examined in this case study are constantly developing, further research on the topic is required.

Keywords: Structured information, information architecture, descriptive markup, DITA

CONTENTS

1	INTRODUCTION	5
2	THEORETICAL PRINCIPLES	7
2.1	Current trends in technical content creation	7
2.1.1	Key notions of structured information.....	10
2.1.2	Information architecture in technical documentation	11
2.1.3	Markup language in technical documentation	11
2.1.4	HTML and XML as descriptive markup languages in technical writing.....	14
2.2	Darwin Information Type Architecture (DITA) in technical documentation.....	16
2.2.1	What is DITA and why do we need it?	17
2.2.2	Core concepts of DITA.....	19
3	CASE PTC ARBORTEXT EDITOR &STYLER	23
3.1	Requirements specification	23
3.2	Technologies and tools used.....	23
3.3	Implementation.....	26
3.4	Results	27
4	FINDINGS AND DISCUSSION.....	36
	REFERENCES	38

1 INTRODUCTION

The amount of technical documentation that is being processed globally by the industry is rising steadily. It is impossible to picture a company involved into any level of production that is not using some form of technical documentation; moreover, such company is likely to be producing documentation in one way or another. Such factors like Constant development of products and services prompt the reuse of technical documentation. For many companies, aspects like mass-production and globalization of markets actualize not only the localization needs but can also include product or service customization and tailoring, adding another factor to underline the demand for reuse and tailoring features in technical content creation.

Current technical content creation and management approaches are often focused on documentation that is medium-independent, sometimes also referred to as “smart” or “dynamic”. In general, those terms refer to data that can be continually edited, expanded, adjusted or updated through revisions to sustain accuracy, relevance and be delivered on demand. The industry has an ever-growing need for holistic, adaptable and customizable documentation solutions, and a variety of tools has been emerging in the market.

This case study spotlights one of such tools, a combination of two PTC Arbortext Family products, PTC Arbortext® Editor™ and PTC Arbortext® Styler™. The main aim of this case study was to develop an Arbortext Editor demo package for Econocap Software. The company provides PTC software products and expert services that offer solutions for challenges in product development, product information management and process digitization.

The objective of the case study was to create a demo that showcases key features of the PTC Arbortext Editor & Styler. These features address such current trends in technical information content creation as reusability of data, media-neutral product and service information handling as well as offering ways to process documentation for use in such currently advancing technologies as augmented reality. While this tool can be used for diverse types of information, the study is focused on creating and managing technical documentation, since this segment is most often addressed.

This case study uses applied research methods that focus on delivering practical application of the findings. The demo needed to be used as such, as well as to be easily adjusted, if necessary, to reflect customers' needs. An analysis based on the initial task requirements was performed to identify the contents of the demo deliverables. The goals were achieved by creating a package that contains several Extensible Markup Language (XML) sources with a varying structure and two different stylesheets to be used for formatting of the XML sources. The content and possible combinations of these sources is described later in this thesis.

This thesis is divided into four major parts. The first part has presented the background to the study. The second part covers theoretical framework that has been used as a basis for this case study, while the third part comprises the requirements for the demo creation and provides an overview of the tools that were used. The final part focuses on the findings that originated in process and possible improvement ideas for future.

2 THEORETICAL PRINCIPLES

Theoretical principles that establish the basis for this case study are examined in this chapter. The chapter starts with overview of the current trends in the field of technical content creation. Further on, the concepts covered in this chapter include structured information approach to technical writing, text markup types and Darwin Information Type Architecture.

2.1 Current trends in technical content creation

Two decades ago, the definition of a “document” included something printed - a technical manual, a book, or a promotional brochure. Today, the technical information is not confined to a specific publication medium. It could be delivered in any form - a printed manual, a brochure in a PDF format, a video with series of animated steps, an online portal with tutorials, knowledge base and articles that are constantly updated, and even in a form of augmented reality instructions that are delivered to the user live on demand. For the ease of access purposes, often the deliverables tend to include similar information that is published on different media simultaneously. In that case, content reuse allows medium-neutrality, accessibility, accuracy and contingency of delivered information on a much shorter time scale than previously.

For the past few decades, technology has been developing as fast as ever, bringing such terms as augmented and virtual reality (AR and VR), Internet of Things (IoT) and, generally, “smart-” everything to the routine of the everyday life. Applied in technical documentation field, these technologies have the potential to fundamentally change the way the content is experienced, improve the quality of information delivered and make the documentation more clear, accessible and detailed.

Augmented Reality (AR) in technical content creation is quite recent, but very promising trend of superimposing additional information to the real world in the form of virtual objects, therefore augmenting the objects or the environment of the real world with the information from the digital source (Kerry 2018, cited 01.06.2018). This forms an interactive system that can be tailored to address specific expectations. One of the examples of adopting AR in modern technical documentation could be instructing a user during difficult repairs by providing a device with built-in

camera that can process images and record the movements of the user for forwarding them to processing software. The software could identify the specific objects and show the specific instructions to the user in the live format.

But how did we get there? The *digitization* of previously printed information became popular just a while ago, prompted by the development of scanners. This approach was relatively less time-consuming way of processing and sharing the data, however, it was not innovative: scanning would mean simple storing a copy of an original medium. There was no way to include any explicit structural information to a scanned part of text or to an image. The modern ways of processing the information had overcome this challenge by adopting the structured information approach to content creation, where the *content* is separated from the *form*, thus creating more navigable, accessible and dynamic data.

Structured information approach to documentation is incorporated by many global actors, one of which is Schneider Electric. Their North America product catalog, the 1,000-page Digest was initially developed as print-first document which is updated once every 3 years. The popular demand to digitalization was successfully met by adopting PTC's authoring and content management tools that are based on Darwin Information Type Architecture (DITA) principles. The use of DITA not only reduced the printing and production costs, but also boosted product's accessibility, at the same time contributing to the sustainable development by reducing the use of paper media. (PTC 2018b, cited 28.05.2018).

In the technical documentation field, the content should be as consistent, comprehensive, clear, and accurate as possible. It is also meant to be easily updated, transferred and otherwise maintained according to changes in the product, disregard the location of the authoring team. To provide good content, managing the authoring tools in a collaborative environment requires a carefully planned managerial approach, such as Information Lifecycle Management (ILM). Gartner IT Glossary (2018, cited 08.05.2018) defines ILM as the following: "(..) an approach to data and storage management that recognizes that the value of information changes over time and that it must be managed accordingly". In contrast to older systems, ILM enables more complex criteria for information management (Tech Target 2005, cited 08.05.2018), and supports another trend in the technical documentation field: a holistic approach to documentation, where documentation is seen as product design. Technical content creation becomes a part of the product development process.

The technology made it possible to add variation to the deliverables and to broaden the definition of collaborative environment. However, in its core, the *process* of creating that data has not changed, rather the *approach* towards content creation and the *tools* that are used for that creation have been re-defined. This can be illustrated by examining the Information Life Cycle (ILC) presented in Figure 1:

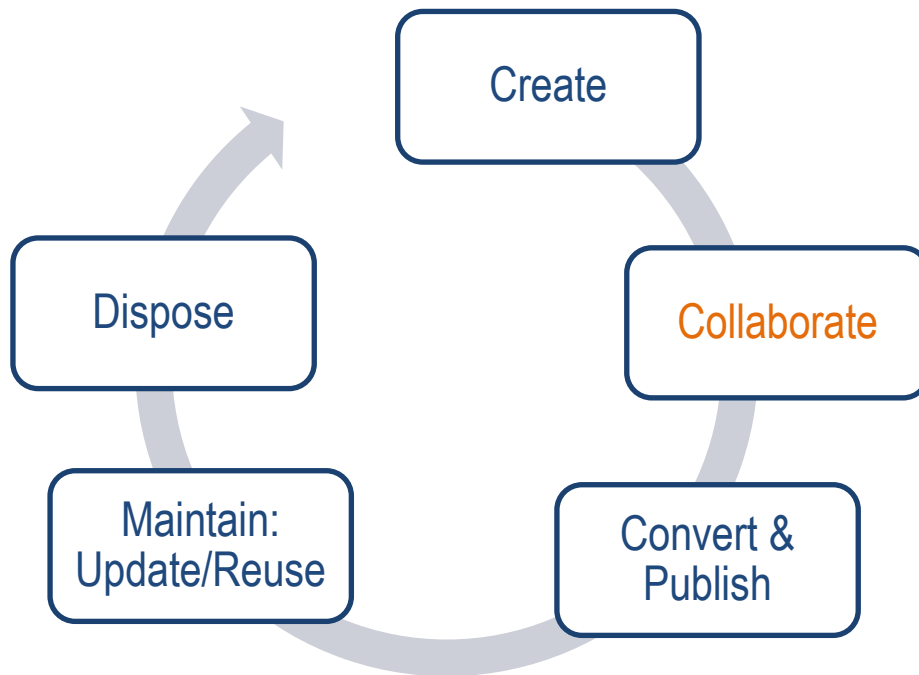


Figure 1: Information Life Cycle

The definition of ILC steps vary depending on the approach towards information handling and industry needs, but in general, these steps can be described as the following sequence: the content is authored or processed collaboratively, based on gathered knowledge (these 2 steps can be happening simultaneously); converted to standardized format and published; maintained (updated or reused) or disposed.

So, clearly, modern documentation is moving away from PDF and print formats and can be accessed and delivered on demand globally. But globalization not only determines the demand for distribution of documentation – it also influences the way the content is created. The team of technical writers in one company can be allocated around the globe and providing the right tools to help the team of cooperating writers to be self-sustained is another one of the trending topics.

The modern ways of processing the information allow industry to use the advantages of more navigable, accessible and dynamic information by adopting structured content creation. In the following chapters the view of technical documentation as structured information and the current approaches towards handling the structured content in the technical field will be examined.

2.1.1 Key notions of structured information

Structured information is information that has been analyzed. During the analysis, the information is divided into smaller logical components. These components can be further on divided into smaller components, and so on. The components, or parts of a structured document are identified and are accessible for further processing. (DeRose 1995, cited on 05.05.2018.)

Such analysis can be done in numerous ways, as a variety of analysis models exist. Finding a proper analysis model is important, since information division, organization and accessibility play a crucial role in the usefulness and utility potential of the information. The parts of a structured document can be identified based on the goals that this division pursues - whether to represent the form of some data, some meta- information of that data, or the content of the data itself. (ibid.)

Form and *content* are the cornerstones of structured information. On paper, the form and content are bound in some way. We identify the elements of the document based on our common knowledge of what document structure should be. For example, a reader will instantly distinguish between a title and an emphasis, even if they are placed in one and the same page and the formatting is matching. For a computer, distinguishing between those two will be impossible, unless the elements have some other identification of their nature. Therefore, the components of structured information should be labelled according to their purpose, to ensure that the information will be understandable to authors/readers, but also processed by computer systems.

The structure of any given document can be defined by identifying its components. However, the definition of structure alone will not guarantee that the document will be properly “assembled” by authors. To develop a uniformity of the documentation, the definition of the names, order and relationships of the elements should be developed. This creates the consistency, which enables the information to be processed automatically by computer software.

2.1.2 Information architecture in technical documentation

Information architecture is another relevant concept that plays a significant role in creation of a good technical documentation content. The Information Architecture Institute (2013, cited 28.05.2018) defines information architecture as an interdisciplinary term, with the origins in “library science, cognitive psychology, semiotics, cybernetics, discrete mathematics, and of course, architecture itself”. The definitions of the term are, therefore, somewhat diverse in different branches of information technology. For the purposes of this case study, the following definition of information architecture will be used: “a practice focused on bringing principles of design and architecture to the digital landscape” (ibid).

One of the pioneers in that field, Richard Saul Wurman (1997, 16), referred to the Information Architect as someone who creates a “systemic, structural, and orderly principles to make something work — the thoughtful making of either artifact, or idea, or policy that informs because it is clear”. In other words, Information architecture principles are broad, but in the core, they are based on the practice of organizing the information being easily usable and findable, presenting the best user experience.

We have already defined that structured information is well-fitted for production of technical documentation. It forms a basis for segmenting and storing information and engaging the content reuse technologies as well as enabling the output to multiple formats and devices.

But what tools to use and how to write in a structured way that matches the principles of information architecture? Ideally, structured writing provides a modular, topic-based content. The documentation should be well-structured, easy to write, read and use, as well as to be updated when necessary. Writing time should be decreased due to content reusability, while the readability and user-comprehension should be improved. There are technologies that employ structured approach towards technical content creation, for example, markup languages.

2.1.3 Markup language in technical documentation

Visual markup of the page has been in use since the initial stages of readable content production, including, for example, handwritten books. The visual markup was not merely decorative, but also

accommodated complex reading practices to gather more meaning or underline significant words. With time, the markup in printed texts became more and more formalized, gaining a prominent level of codification. (Flanders 2007, cited 18.05.2018.) In the digital world, however, the markup system has quite divergent functions. To research those functions, we need to examine the definition of markup and its types.

Miriam-Webster offers the following definition (2018, cited 08.05.2018): "Markup language is a system for marking or tagging a document that indicates its logical structure (such as paragraphs) and gives instructions for its layout on the page especially for electronic transmission and display".

There are several types of markup, ranging from physical to logical: punctuational, presentational, procedural and descriptive. Punctuational markup includes punctuation means (spaces, commas, dashes, semicolons, etc.). Presentational markup is based on binary codes embedded within document text and is used by traditional word-processing systems; it includes horizontal and vertical spacing such as page breaks, double line spacing, and indentation. Procedural markup is embedded in text to provide instructions for software on how to process the text to generate presentational and punctuational markup. Finally, descriptive markup is used to label the parts of the document to define a logical structure, and is often referred to as semantic, logical or conceptual. It's designed to be understandable by computers and humans. (Coombs, Renear & DeRose 1987, 935-936.)

The functions of these markup types sometimes overlap. For example, a list of words that is separated by commas is using punctuational markup being represented as a list of bullet-points by presentational markup (see *ibid*, 934).

Current trends in the field of technical documentation often focus on the advantages of using descriptive markup, as opposed to procedural. While procedural markup is used by most of word processing systems to describe *how* to process text and images to prepare the document for presentation (usually for publishing on paper), descriptive markup describes *what* each element in a document is, therefore, serves as a base for the structure of a document.

The following is a comparison of same text handled procedurally by troff, a Unix document processing system, and descriptively by XML (Flanders 2007, cited 18.05.2018):

- troff says: "center this text",

- XML says: "just FYI, this text is a (centered) heading".

As mentioned before, in the digital world one digital medium is very often used to represent materials from other media. On practice, assigning a specific format (e.g. Times New Roman, 14pt, italic) to certain elements of text (e.g. headings) to visually distinguish between them will work within one system, but transferring that code to other media might be challenging. Other drawbacks of procedural markup would include inability of changing the format of all headings at once and inability to automatically generate table of contents. Descriptive markup, on the other hand, will provide information about text's structure and function; and once the information about the logical role of every element is identified, the presentational markup can be added to these logical elements by associating formatting parameters with the structural elements (Kirsanov 1997, cited 05.05.2018).

To sum up, descriptive markup does not provide instructions to a system how to render the text, but rather identifies the structure and function of the document and its elements: this information can be used by a variety of systems to be processed. This is the biggest advantage of the descriptive markup philosophy, which foundational assumptions are (Flanders 2007, cited 18.05.2018):

- Presentation expresses structure and function,
- Markup should identify structure (primary),
- Stylesheets produce presentation (secondary).

With this philosophy applied to technical documentation, all business information "lives" in one file; it's content-focused; the ways that this information will be formatted can vary depending on the medium; content, structure and style are not blended into one file; because the elements are identified, new documents with varying structure can be created based on one source by reusing the existing elements; the source elements can be proofread only once and reuse ensures high level of accuracy.

In the next subchapter we will take a closer look at two instances of popular descriptive markup languages, specifically, HTML and XML.

2.1.4 HTML and XML as descriptive markup languages in technical writing

Both being markup languages and developed by World Wide Web consortium, HTML and XML functions and uses in technical documentation vary. They do share some similarities; however, XML is not a substitute for HTML. A combination of these languages makes it possible to achieve different results, for example, by using XML for storing data inside the HTML documents. (XML Objective 2013, cited 18.05.2018.)

HyperText Markup Language, or HTML, was initially developed as a relatively easy to learn computer language for website creation, understandable both for humans and the machines. In its definition, "HyperText" stands for the method of non-linear navigating around the web by using hyperlinks; "Markup" defines the functionality of tags and "Language" means that it has code-words and syntax. (Shannon 2018, cited 28.05.2018.) Being the first Internet-based language, HTML is constantly developing and undergoing revision. Its most recent version, HTML5, allows defining the way videos, images, and text look. HTML remains a core markup for web pages, or any content that is displayed in a browser. (Wodehouse 2018b, cited 01.06.2018.)

Extensible Markup Language, or XML, is a computer language that derived from more complex SGML (Standard Generalized Markup Language). As its name suggests, SGML is a standard for encoding documents into an electronic format (The World Wide Web Consortium 2018, cited 01.06.2018). In comparison with SGML, XML contains more limited set of features to provide greater accessibility and fit the intended use. SGML, and consequently XML are based on industry standards.

Main difference between HTML and XML include the following:

- HTML is about displaying data, while XML is about describing information;
- XML is extensible (new tags may be defined by author);
- XML "stores" information content that can be used in various media, including publishing structured information to HTML for the Web.

In other words, the functionalities of XML in the technical documentation are vaster since it describes the content of information, as opposed to presentation-oriented HTML (Perugini 2003, cited 28.05.2018). In XML, one can customize the names of tags, as well as a position of the tag in relation to other tags, by incorporating a Defining the Document Type principle (DDT), which is

used in XML to formally describe the data (XML Objective 2013, cited 18.05.2018). Another use of XML is storing data in files or databases or to serve as a standardized format to exchange information. Converting XML data reduces the complexity of information stored and allows the data to be read basically by any kind of application. Finally, XML is used as a core building block of AJAX (Asynchronous JavaScript + XML), a technology that allows updating separate parts of a web page without reloading the entire page, by asynchronously exchanging small amounts of data with a server (Wodehouse 2018a, cited 01.06.2018).

HTML and XML share certain similarities. They are both human and machine readable; the elements of both languages have a tag attribute at the beginning; XML is a resembling language to HTML, so it can be easily used by people familiar with HTML. Tables 1 and 2 provide code samples that illustrate the differences and similarities of these two languages.

TABLE 1: An HTML code sample.

HTML code sample:	HTML code sample rendered in a browser:
<pre> <html> <head><title>Books in inventory</title></head> <body> <h2>Books</h2> <hr> The Adventures of Sherlock Holmes, A. Conan Doyle, 1892
 Winnie-the-Pooh, A. A. Milne, 1926
 Alice in Wonderland, Lewis Carroll, 1866
 </body> </html> </pre>	<p>Books</p> <hr/> <p><i>The Adventures of Sherlock Holmes</i>, A. Conan Doyle, 1892 <i>Winnie-the-Pooh</i>, A. A. Milne, 1926 <i>Alice in Wonderland</i>, Lewis Carroll, 1866</p>

As illustrated in Table 1, the HTML sample is aimed to present the information to a reader to view in a web browser, while for the processing machine tagging Lewis Carroll in bold does not provide means to determine that this is the author of the book.

TABLE 2: An XML code sample.

XML code sample:	XML code sample rendered in a browser:
<pre> <books> <book> <title>The Adventures of Sherlock Holmes</title> <author>A. Conan Doyle</author> <year>1892</year> </book> <book> <title>Winnie-the-Pooh</title> <author>A. A. Milne</author> <year>18926</year> </book> <book> <title>Alice in Wonderland</title> <author>Lewis Carroll</author> <year>1866</year> </book> </books> </pre>	<pre> <?xml version="1.0"?> - <books> - <book> <title>The Adventures of Sherlock Holmes</title> <author>A. Conan Doyle</author> <year>1892</year> </book> - <book> <title>Winnie-the-Pooh</title> <author>A. A. Milne</author> <year>18926</year> </book> - <book> <title>Alice in Wonderland</title> <author>Lewis Carroll</author> <year>1866</year> </book> </books> </pre>

In Table 2, the XML sample presents the tag `<author>`, which is self-descriptive for the reader, and is comprehensible for the parsing software, if the description of the language “*grammar*” (a document type definition (.dtd), or an XML Schema) is provided along the code.

Opposite to the HTML example, XML sample rendering does not provide any formatting information. This feature assures that the same information can be presented in different formats without replicating any data.

2.2 Darwin Information Type Architecture (DITA) in technical documentation

The knowledge that is stored in company’s technical documentation certainly has value, therefore, it can be treated as an asset. At the same time, technical documentation produced by a company is often versatile, therefore it is important that the information is appropriately standardized. There are several standards in the industry, including a series of standards published by International Standardization Organization (ISO) related to technical product documentations. Source data standards include DITA, DocBook, and S1000D. In this study, we will focus on DITA.

2.2.1 What is DITA and why do we need it?

DITA is an open standard, XML-based data model for authoring and publishing, developed to address the streamlined content creation process, separation of content from formatting and introducing simpler ways of publishing. IBM, which developed DITA initially, offers the following definition (Day, Priestley& Hargis 2005, cited 01.06.2018):

“The Darwin Information Typing Architecture (DITA) is an XML-based, end-to-end architecture for authoring, producing, and delivering technical information. This architecture 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”

As reflected in the concept’s name, DITA uses the principles of specialization and inheritance that resemble the principle of variation in species proposed by Charles Darwin. Information Typing in the name stands to illustrate that DITA capitalizes on the semantics of topics and of content, and Architecture refers to development of new applications, and specializations of new types for information (ibid, cited 01.06.2018).

IBM has donated DITA as an open source architecture, and it is now maintained by Organization for the Advancement of Structured Information Standards (OASIS), a “nonprofit consortium that drives the development, convergence and adoption of open standards for the global information society” (OASIS 2018, cited 01.06.2018).

OASIS is closely linked to above-mentioned predecessor of XML: SGML. Originally, OASIS was founded as a consortium of both vendors and users of SGML to develop guidelines that accommodate the interoperability among products that support SGML. This aim was reflected in consortium’s previous name, “SGML Open”, which was changed in 1998 to match the broader scope of technical work. (ibid.) As of now, the latest DITA version is 1.3, approved by OASIS DITA Technical Committee on 17th of December 2015.

So why do we need DITA standards in technical documentation? To answer this question, let us look at the content creation ways that do not incorporate DITA principles. As mentioned before, a lot of content is being created in a variety of authoring tools: Word applications, email, WordPress, HTML, InDesign, FrameMaker, or any other format. As a result, such content is bound to the limited number of formats, it cannot be efficiently reused or repurposed, it’s time-consuming to tailor, and

overall is costly. DITA principles are oriented towards managing content as an asset. (Samuels 2014, cited 01.06.2018.)

DITA leverages structured information approach and uses descriptive markup language, specifically, XML, to add versatility to the content, making it manageable, portable, intelligent and media-neutral. The concepts of reuse and repurpose received a new meaning in DITA, rendering the Copy Paste between the files obsolete. In DITA, the content is free from the form or format, and can be published to a variety outputs: PDF, HTML, RTF, PowerPoint, mobile devices, ePub, etc. Another innovative feature of DITA is the increased capacity of collaboration between the content creators: content can be authored by several authors with high consistency and quality. And finally, DITA allows reducing translation efforts by up to 80% (Samuels 2014, cited 01.06.2018).

In technical documentation, DITA has already become a traditional approach. Complex technical content (manuals, user guides, online portals, help pages, etc.) demand high accuracy but also reuse and repurposing of the information, which is provided by DITA. With a multitude of authors contributing to the content, detailed illustrations, hundreds of tables, thousands of pages and a variety of updates, creating a content that is accurate and can be published to different outputs at once (e.g. HTML, PDF, RTF) is very challenging. DITA principles address those challenges by taking reuse to a new level, serving as a tool for, as it is mentioned in OASIS's DITA purpose definition, "(..) advancing a document creation and management specification that builds content reuse into the authoring process" (OASIS 2017, cited 01.06.2018).

Figure 2 illustrates the flow of content creation and maintenance in accordance to DITA principles. The content is authored in XML, organized into hierarchies, formatted and processed (published or delivered online). The final deliverable can be later easily updated based on latest requirements. This flow corresponds to Information Lifecycle presented previously in Figure 1, as the information is created, published, updated and maintained, possibly via collaboration.

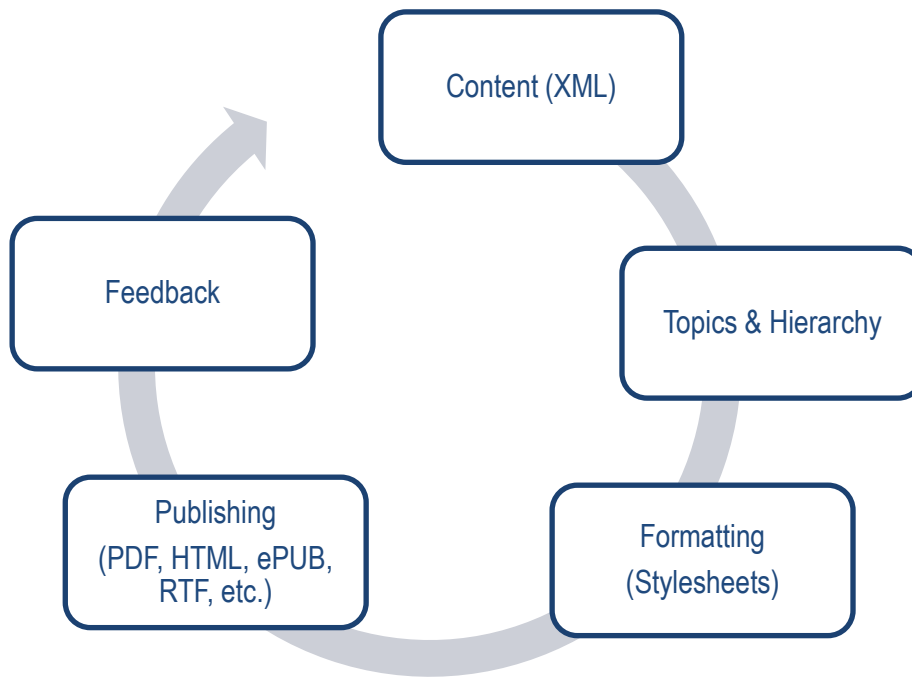


Figure 2. DITA illustrated

To sum up, DITA not only streamlines the content creation process and increases the quality of the content by incorporating the standards, but also allows to use efficiently the “asset” aspect of the content, by leveraging it in a variety of ways, including reuse, multiple formats support, efficient translations, etc.). In other words, DITA adds a level of extra efficiency to the content, which tends to save time, effort and money, and decrease repetitiveness in authoring process.

2.2.2 Core concepts of DITA

The Information Typing part of DITA definition is a key verge of DITA. As it is mentioned above, DITA uses XML to form the structure underneath the content. The information is made clear and concise by separating the content into logical divisions. Every unit of the content is stored within the tags, and DITA uses a set of rules that define what tags are allowed within each unit.

The content of these units is based on the type of information that the unit holds. The two main units are called topics and maps, which can be extended into new structural types and domains using specialization. For example, a *procedure* is a specialization of a *topic*, and the content of a procedure will only include information that is directly related to the procedure. Any information that is not directly related to that procedure should be put into a separate topic (or topics specialization).

This practice is called *topic-based authoring*, and it improves the value of content by sorting the information to designated places, where it can be easily found. (Samuels 2014, cited 01.06.2018.)

Furthermore, not only topic types are predefined by DITA. The tags or elements that should be used in each topic, as well as their placement within the topic are laid out, and are meant to be validated against a data model. This would be a well-structured, but very limited approach, if DITA was not incorporating extensibility. Use of socialization provides content creator a way to define own rules to meet the needs of the content creation process.

The structural components of DITA include the two main units, topics and maps. DITA topics are the basic units of DITA content. Each topic should be organized around a single subject. A topic is short enough to be specific to a single subject, but the content should be sufficient to make sense as a single unit. Topics that contain distinct kinds of information (concepts, tasks, references, etc.) are categorized as different information types. Topic's basic structure remains the same disregard the information type: it includes a title, a short description, a prolog, a body and related links. (OASIS 2005, 7-9, cited 01.06.2018.) The structure of the topic is illustrated in Figure 3:

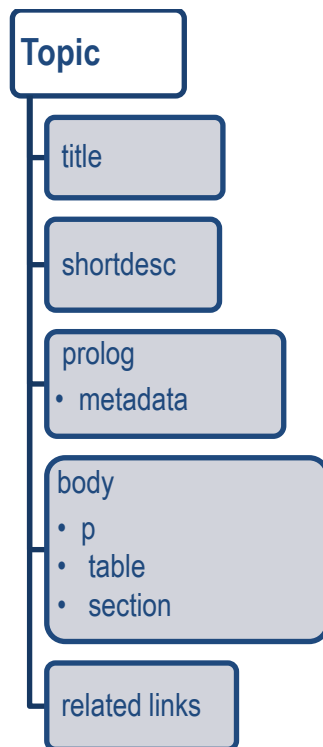


Figure 3. DITA Topic structure.

DITA maps are documents that organize topics by referencing them into hierarchies, tables, and groups for specific deliverables. DITA maps also generate the navigation files and links to the nested topics to indicate the relationships among the topics. For example, a simple catalog can be created as a DITA map that contains several nested maps, where each map would include interrelated topics, each concentrated on a single subject. DITA maps leverage reuse of content for multiple contexts and can be used by information architects and content creators to plan and deliver content. (OASIS 2005, 13-18, cited 01.06.2018.) Figure 4 illustrates a basic DITA map structure.

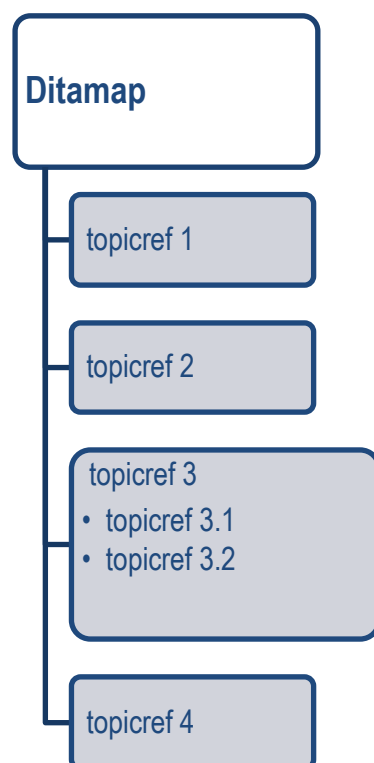


Figure 4. DITA Map structure

Besides the maps and topics, DITA's core building blocks include common metadata elements and common attributes. Metadata elements are available for both DITA maps and DITA topics. Furthermore, metadata elements are the same for both unit types, so that metadata assigned to a topic can be supplemented or overridden by metadata of the DITA map. OASIS defines different metadata element types that serve a variety of purposes, for example (OASIS 2005, 18-20, cited 01.06.2018):

- Provide standard information about a topic as publication
- Provide basis for managing the publication process for topics.
- Qualify the topic for processing.

- Specify the topic's properties.

Metadata for a map can include a copyright information for the publication, a version of the document, the author(s), the publisher, as well as other information that can be published or kept in the source code for further references example (OASIS 2005, 18-20, cited 01.06.2018).

Attributes contain data related to a specific element. Most of DITA elements have the following set of attributes in common: identity attributes, content reference attributes, metadata attributes, miscellaneous attributes, architectural attributes and, finally, conditional processing attributes. The purpose of these attributes groups is the following: identity attributes are used to identify the content, content reference attributes store references to other elements, metadata attributes express qualifications of the content, architectural attributes provide type information to processors, conditional processing attributes filter or flag the information based on processing-time criteria, and finally miscellaneous attributes identify miscellaneous data such as the language of an element or a classifying label.(OASIS 2005, 20-24, cited 01.06.2018).

The use of metadata and attributes allows content to be tailored and reused in efficient ways. For example, one of the possible uses for metadata attributes is to specify content properties for processing the content during publication. If a publication content for a product in different markets needs to cover country-specific cautions, the topic that cover the irrelevant information can be excluded out of publication by assigning a value to metadata element that would indicate that this topic does not need to be published. Another example is localization and translation attributes that are used to identify the language of the content, to determine whether the element needs translation, and can determine the direction of content rendering (for example, right-to-left for Hebrew or Arabic language outputs).

3 CASE PTC ARBORTEXT EDITOR &STYLER

This chapter reports on the process of developing the demo materials to address the goals of this study based on the requirement specification, as well as an overview of tools that are in the spotlight of this study.

3.1 Requirements specification

The objective of the case study was to create a demo for marketing purposes that showcases key features of the PTC Arbortext Editor with Styler for technical content creation and underline the benefits of structured information approach. The features should highlight media-neutral product and service information handling and reusability of data. For future use, the source XML-based documentation should be possible to tailor for use with augmented reality tools, which means that the demo should focus on media-neutrality of the source. The following requirements for the demo were identified:

1. To visualize the media-neutral approach,
2. To illustrate the uniformity and standardization,
3. To illustrate reuse of component-based content.

The development of demo package content proceeded based on these requirements.

3.2 Technologies and tools used

A combination of two tools from Arbortext family has been applied to create the demo. PTC Arbortext Editor, an XML-authoring software for DITA-based authoring was used to create XML source documents. After the source documents were created, PTC Arbortext Styler was used to develop and apply stylesheets for different formatting outputs, which would encompass this case study goals.

PTC Arbortext Editor is a tool from Arbortext family that is used for creating reusable component-based content. PTC Arbortext Editor supports the latest DITA 1.3 specification. The content is topic-based and product-centered, the possible output formats include multiple media (.pdf, .rtf, .html, .html help, mobile, etc.). Structured authoring with PTC Arbortext Editor enables reuse of the

content, which helps to optimize the time that content creators spend on inputting information for multiple versions of the same source and improve information accuracy during authoring process. The use of standardized rules for content promote authoring consistency. Additional functionalities and features include, for example, use of hotspot links, intelligent graphic support, embedded animations and illustrations, as well as robust standard support (XML, SGML, XSL, XML Schema, Schematrons, XPath, XInclude, DOM and other next-generation Web standards for sharing content). (PTC 2018a, cited on 05.06.2018.)

The User Interface (UI) of the tool is intuitive, utilizes common word-processing capabilities (change tracking, drag-and-drop, and keyboard macros) and includes customizable toolbars. Figure 5 presents a basic, out of the box view of the PTC Arbortext Editor interface.

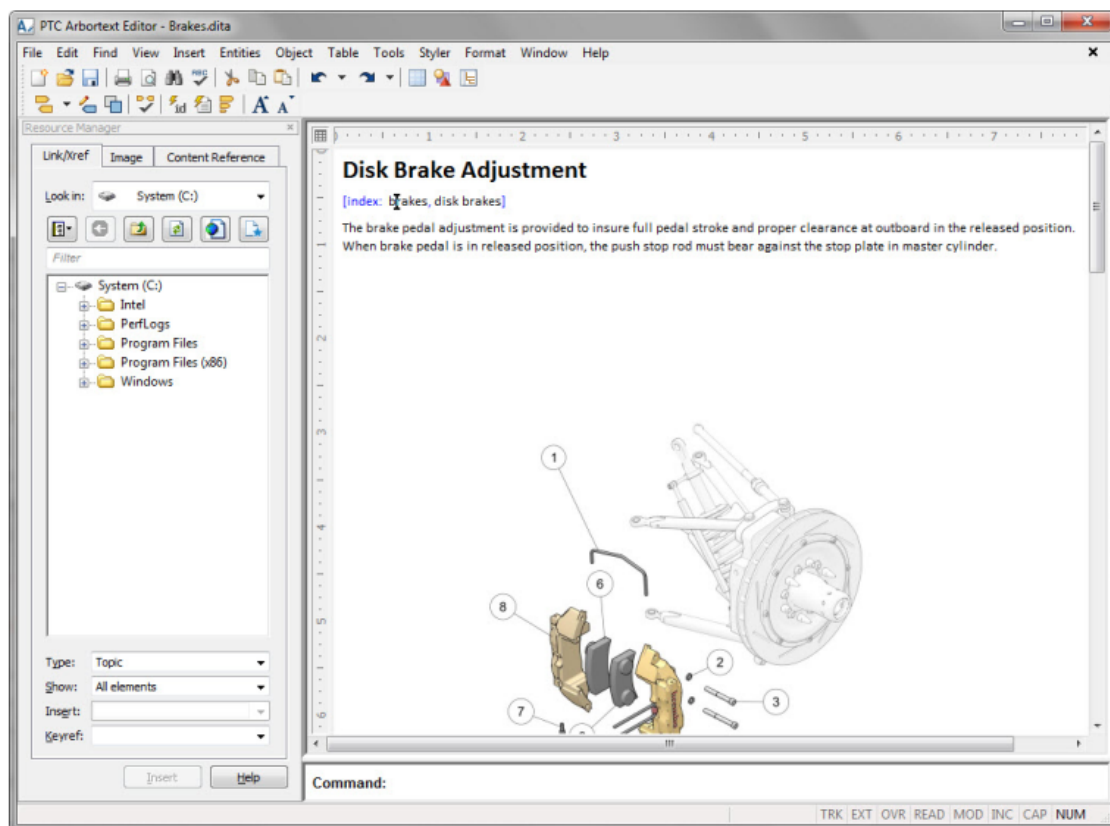


Figure 5. PTC Arbortext Editor UI, out of the box configuration.

Image source: PTC 2018a, cited 05.06.2018.

Customizable views and extensive preference settings allow to modify the working environment according to content creator's needs. Figure 6 illustrates the customizes PTC Arbortext UI with several toolbars removed, tag color changed and tag view set to full.

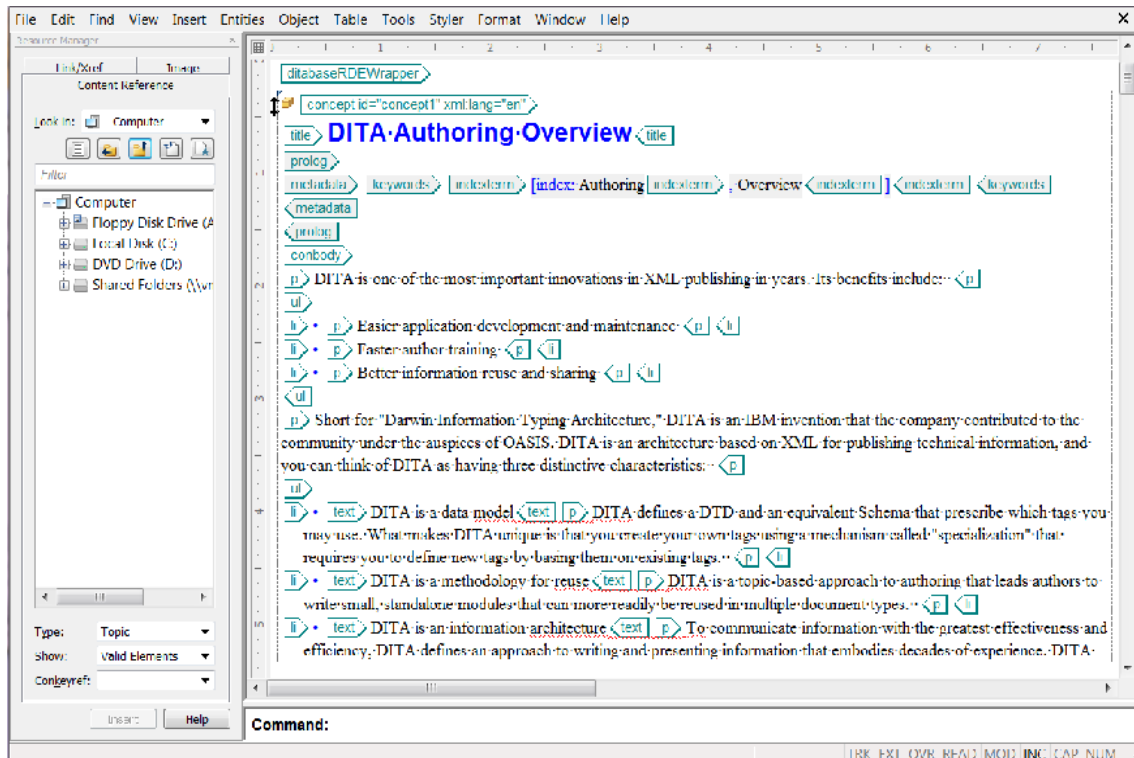


Figure 6. Custom window configuration in PTC Arbortext Editor.

Arbortext Styler is another tool from Arbortext family. It is used to develop and configure stylesheets for automated publishing of structured content using a single source of formatting for various outputs. With PTC Arbortext Styler, developing XML-based stylesheets does not require programming skills, however, it's also possible to create custom source edits if it is required.

Stylesheets created with PTC Arbortext Styler cover all aspects of the document formatting, providing such valuable additional functionalities for mass-formatting as customizable Property Sets, which are a combination of formatting options that can be applied to elements. The content creators can also develop own User Formatting Elements that can be referenced in many ways to optimize the output. Generated content can be assigned to elements, mostly useful in page regions (headers/footers), list elements and notes, warnings and cautions.

The stylesheets created in PTC Arbortext Styler are used by Arbortext publishing software to transform to different outputs and support different languages. A single interface is used to create all structured content layouts. Within a single stylesheet, formatting can be shared across media types, or set to accommodate specific output medium. It's also possible to import existing FOSI stylesheets and export multiple stylesheet formats, including XSL-HTML, XSL-FO, and FOSI.

The user interface of PTC Arbortext Styler is shown in the Figure 7.

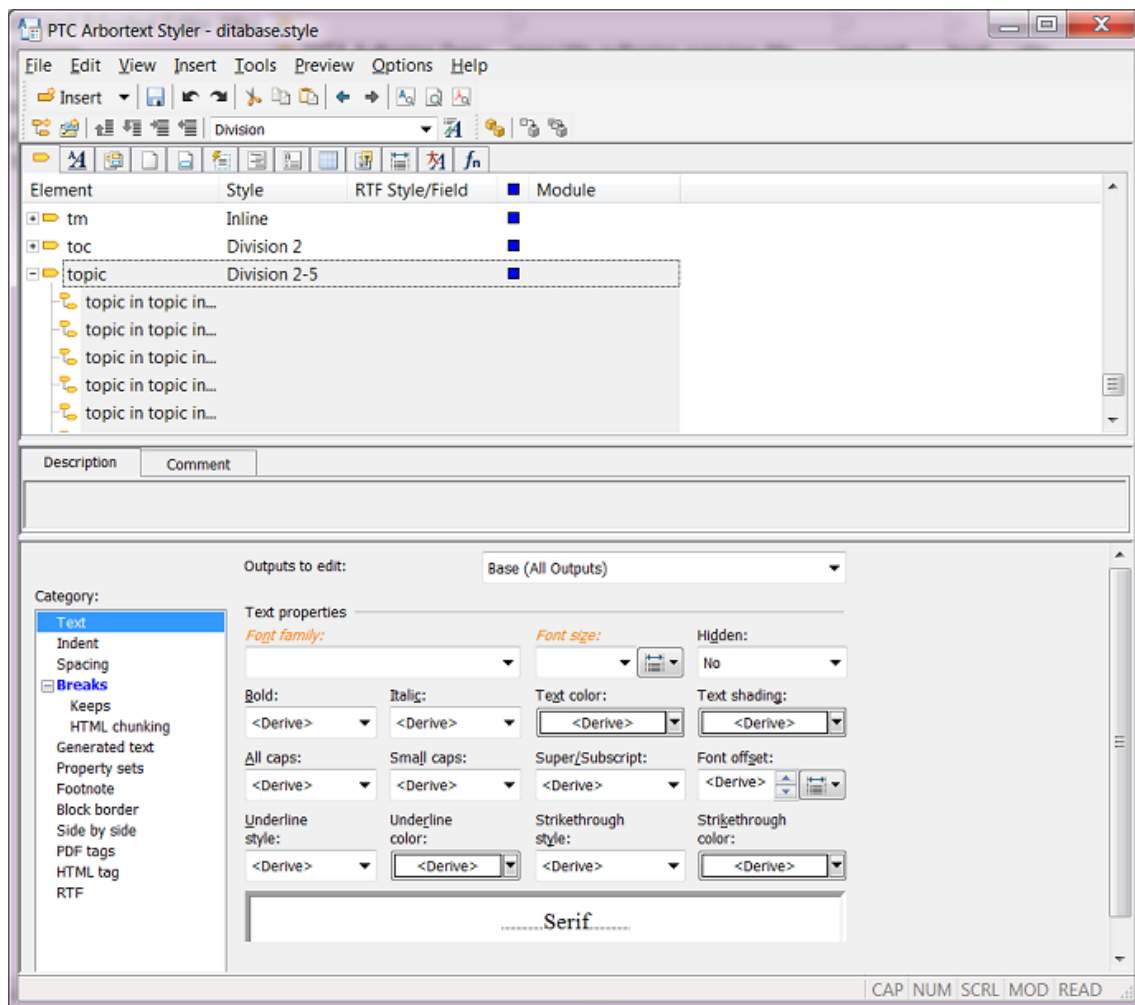


Figure 7. PTC Arbortext Styler UI.

The interface of PTC Arbortext Styler is not customizable, but it is rather compact and still quite intuitive as the formatting options are verbally described and elements names and the whole environment is synchronized with Arbortext® Editor. It's also possible to preview the formatting of the elements in Arbortext Editor window, where applicable. As mentioned before, PTC Arbortext Styler does not require programming skills from the content creator.

3.3 Implementation

The demo needed to be used as such, as well as to be easily adjusted, if necessary, to reflect customers' needs. An analysis based on the initial task requirements was performed to identify the contents of the demo deliverables. After the theoretical framework has been established and reviewed, the following goals were identified for demo package contents:

1. Develop two stylesheets with different formatting that will be used for the same XML-source to illustrate structured information approach when **content is separated from the form**.
2. Develop one more XML-source with different structure but several same elements as in the previous source to illustrate structured information approach to content when **content is fragmented into separate logical elements** that form a system.
3. Develop a stylesheet that accommodates various outputs (.pdf and Web as most used ones) for the same XML source to illustrate **media-neutral** approach to content in structured information.
4. Develop two different specialization type XML-sources that are formatted with the same stylesheet to illustrate the **uniformity and standardization**.

The goals were achieved by creating a package that contains several Extensible Markup Language (XML) sources with a varying structure and two different stylesheets to be used for formatting of the XML sources. The content and possible combinations of these sources are presented in the following sub-chapter. Two stylesheets for formatting of outputs were developed in process (Stylesheet1 and Stylesheet2), and some visual examples of the outputs are presented as well. The textual content of XML-sources is taken from the available samples in the XML-authoring software, whereas the illustrations are obtained from free sources listed in references.

3.4 Results

As the main goal of the case study was to address the structured approach to the technical content creation by using PTC Arbortext Editor, the content of demo package was designed with the concepts in mind listed below.

(1) Separation of form and content

The demo focused on the benefits of the core concept of structured information philosophy: the separation of form and content. Authored in XML, the content of the demo was literally free of the

form, which can be seen in Figure 8 that pictures the content of one of the DITA topics (specialization as a task) opened in Notepad++.

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!--Arbortext, Inc., 1988-2016, v.4002-->
3 <!DOCTYPE task PUBLIC "-//OASIS//DTD DITA Task//EN"
4 "task.dtd">
5 <Pub Inc?>
6 <task id="NewTopicTab" xml:lang="en">
7 <title>Creating New Topics</title>
8 <prolog>
9 <metadata><keywords><indexterm>Resource Manager</indexterm><indexterm>Creating</indexterm>
10 </metadata></indexterm></indexterm></indexterm></keywords></metadata>
11 </prolog>
12 <taskbody>
13 <steps>
14 <step><cmd>To open the Resource Manager select one of the following:</cmd>
15 <substeps>
16 <substep><cmd><menucascade><uicontrol>View</uicontrol><uicontrol>Resource
17 Manager</uicontrol></menucascade> menu choice.</cmd></substep>
18 <substep><cmd><menucascade><uicontrol>Insert</uicontrol><uicontrol>
19 New Topic</uicontrol></menucascade> menu choice.</cmd></substep>
20 <substep><cmd><uicontrol>Resource Manager</uicontrol> toolbar button.</cmd>
21 </substep>
22 </substeps>
23 </step>
24 <step><cmd>The <uicontrol>Resource Manager</uicontrol> opens. Ensure
25 that the <uicontrol>New Topic</uicontrol> tab is active.</cmd></step>
26 <step><cmd>Use the <uicontrol>Document Type</uicontrol> field to select
27 the type of topic you want to create from the list. </cmd></step>
28 <step><cmd>Check the <uicontrol>Sample</uicontrol> check box if want
29 to use the sample document (for the selected topic type) to create
30 the new topic.</cmd>
31 <info><p>By default, the template document is used to create a new
32 topic. </p></info>
33 </step>
```

Figure 8. XML source DITA Task opened in Notepad++.

This content is readable for a human, but it is not very user-friendly to modify. Figure 9 shows the same content that is opened in PTC Arbortext Editor. The content is significantly more comfortable to edit, especially if the content creator doesn't have previous experience with programming.

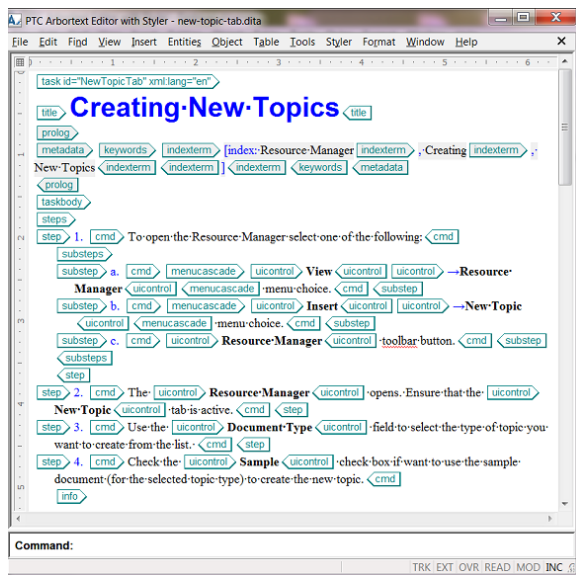


Figure 9. XML source DITA Task opened in PTC Arbortext Editor.

In the demo, the content is presented by various XML sources, whereas the form differed in the published outcome based on the formatting specification set in the stylesheet for varying outputs. During the demo package creation, two stylesheets (Stylesheet1 and Stylesheet 2) were developed

to accommodate outputs for .pdf publication and Web for a DITA Map. For the purposes of this demo, DITA Map is used as a main XML content source. Its structure includes several nested DITA Topics and one DITA topic specialization, DITA Task. In the example below, DITA Map has yet no additional metadata or attributes. This map version is further on referred to in this thesis as DITA Map1.

The following figures illustrate the publications that were produced at this stage of the case study. The publication is focused on media-neutrality that is achieved by structured information approach, and DITA topic-based authoring principle is also presented.

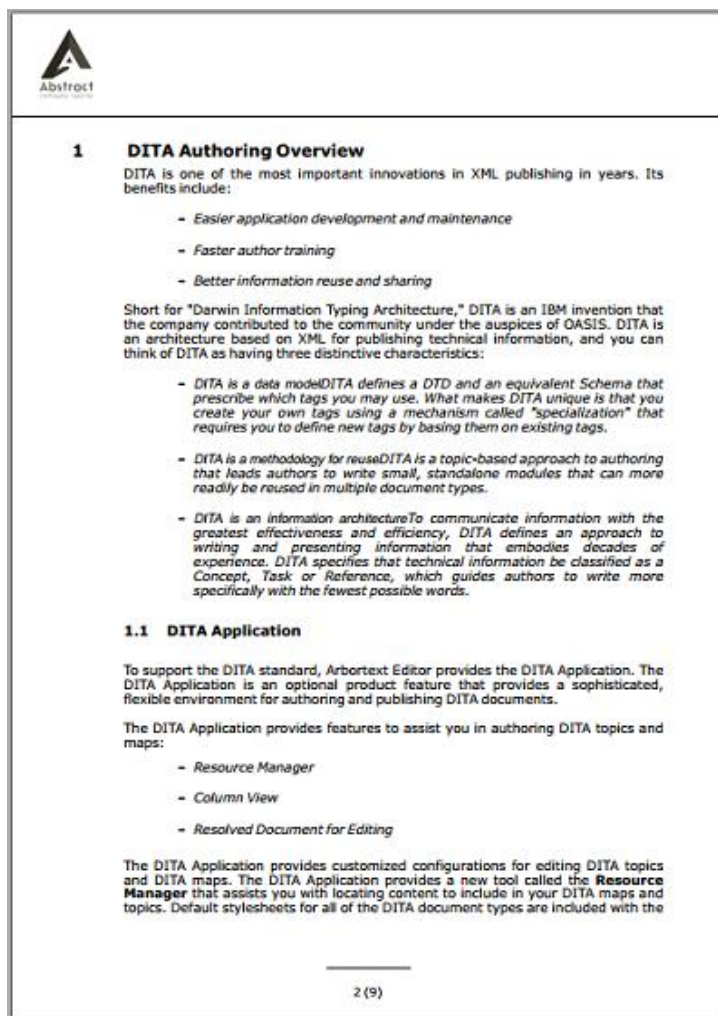


Figure 10. A page from .pdf publication created with DITA Map1 and Stylesheet1.

Logo used in publication: Freepik 2018, cited on 01.06.2018, https://www.freepik.com/free-vector/a-abstract-logo-design_822546.htm#term=free%20logo%20design%2

The formatting of the same XML source differs for the same output (.pdf) with different stylesheets. This could be useful for example when a company needs to produce significant amount of similar content for different products. The same approach could be used to accommodate different publication standards in different countries: for example, follow some country-specific standards for certain publications in terms of page size, margins, warning format and other formatting styles.

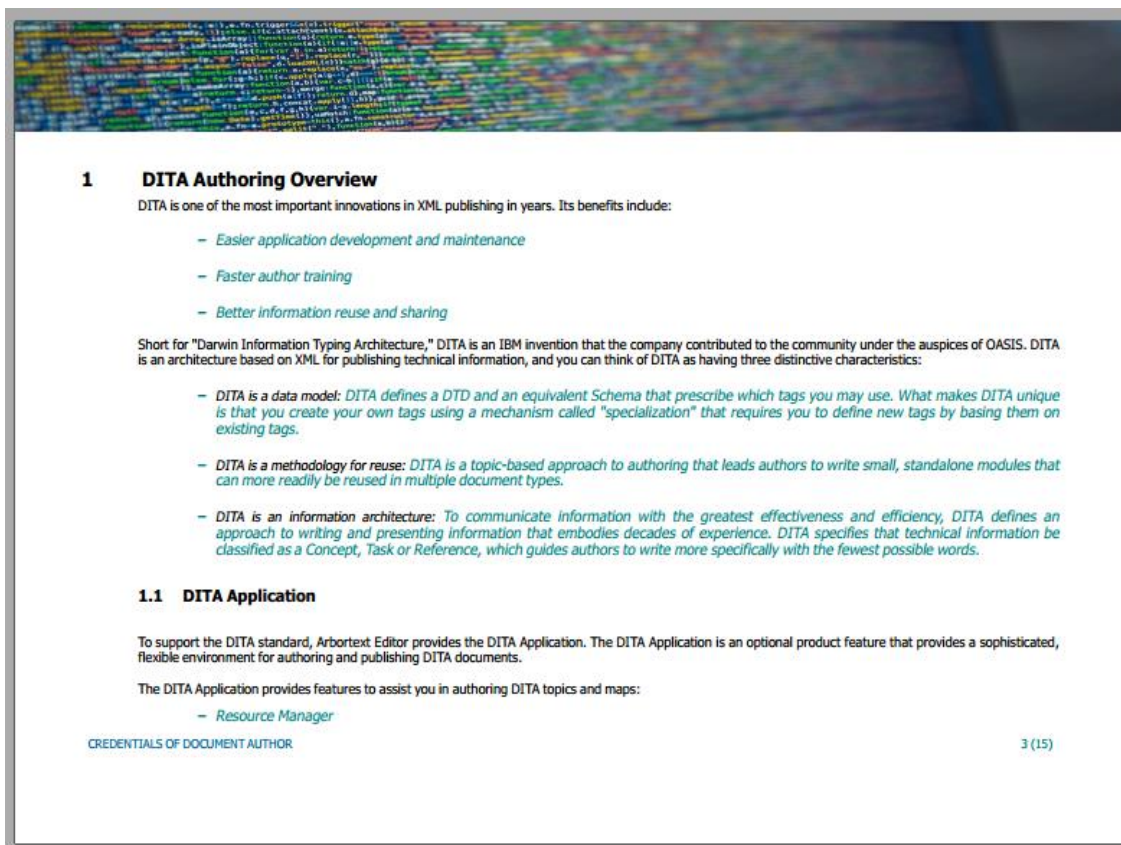


Figure 11. A page from .pdf publication created with DITA Map1 and Stylesheet2.

Header image used in publication: Guardxinc 2018, cited 05.07.2018, <http://www.guardxinc.com/blog/slider-coding/>.

In the figures above, the same page of the DITA Map is captured published with two different visual outputs. The content is the same, but the way the information is presented differs. Some of the elements in this example look similar in the first output variant with other elements, for example, <text> element formatting in the unordered list is the same as formatting of a <p>(paragraph). In the second output it has different formatting assigned to separate it from the paragraphs content and perhaps emphasize it. This is achieved purely via stylesheet use, without the need for author to decide whether this element will be emphasized in the visual output or not. The variation of handling ways for this element is potential but does not necessary need to be applied.

The next example illustrates DITA Map1 published with Stylesheet1 for Web output. The formatting of paragraphs and lists is similar, whereas the formatting of titles is different from the .pdf output, and a section title (DITA Application) is excluded out of the Table of Content. This is achieved by applying a different set of rules specifically for Web output via same Stylesheet1.

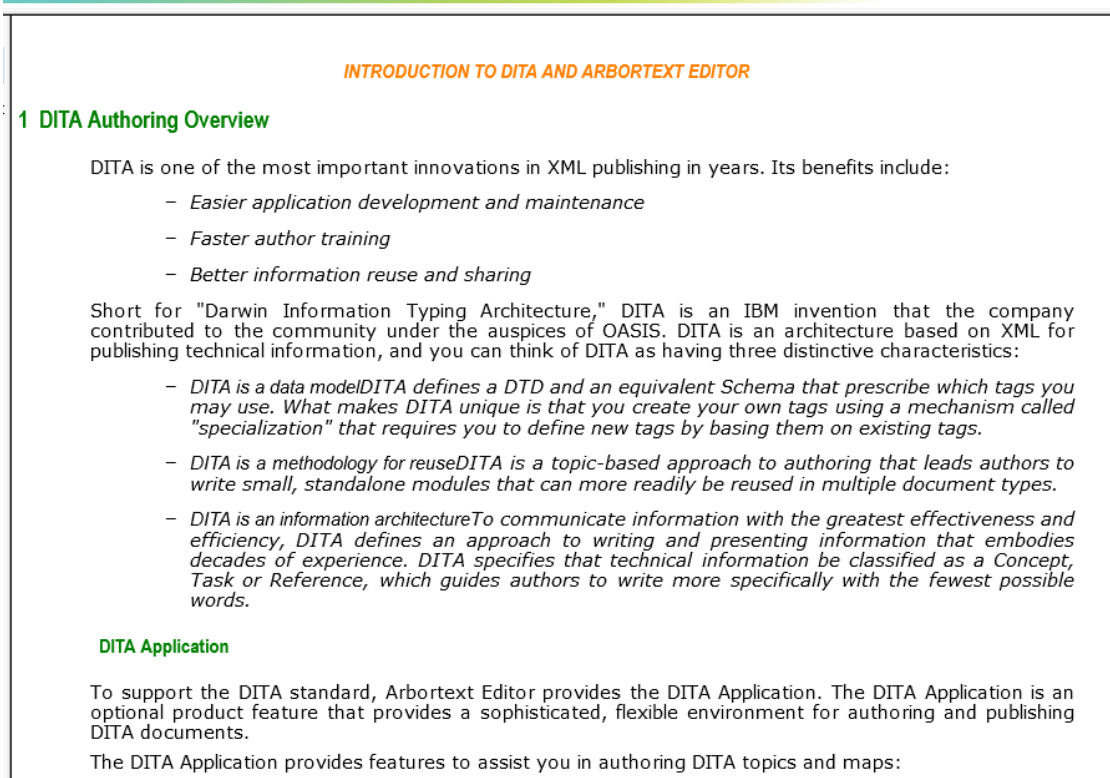


Figure 12. A page from publication for Web created with DITA Map1 and Stylesheet 1.

(2) Information architecture

To follow the principles of information architecture, the document production should be as clear as possible to accommodate collaboration, so that the documentation that is produced is accurate and consistent. In this demo, the content creation process is streamlined: content is authored in xml, assembled into a structure (maps and topics), formatted via use of stylesheet and is published to different outputs. The accuracy and consistency are achieved by following DITA standard that is supported by PTC Arbortext Editor. The demo illustrates how data is divided into units of information during the first stage of Information Life Cycle, the creation. Authoring is topic-based, and the structure and relationship of topics and maps is clear for all content creators, if there would be any need for collaboration.

Structured information approach to content where content is fragmented into separate logical elements that form a system is illustrated by this demo as well. Two types of DITA topics, DITA Topic1 and DITA Task1 were included in publication within DITA Map1. Since both topic types share the same purpose (to provide information) they are placed and numbered as equal division levels, where each of them forms a chapter. However, the latter chapter provides information on how to perform a series of steps, so the unit of information is set to be task, rather than simply topic. According to DITA standards, task includes such elements as “steps”.

The two above-mentioned XML-sources have different structure but share some common elements, one of which is “Title in Figure”. In PTC Arbortext Styler, it is possible to assign varying formatting for the same elements if they have different parent or ancestor elements, but for the purposes of this study the formatting for same elements within the whole DITA Map was set to match in Stylesheet1.

Figure 13 and Figure 14 demonstrate publishing outcomes for DITA Topic and DITA topic specialization (Task) that have same element “Title in Figure” formatted in the same way for both topic types.

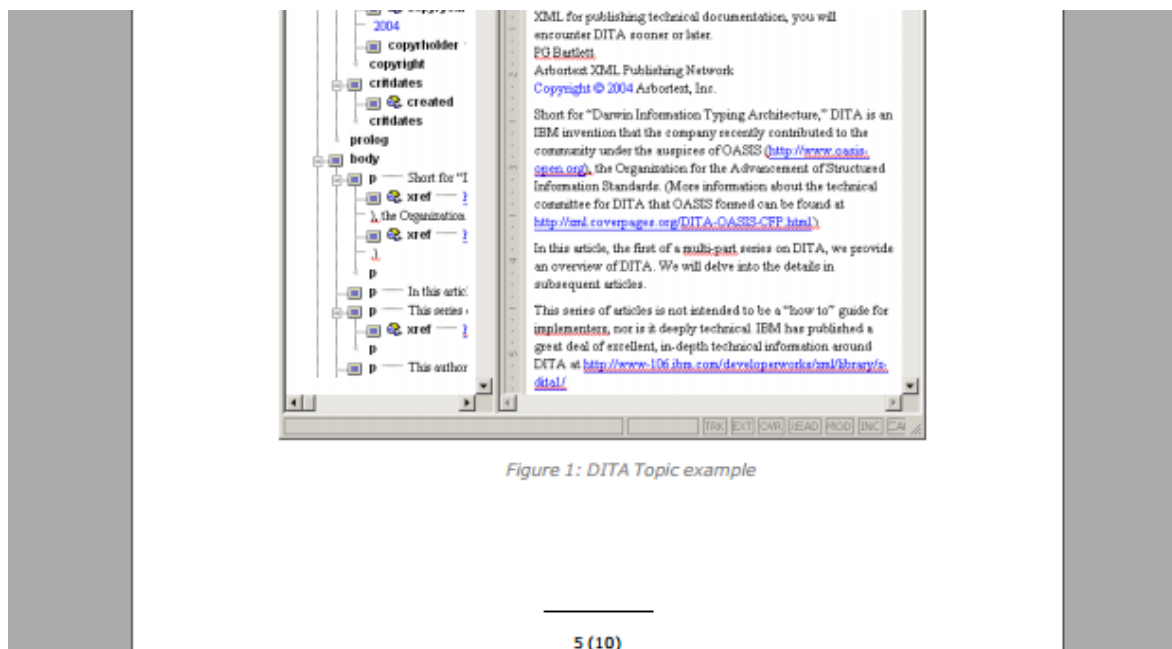


Figure 13. Element “Title in Figure” in DITA Topic published with Stylesheet 1.



Figure 14. Element “Title in Figure” in DITA Task published with Stylesheet 1.

In this example, the formatting of figure title is set to follow the same visual style as the similar element that is used in different topic specialization in the Figure 14.

(3) DITA core concepts

To illustrate the core concepts of DITA, a map containing a system of nested topics was developed. A publication to .pdf was created with all topics included, then DITA Topic1 was excluded out of a publication by assigning an attribute that controls publishing. Table of content was created for DITA Map1 by adding a map processing attribute value for an attribute “toc”. The updated DITA Map will be later referred to as DITA Map2.

Abstract
company logo

TABLE OF CONTENTS

1	DITA AUTHORING OVERVIEW	3
1.1	DITA Application	3
2	AUTHORING DITA TOPICS	5
3	AUTHORING DITA MAPS	6
4	RESOURCE MANAGER DIALOG	7
5	CREATING NEW TOPICS	9

Figure 15. Published Table of Contents of DITA Map2 with all topics included in publication.

Logo used in publication: Freepik 2018, cited on 01.06.2018, https://www.freepik.com/free-vector/a-abstract-logo-design_822546.htm#term=free%20logo%20design%2




TABLE OF CONTENTS

1	AUTHORING DITA TOPICS	3
2	AUTHORING DITA MAPS	4
3	RESOURCE MANAGER DIALOG	5
4	CREATING NEW TOPICS.....	7

Figure 16. Published Table of Contents of DITA Map2 with first topic excluded out of publication.
Logo used in publication: Freepik 2018, cited on 01.06.2018, https://www.freepik.com/free-vector/a-abstract-logo-design_822546.htm#term=free%20logo%20design%2

To illustrate another basic concept of DITA, metadata element “author” was set in Stylesheet2 to be included in the footer, whereas in Stylesheet1 it was not used in the output at all. When added to DITA Map2, the information that was stored in “author” metadata was published in the footer at the place assigned via Stylesheet2. The outputs are presented in Figure 17 and Figure 18.

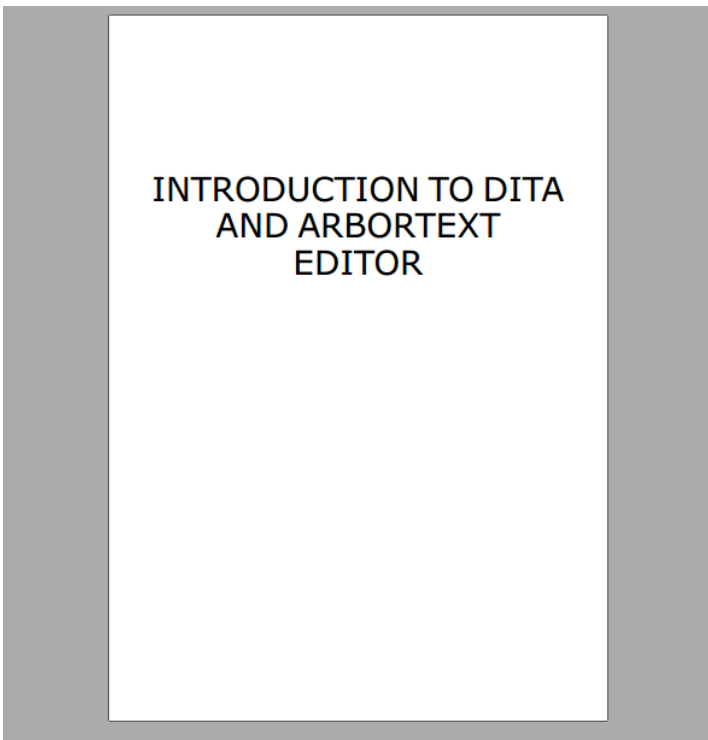


Figure 17. Title page of DITA Map1 published with Stylesheet1.

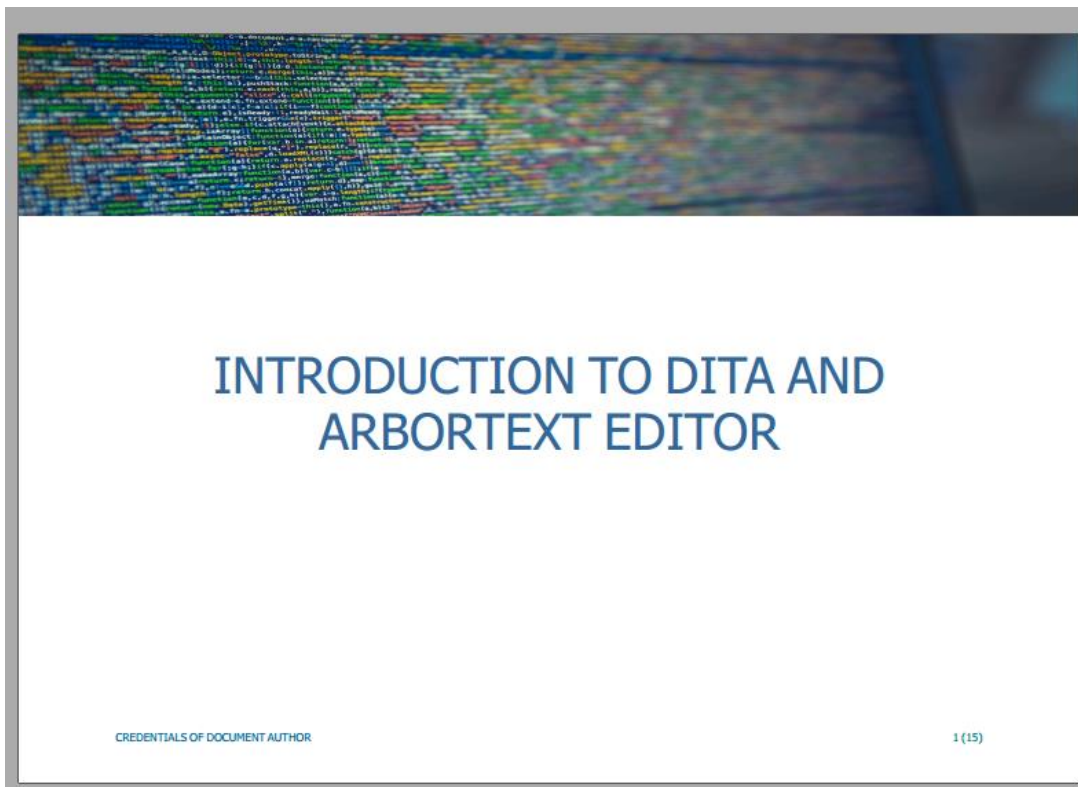


Figure 18. Title page of DITA Map1 published with Stylesheet2.

Header image used in publication: Guardxinc 2018, cited 05.07.2018, <http://www.guardxinc.com/blog/slider-coding/>.

The results of the demo package creation that are presented above correspond to the goals initially set in the requirement specification. The advantages of structured information approach to technical documentation content creation as well as the principle aspects and core elements of DITA can be demonstrated with the aid of the demo package. PTC Arbortext Family tools can facilitate the modern ways of content creation, where information architecture principles are followed throughout the Information Life Cycle.

4 FINDINGS AND DISCUSSION

This chapter comprises the description of the findings that emerged during the research process and the observations that might be useful for the future. Structured information approach, information architecture and modern technologies are in the spotlight of the current trends technical documentation content creation. DITA is already very popular in the industry due to its various benefits, which is especially true in the scenery of technical documentation production for bigger companies. However, the companies of a smaller scale still can find this standard useful to adopt.

DITA philosophy in technical content promotes media-neutrality and principles of reuse. These two concepts form the basis for development of solutions for modern technical documentation that need to address the rising expectations prompted by development of technology. With structured information approach to technical content creation, the way the information is perceived is changing, treating the information as a competitive advantage or an asset, which should be treated in the most efficient way. The information can be shared across the company in a form of units and structures, which can also be authored collaboratively, and its accessibility is enhanced. Using the structured approach, the storage and management of data can be centralized and automated. Once created, the content is easily tailored, freed from the page and can be reused, updated and linked to other entities (for example, illustrations).

DITA principles are based on the use of descriptive markup. Using descriptive markup in production of technical documentation provides some practical advantages to the digital objects that are produced by adopting single-sourced publishing, where same source can be used with multiple presentations and the presentations can be easily changed through stylesheets.

Another advantage of descriptive markup use is the versatility of the ways the content can be created. Since the source is separated from the form, authoring process of the document is brought to more basic, natural level, when the actual creation of the textual content and the visual aspects of the object produced can be performed by different people with different areas of expertise. For example, a technical writer does not need to take the aspects of visual design in consideration while following the standard of writing. The output formatting can be developed by the professional in the visual design sphere. At the same time, these two tasks can be easily combined if the content creator's tasks are to develop both the output formatting and the source.

This is especially applicable for the tools in PTC Arbortext product family. Using PTC Arbortext Editor&Styler does not require any specific coding skills, so both authoring of the content and designing its format can be performed by content creators without undergoing training for specific coding language.

There are also some additional benefits of the descriptive markup in technical documentation. One of the benefits could be identified as a consistency of the relationship between structure and presentation (each element is assigned specific format based on the element's position within the source and its relations to other elements). Another one would be the nature of presentation: even though its aimed to be visually appealing, user-friendly and clear, the form's core purpose is to express the function of the content (the ways content can be applied). As to the content, with the descriptive approach it becomes purely structural. The content is structured into elements that can later can be stored separately and combined into new content, which offers a lot of reuse possibilities for content creation.

Modern technical documentation can offer way more than simply displaying documents. Enhanced by the innovative technology, for example, augmented reality, the form is as interactive, user-friendly, accurate and clear as ever. The content can be delivered on demand and tailored to address any changing requirements, all done separately from adjusting the form.

REFERENCES

Coombs, J. H., Renear, A. H. & DeRose, S. J. 1987. Markup Systems and the Future of Scholarly Text Processing. *Communications of the ACM* (30) 11, 933-947.

Day, D., Priestley, M. & Hargis, G. 2005. Frequently Asked Questions about the Darwin Information Typing Architecture. Answers about the XML-based Darwin Information Typing Architecture (DITA) for documentation. IBM. Cited 01.06.2018, <https://www.ibm.com/developerworks/xml/library/x-dita3/index.html#N210>.

DeRose, S. J. 1995. Structured Information. Navigation, Access, and Control (Berkeley Finding Aid Conference, April 4-6). Cited 05.05.2018, <http://xml.coverpages.org/deroseStructure.html>.

Flanders, J. 2007. Descriptive Markup. From Introduction Markup Lecture. Women Writers Project, Northeastern University, Boston. Cited 18.05.2018, www.wwp.northeastern.edu/outreach/seminars/UCLA/presentations/html/introduction_markup_lecture.xhtml.

Gartner IT-Glossary 2018. Information Life Cycle Management. Cited 08.05.2018, <https://www.gartner.com/it-glossary/information-life-cycle-management-ilm/>.

Kerry, M. 2018. Macmillian Dictionary Buzzword. Augmented Reality. Cited 01.06.2018, <http://www.macmillandictionary.com/buzzword/entries/augmented-reality.html>

Kirsanov, D. 1997. HTML Unleashed. SGML and the HTML DTD: Procedural and Descriptive Markup. Cited 05.05.2018, <http://webreference.com/dlab/books/html/3-1.html>

Merriam-Webster Dictionary 2018. Markup. Cited 08.05.2018, <https://www.merriam-webster.com/dictionary/markup%20language>.

OASIS 2005. Darwin Information Typing Architecture (DITA) Architectural Specification v1.0. OASIS Standard. Cited 01.06.2018, <http://docs.oasis-open.org/dita/v1.0/dita-v1.0-spec-os-ArchitecturalSpecification.pdf>.

OASIS 2017. Darwin Information Typing Architecture (DITA) TC. Cited 01.06.2018, https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=dita.

OASIS 2018. About Us. Cited 01.06.2018, <https://www.oasis-open.org/org>.

Perugini, S. 2003. Introduction to XML. Computer Science Dept, College of Engineering, Virginia Tech. Cited 28.05.2018, <http://courses.cs.vt.edu/~cs1204/XML/htmlVxml.html>.

PTC 2018a. Arbortext ® Editor™ datasheet. PTC Inc. (PTC). Cited 05.06.2018, <https://www.ptc.com/-/media/Files/PDFs/SLM/Arbortext-Editor-data-sheet.pdf?la=en&hash=326216591079709AA3BC85EAED07884F772BFE41>

PTC 2018b. Schneider Electric Digest. Cited 28.05.2018, <https://www.ptc.com/en/case-studies/schneider-electric-digest>.

Samuels, J. 2014. What is DITA? Darwin Information Type Architecture. TechTwirl. Cited 01.06.2018, <https://techwhirl.com/what-is-dita/>

Shannon, R. 2018. What is HTML? Cited 28.05.2018, <http://www.yourhtmlsource.com/starthere/whatishtml.html>.

TechTarget 2005. Information Lifecycle Management. Cited 08.05.2018, <https://searchstorage.techtarget.com/definition/information-life-cycle-management>.

The Information Architecture Institute 2013. What is IA?. Cited 28.05.2018, <https://www.iainstitute.org/file/whatisiapdf>.

The World Wide Web Consortium 2018. Introduction to SGML. Cited 01.06.2018, www.w3.org/TR/html4/intro/sgmltut.html#h-3.1.

Wodehouse, C. 2018a. How AJAX works? Cited 01.06.2018, <https://www.upwork.com/hiring/development/how-ajax-works/>.

Wodehouse, C. 2018b. The Basics of Web Development. Cited 01.06.2018, <https://www.upwork.com/hiring/development/the-basics-of-web-development/>.

Wurman, R. S. 1997. Introduction. Published in Wurman, R.S. (ed.) Information Architects. 1st edition. New York: Graphis Inc., 16.

XML Objective 2013. What is the difference between XML and HTML? Cited 18.05.2018, <http://www.xmlobjective.com/what-is-the-difference-between-xml-and-html/>.