An Approach for Efficiently Creating and Managing Content for Multimedia Environments

Michael Klenner[1*]

[1] Zwickau University of Applied Sciences, Informatics, Zwickau, Germany

* Corresponding author. email: Michael.Klenner@fh-zwickau.de
Manuscript submitted September 24, 2014; accepted October 29, 2014.
doi: 10.7763/ijeeee.2014.v4.352

Abstract: In some communication environments, it is advantageous to provide contents for multiple modal channels redundantly and not only in a single media representation. For the case of educational content, this can be for example: visual image-heavy presentation slides for face-to-face teaching, text-based lecture notes with full details for following up at home, and self-study online content for e-learning. There is an overlap in content and structure between all those media files whereas there are differences in style and extent.

The requirements for creating and managing content for those multimedia documents are high: On the one hand, the creation process should take care of all media-specific characteristics to obtain the best possible documents, and on the other hand, authors want manage and reuse content in an efficient way. This paper analyzes the lack of current authoring technology, and presents a new approach which suggests a combination of traditional media-specific files and authoring tools with a connected single-source content system.

Key words: Cross media publishing, media specific writing, multimedia content management, single sourcing.

1. Introduction

Over the years, the perception of the term *multimedia* has changed. In the past "the term has meant the use of several media devices in a coordinated fashion (e.g., synchronized slides with audiotape). Advances in technology, however, have combined these media so that information previously delivered by several devices is now integrated into one device" [1]. Thus, nowadays when speaking about this term, normally the point of interest is a single document which combines different forms of visual and audio media. Nevertheless, there still exist situations when several independent documents with different media representation were used in some kind of coordinated fashion to depict similar content in distinct ways. Or, to put it in the words of [2]: "Multiple media means that the same content is delivered to end-users in more than one medium". This is usually the case when the same content should be delivered on different communication channels or in distinct communication contexts.

This occurs, for example, when lecturers provide educational materials for distinct communication channels with multiple media types. For instance: visual image-heavy presentation slides for face-to-face teaching, text-based lecture notes with full details for following up at home, and self-study online content for e-learning [3]. Maureen Walsh analyzed in [4] three studies, where multiple educational materials were used in some multi-modal learning environments, and came to the conclusion that learners can profit by the

mixture of different media forms. Other multimedia scenarios can be found in journalism (providing news as print and online version) or product advertising (e.g. product description as website, printed catalogue, presentation slides for sales events).

Depending on the context of the reception, authors have to choose adequate communication channels and provide the content in appropriate documents. Not all materials suit to every reception context, because every media form has some specific style characteristics [5]. Each one of the media is different as to the semiotic style, degree of specificity, content fragmentation and expression variety. An author has to care about these characteristics to be able to create documents which fit the target environment. For learning content, it is certainly possible to use only one single document for all teaching situations. However, the desired learning success can then fail because the learners cannot follow the content of the course in an appropriate manner. The consequence is that several unique documents are required to support each context of reception adequately. The author has to create and maintain every single document, although there is an overlap in content and structure between all the materials. The problem is, these materials differ in style and extent.

Working with the example of multiple media educational materials, the lack of current authoring technology will be analyzed in this paper, and in the second part, a new approach will be demonstrated on how authors can create and maintain those documents more efficiently.

2. Current State of Authoring Technology

This section will provide an overview about current technology approaches which can be used to author and manage content for multiple media. It is necessary to distinguish between two basic strategies: firstly, creating individual separated files by using media-specific editors, and secondly, generating materials from single-source contents automatically.

2.1. Media-Specific Files and Authoring Tools

The most common way to create educational materials is certainly the usage of some media-specific editors. There is a wide range of authoring software solutions which are designed to create and edit only one particular media type. For instance, there are the well-known software packages like Microsoft Office or OpenOffice. All of these packages contain editors which support some specific media forms like presentation slides (e.g. Microsoft PowerPoint) or continuous text documents (e.g. Microsoft Word). Using these tools lends a big advantage to the creation process. Usually these editors are able to take care of all media specific characteristics of the particular target media. For instance, almost all word processors offer an empty sheet of paper after starting a new document. From the beginning, it is clear which media type is the most desirable. Additionally, an editor like this mostly offers designing tools which are specialized to create content for the current media form. Endowed with a media-specific editor, it is easy to produce educational materials which fulfill the most common design requirements.

Returning to the fundamental issue of multiple media documents, it has to be clarified how such tools support the creation and maintaining of different education materials. Depending on the target media types of the educational materials, a number of media specific authoring tools are deployed simultaneously. Usually an author has to create an independent document file for each teaching context to take into account all relevant media characteristics. Therefore, the number of files increases with every new teaching situation. For example, when an author offers presentation slides, lecture notes, and e-learning resources for the same learning content, there are already three separate files and three different authoring tools involved. The most significant downside is that the files are not connected to each other, hence the only way to share redundant content between the files is to use the well-known copy and paste function [6], [7]. This leads to problems when parts of the content or structure are to be changed. This can happen, for instance,

when an old course may be updated and reused in a new semester. New content should be added, and existing content should be moved to another position or deleted. The author has to apply all changes to every independent file redundantly and manually.

2.2. Single-Source Content Management and Cross Media Publishing

Fig. 1 shows the basic components of a typical cross media workflow with a single-source content management (also referred to as single-source publishing). The main idea of this approach is that there is only one single-source storage which contains the whole content base. In there, a document is not saved as a traditional file, but the whole composition is split into smaller content fragments (individual texts, pictures, videos etc.). All these fragments are saved without any layout or formatting information. With the help of the publishing system and some predefined transformation rules, the content gets transformed into the actual output media automatically. These rules contain information about format and layout of the target medium, and instruct how the single-source content gets transformed into media-specific output material.

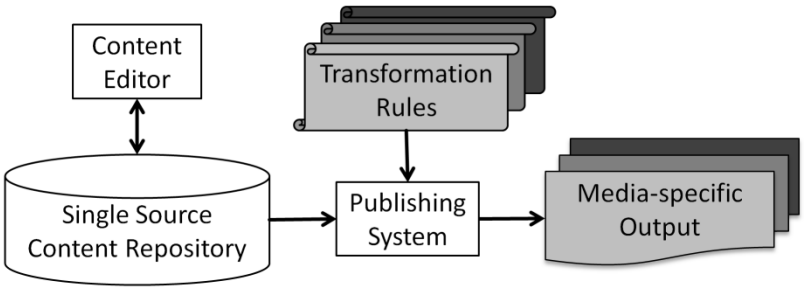

Fig. 1. Typical single-source publishing workflow.

The single-source publishing approach was developed originally for technical documentations in the early 1990s. The goal was to reuse existing content for different document versions to support various communication channels [6]. Because of similar requirements, authors of more recent studies have proposed that this authoring technology can also be used to produce educational materials [8], [9], [3].

According to [10], this "method particularly suits projects where the content of the different versions overlaps significantly and the information is likely to be revised or updated frequently". This way has a huge advantage over the first approach which was dealing with several separate files. With a single-source management, it is possible to perform content modifications (like inserting, moving, or deleting) at a central point. "A document is updated one time and then 'transformed' rapidly and dynamically into multiple formats as needed" [11].

When one publishes documents from single-source content, it does not mean that all output materials are using the exact same content and structure, merely with another media style. With the help of *conditional processing*, some selected content fragments can be annotated with special instructions for the following publishing step [12], [13]. Hence single-source publishing offers two approaches to create output media with divergent content:

- From the basic set of single-source content fragments, pieces can be chosen arbitrarily and reused for generating new materials. Not every fragment must be included in every output medium.
- It is possible to define alternatives. Thereby an author can use a video clip in the e-learning medium and a replacement text for print media.

Despite all the advantages, there are also downsides of this approach. Usually, the single source content repository is based on a domain specific content schema. Quite in line with this, the single-source content gets created and changed by a specialized editor supporting exactly this domain specific content schema.

Like any other authoring software, this editor supports only a limited set of expressions to depict the content. The consequence is that the output materials can only contain content expressions which are supported by the specific single source authoring environment. For instance, when the editor or the content repository is not able to handle video material, it is hard to get a movie clip into the output material.

Additionally, it should be noted that publishing from a single-source increases the requirements for the author, who is also responsible for content management from then on. It is not possible to focus completely on the writing process of a particular medium, because one is working on all output media at all times. While working on the single-source content, the author must beware of the impacts of all decisions for other media forms, and manage media-specific content fragment alternatives additionally. Out of convenience, this often leads to the situation that authors only using elementary ways to depict the content, which are supported by all media types, and don't pay attention to some superior features of more expressive media types. In this case, the output documents have little in common with a multimedia environment.

Another problem becomes visible by looking closer at the transformation rules and the automatic output creation process. The problem is, an author can influence the appearance of output material only in an indirect way. The author has to revise the transformation rules for the target media type to change the output appearance. These rules contain strict instructions for a computer on how to transform the single-source content into a media-specific file. On the one hand, this works fine for media-types with strictly defined layouts, like fluent continuous texts with predefined style templates. On the other hand, there are media types which allow a bit more creative designing, as for instance presentation slides. It starts with the allocation of content fragments to individual slides, and ends with free positioning and orientation of pictures and texts (Pötzschwas able to show in [14] that a lot of PowerPoint users often create free designed text-image-collages without use of the default layouts). For a computer algorithm, it is impossible to do something like this on the base of a transformation rule.

Finally, it can be summarized that on the one side, the single-source publishing approach is able to manage overlapping content at a central point, but on the other side, it can be seen that this technology is unable to take care of all media specific characteristics of the output materials.

3. New Approach

Based on these previous findings, a new technological approach is demonstrated in this section.

3.1. Basic Idea

This approach is based on the idea that the features of multimedia documents can be divided into two groups: shared characteristics which are the same across all concurrent used documents and media-specific characteristics which are unique for every single document. As mentioned earlier, there is an overlap in content and structure between those media. Consequently, both of these features belong to the group of shared characteristics. In contrast, layout, formatting and the actual depiction of the content are media-specific characteristics.

The two above-mentioned authoring approaches do not care about this distinction of media features on the technical level. On the one hand, all media features are saved in independent files (media-specific file approach), while on the other hand, all characteristics are embedded centrally in the single-source content and transformation rules (single-source publishing approach). The approach developed here combines both authoring strategies. The objective is to get the best of both worlds: using media specific editors for handling the media-specific characteristics, and connecting them with some kind of single-source content management to allow central modification of the common features. Fig. 2 demonstrates this setup in an exemplary manner.

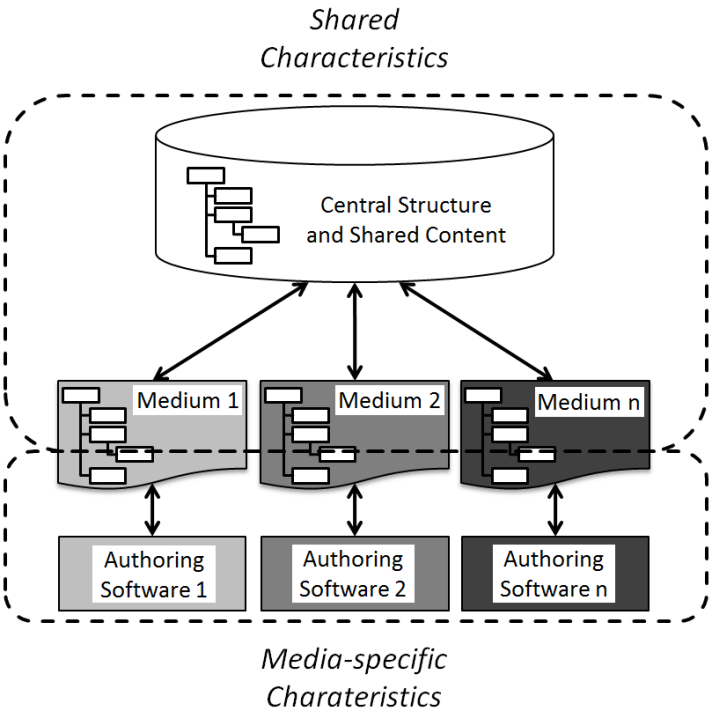

Fig. 2. Media-specific files connected with a single-source base.

All of these media forms are saved in traditional media-specific file types. Therefore, these files can be edited with usual authoring tools. This means the author can use all options which are offered by the media-specific authoring software to create materials in an optimal way.

3.2. Central Structure and Content Referencing

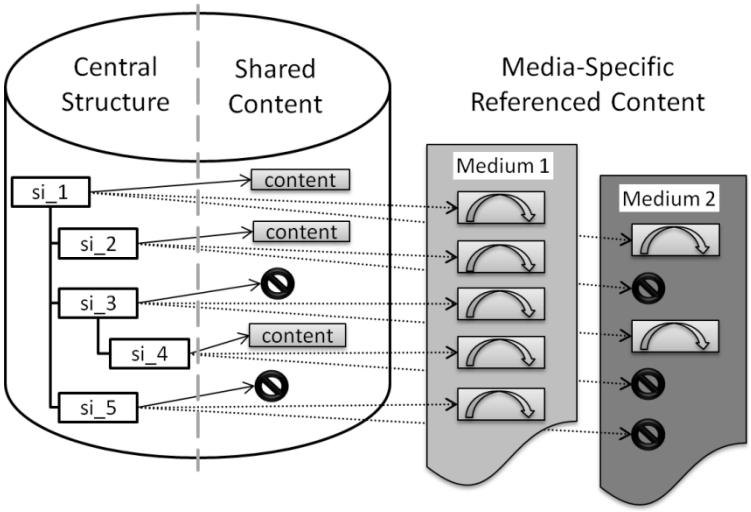

Fig. 3. Exemplary visualisation of shared and referenced content.

This approach, inspired by the single-source publishing, introduces a central data store to handle the shared media characteristics. Since the document files are connected to this, all the concurrent media files belong to the same super-structure. In principle, it is possible to use all types of structures (even linear or net-like), however, for the following demonstration, it is assumed that this is a hierarchical structure. In the first place, this super-structure is managed in the central data store and consists only of a set of ordered and encapsulated structure items. A structure item does not contain any content depiction itself. In the mind of

Plato's theory of forms [15], it is only a non-material abstract idea. The materialization of these structure items can be found in the connected documents. A structure item in the central data store can reference therefore a set of media-specific depictions in different connected media files.

Fig. 3 shows an example how the content fragments of two media-files are referenced by a super-structure. There are three ways to handle a structure item in a connected medium.

3.2.1. Particular media-specific content depiction

Each connected medium can use particular expressions to depict the same content (This is demonstrated in Fig. 3 for the structure item si_3). For instance, in the central super-structure exists a structure item with the idea of the anatomy of a horse. This structure item can reference to different media-specific content fragments in different media, like a horse picture in presentation slides, or a detailed text about the components of a horse in lecture notes.

3.2.2. Shared content

It is possible to use shared content, which is synchronized between all connected materials. Shared content is assigned directly to the belonging structure item, and is also saved in the central data store. When there is no shared content for a structure item, the shared content assignment remains empty (This is the case for the structure items si_3 and si_5).

If an author decides to reuse the horse picture in some other files, the connected media retrieves it from there. Due to the reason that all connected files saving their own copies of the horse picture in their own proprietary file format, it is important that the central structure item holds valid content references to all the media-specific redundant horses in the individual files.

3.2.3. Omission

It is not required that every media file implements all abstract structure items from the central data store (This is demonstrated in Fig. 3 for the materialization of the structure items si_2, si_4 and si_5 in medium 2). This can have technical (e.g. the medium is not able to show pictures) or didactical reasons (e.g. the medium describe the topic on more abstract level and leaves out detailed descriptions).

It therefore means that each connected material has only a subset of the central super-structure. So, it is possible to implement divergent media-specific document structures within the framework of the central super-structure.

3.3. Synchronization of Connected Media

As all documents are connected to the central data store, it is possible to apply changes of common media features to all other connected documents. The synchronization comprises two steps: the commit step and the update step. These behave similar to the same-named operations in the software versioning system Apache Subversion [16]. There are four different use cases for the usage of these operations.

3.3.1. Create a new central data store

The commit operation has the task to transmit the shared media characteristics to the connected central data store. If an author is working on a new document or opened an existing media specific file, the commit operation will create a new central data store, extract the document structure of the currently loaded medium and elevate it to the initial super-structure in the central data store. Subsequently, the abstract structure items in the new central super-structure get connected to their media-specific materialization in the source document. Finally, all referenced content fragments from the source medium are stored as shared content assignments in the central data store for further use by other media.

3.3.2. Create a new document from a central data store

The counterpart of this is the update operation, which synchronizes the shared features from the central data store into the connected media-specific document. For the case that there still not exists an updateable

document, the update operation creates a new document on the base of the central data store. The author has to select a source medium, which must be connected to the central data store already. This source medium acts as blueprint for the newly created document, because its structure and shared contents are transferred to the newly created document.

3.3.3. Commit document changes

When an author is working on an existing document which is already connected to the central data store, there are various characteristics that can be changed: structure (move, create, or delete content fragments), content (edit or replace content fragments), formatting (e.g. alter color and size of a text) and layout (e.g. change the visual position of a content fragment). During the next commit operation, the whole structure with all content gets transmitted to the central data store. At this place all changes get synchronized. Shared content, which was changed in the media-specific document, gets updated in the central data store. Furthermore, all local structure changes get applied to the central super-structure.

On the other hand, the changes for formatting, layout, and particular content depiction are saved only in the local media-specific file without submission to the central data store.

3.3.4. Updating existing documents

When a document is already connected to a central data store and another connected document intermediately applied some changes, the update operation has to synchronize these to the local media file. This is the most sophisticated use case in this approach. Some media types are easier to update than others. For instance, continuous text documents with predefined style templates can be updated automatically. With the help of the content references, outdated shared content gets found and replaced by newer versions from the central data store. The same happens with structure changes. After comparing the local structure with the central super-structure, it gets calculated which local content fragments have to get removed, inserted, or deleted. All these operations can be done by machine, because in continuous text documents the content fragments get laid out automatically by the word processor, and it is not possible to create an unusable state of the text document.

It is even harder for media types with more complex layout possibilities, like presentation slides. Applying shared content changes can be done automatically, but one must take into account that this can have impacts on formatting and layout also. For Example, when the outdated text was very short and the new one is much longer, it is possible that the text has not enough space on the slide without resizing the text fragment. Another problem appears when it comes to moving or inserting new content fragments. A computer algorithm cannot decide in a meaningful way on which slide or which position the content should be placed. For problems like this, it is necessary to involve a human being. This can be realized on the base of a wizard which guides the author through all problematic update changes.

4. Software Implementation

The most sophisticated task in this approach is to handle the connection of the media-specific content with the central data store. To the idea of accessing proprietary document formats [7] states:

"Furthermore, the content of a document is normally stored in a proprietary format which makes it hardly impossible for third-party applications to access any structural metadata that is completely controlled by the monolithic application 'owning' the document. These closed document formats prevent third-party applications to offer supplemental services related to parts of a document and make the external annotation of document substructures impossible."

This section will explain that this quote is no longer sustainable and present a proof-of-concept prototype. Based on some preliminary considerations, some technical key concepts will be provided to show the

proper functioning of the software prototype.

4.1. General Considerations

For software engineers, the most interesting question surely is how to realize the referencing mechanism between the structure items in the central data store and the media-specific content fragments in the proprietary media files. On the technical level, this has been solved by unique IDs. Every abstract structure item has one, and all belonging content implementations must be labeled with the same ID to set up the content reference. It is important, that this ID will never change. It is the task of the authoring software to ensure, that a content fragment remembers its reference ID for the whole lifetime of the document file. Only on this basis it is possible to track modifications of content fragments.

This can be done by two ways. The first strategy is to add the IDs as meta data directly into the media file. In the area of Office, DTP and online application, a lot of modern document file types are based on XML [17]. Due to the extensibility of XML it may be possible to integrate the content fragment IDs directly into the normal document file. When this is not possible, consideration might be given to saving the IDs in a separate document, which references itself to the content fragments in the proprietary media file.

When this fundamental issue is solved, it is time to think about a way to implement the commit and update operations. One approach is to realize the connection between the central data store and the material files with software plugins inside the media-specific authoring tools. This places high demands on the authoring software. In the first place, this means that it must be possible to develop plugins for it. Furthermore, these plugins must be able access some kind of interface for the communication with the central data store. And last but not least, the plugin must be able to access and modify the content of the currently loaded document to implement the commit and update operations.

4.2. Prototype Implementation

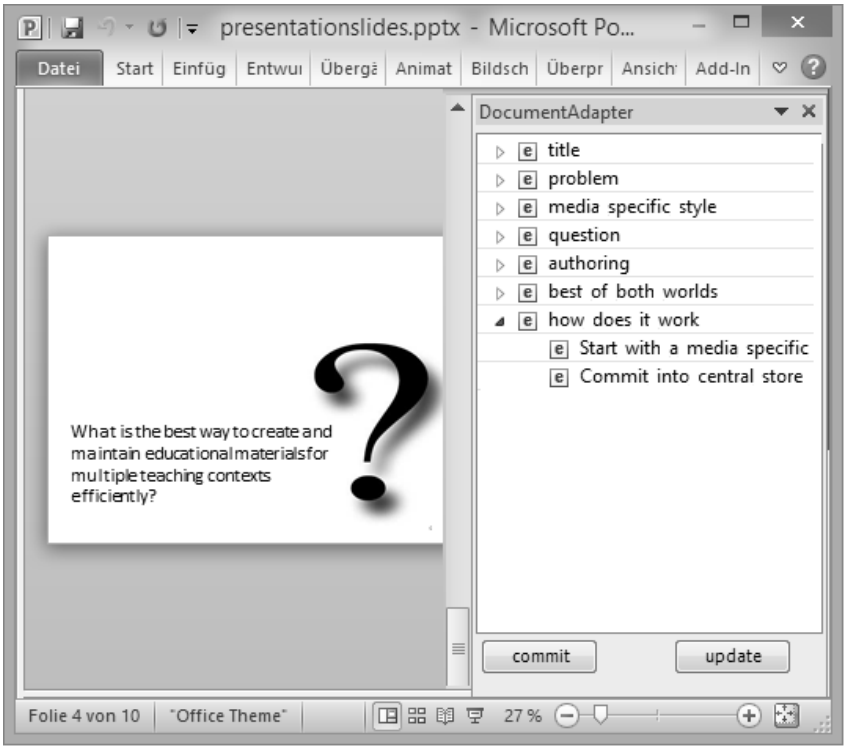

Fig. 4. A screenshot of the PowerPoint-plugin.

In this project, a basic central data store was designed in form of a java application with an open

interfacefor the commit and update operations. Technically this interface is based on the SOAP protocol for structured data exchange between heterogeneous applications. This service interacts with plugins for Microsoft Word and Microsoft PowerPoint which were developed on the base of C# with the Microsoft Office Developer Tools. The ID assignment of the content fragments was realized by the use of the *TextRange*concept ([18] for Word and [19] for PowerPoint), which allows the tracking of specific content fragments (to track non-text content fragments in PowerPoint it was also required to use the shape interface [20]). At the moment, the IDs to identify the content fragments are saved in a separate XML file additionally to the proprietary document file.

Fig. 4 shows a screenshot of the PowerPoint plugin inside a running application. Beside the buttons for commit and update, there is structure outline widget on the right side which is used to interact with the central super-structure and the shared content.

5. Related Work

The document model DITA [21] offers a framework to create domain specific content schemas. Thus, it is possible to define new structure types to support individual content expressions. This solves the problem of the limited expression variety of the most single source authoring environments. With the DITA Open Toolkit [22] it is possible to publish DITA content into many deliverable formats. Reference [23] describes an authoring system for educational materials using these technologies. Unfortunately, this publishing system has the same conceptual problem of the strict transformation rules. Nevertheless, with DITA maps it is possible to use content references, different structures for the same content and conditional processing of alternative content fragments. It appears that it would be worthwhile to consider if DITA could be used for the implementation of the central data store.

There are approaches like [24] and [9] which are able to import contents from media-specific documents into a single-source base. The problem is that this is only a one-way solution. Changes in the single-source contents cannot be reassigned to the existing media-specific file.

LaTeX [25] can also be used, to create multiple media forms. This publishing step has the same conceptual problem of the strict transformation rules. It is not possible to create free layouts.

6. Conclusion

The presented new approach and the working prototype show that it is possible to connect media-specific authoring with the advantages of single-source data management. The expressivity of the output media files is not limited by the underlying single-source management. This concept takes the separation of content and formatting to a new level, because it allows media-specific designing without the restrictions of traditional single-source systems. Due to the concept of the abstract structure items and the content referencing to external content materializations it is possible to support all kinds of possible content depictions. This is important to manage the structure and the content for multimedia files at a central place.

References

[1] Moore, D. M., Burton, J. K., & Myers, R. J. (2004). Multiple-channel communication: the theoretical and research foundations of multimedia. In D. H. Jonassen (Eds.), *Handbook of Research on Educational Communications and Technology* (2nd ed.) (pp. 994). Mahwah, N.J.: Lawrence Erlbaum.

[2] Veglis, A. (2008). Comparison of alternative channels in cross media publishing. *Pub Res Q, 24(2),* 112.

[3] Myrach, T., & Röthlin, M. (2002). Cross-media-publishing of materials for e-learning. *Proceedings of the 4th International Conference on New Education Environments* (pp. 39-42). Retrieved October 29, 2014,

from http://www.opess.ie.iwi.unibe.ch/download/ICNEE02-TM-MR.pdf.

[4] Walsh, M. (2009). Pedagogic potentials of multimodal literacy. In W.-H. T. Leo, & R. Subramaniam, (Eds.), *Handbook of Research on New Media Literacy at the K-12 Level: Issues and Challenges* (pp. 32-47). Hershey, PA: Information Science Reference.

[5] Fahy, P. J. (2008). Media characteristics and online learning technology. In A. Terry (Ed.), *Theory and Practice Of Online Learning* (2nd ed.) (pp. 137–174). Edmonton: AU Press.

[6] Rockley, A., & Cooper, C. (2012). *Managing Enterprise Content: A Unified Content Strategy (*2nd ed.) (pp. 50). Berkeley, CA: New Riders.

[7] Signer, B. (2010). What is wrong with digital documents: A conceptual model for structural cross-media content composition and reuse. In J. Parsons, M. Saeki, P. Shoval, C. Woo, & Y. Wand (Eds.), *ER'10 Proceedings of the 29th International Conference on Conceptual Modeling* (pp. 391–404). Berlin: Springer, 2010.

[8] Closs, S. (2011). *Single Source Publishing: Modularer Content for EPUB & Co.* (pp. 26). Frankfurt, M: Entwickler Press.

[9] Walsh, L. (2007). Using extensible markup language (XML) for the single source delivery of educational resources by print and online a case study. *AACE Journal, 15(4)*, 389-411.

[10] Mackenzie, J. (2011). *The Editor's Companion* (2nd ed.) (pp. 92). Port Melbourne, Vic, New York: Cambridge University Press.

[11] Sapienza, F. (2002). Does being technical matter? XML, single source, and technical communication. *Journal of Technical Writing and Communication, 32(2)*, 157.

[12] Schaeffer, B. (2012). *Single Source Publishing: Creating Customized Output.* Retrieved October 29, 2014, from http://www.cmswire.com/cms/information-management/single-source-publishing-creating-customized-output-015069.php.

[13] Kramer, R. (2003). Single source in practice: IBM's SGML toolset and the writer as technologist, problem solver, and editor. *Technical Communication, 50(3)*, 328–334.

[14] Pötzsch, F. S. (2007). Bringing the Evidence: Iconography and Pragmatics of Powerpoint Slides. In B. Schnettler, & H. Knoblauch (Eds.). *Powerpoint-Präsentationen: Neue Formen der Gesellschaftlichen Kommunikation Von Wissen* (pp. 86–103). Konstanz: UVK Verlagsgesellschaf.

[15] Brumbaugh, R. S. (1991). *Plato for the Modern Age.* (p. 63). Lanham, MD: University Press of America.

[16] Pilato, C. M., Collins-Sussman, B., & Fitzpatrick, B. W. (2008). *Version Control with Subversion* (2nd ed.) (pp. 14).Sebastopol, CA: O'Reilly Media.

[17] Koch, D. (2008). *XML: Getting Started for Sophisticates.* (p. 17). München: Addison Wesley in Pearson Education Deutschland.

[18] Microsoft Developer Homepage Word. Range interface. Retrieved October 29, 2014, from http://www.msdn.microsoft.com/en-us/library/microsoft.office.interop.word.range.Aspx.

[19] Microsoft Developer Homepage Power Point. Text Range interface. Retrieved October 29, 2014, from http://www.msdn.microsoft.com/en-us/library/microsoft.office.interop.powerpoint.textrange.aspx.

[20] Microsoft Developer Homepage Power Point. Shape interface. Retrieved October 29, 2014, from http://www.msdn.microsoft.com/en-us/library/microsoft.office.interop.powerpoint.shape.aspx.

[21] Darwin Information Typing Architecture (DITA) Version 1.2. (2010, December 1). OASIS Standard. Retrieved October 29, 2014, from http://docs.oasis-open.org/dita/v1.2/spec/DITA1.2-spec.html.

[22] DITA Open Toolkit Homepage. Retrieved October 29, 2014, from http://www.dita-ot.github.io.

[23] Haubold, T., & Klenner, M. (2012). An approach for publishing multi-lingual educational materials for various forms of representation. In J. Kawalek (Ed.), *Tagungsband: 10. Workshop on e-Learning (WeL '12)*

(pp. 57–67). Zittau: Hochsch.

[24] Hamilton, M. (2008). *MadCapFlare–An Introduction to Topic Based Authoring: (Part 1)*. Retrieved October 29, 2014, from https://www.madcapsoftware.com/blog/wp-content/uploads/2008/10/doctrain-flare-training-a.ppt.

[25] Sojka, P. (2007). Single-source publishing in multiple formats for different output devices. *Proceedings of the Euro Bacho T E X 2007, XVII,* (pp. 118–124). Retrieved October 29, 2014, from http://www.tug.org/TUGboat/tb29-1/tb91sojka.pdf.

Michael Klenner received the M.S. degree in computer science from University of Applied Science Zwickau, Germany in 2011. He is currently pursuing the Ph.D. degree. The dissertation project was funded by the European Union and the Free State of Saxony. His main topics of interest are software engineering, media specific authoring and international communication.