Cross Media Publishing of MediaWiki Content

Thorsten Kisner
Management and Engineering
AHT GROUP AG
Essen, Germany
Email: t.kisner@aht-group.com

Abstract—Wikis are excellent tools for distributed authoring, knowledge management and collaboration. Even for inexperienced users the used markup syntax is easy to understand and tends not in an obstacle to participate in the authoring process. Widely used in universities, enterprises and communities, Wikis represent a web based content management system for collaborative working and authoring. In many situations users want to have the content of a Wiki or selected articles available in different types of media other than the web based online version. This is the domain of cross media publishing systems being able to create several output formats from one single source. In this paper we present a module of the cross media publishing framework openFuXML to integrate Wiki content and render it to different output formats.

I. INTRODUCTION

Wikis have become a widely used tool in the Internet, universities, schools and enterprises since their invention in 1995 with a significant impact on society with Wikipedia. A impressive increase of Wikis as a platform for collaborative knowledge sharing [1], ranging from some loosely connected articles up to vast databases with hundreds of thousands of articles and dense connections between these, has taken place during the last ten years.

In the area of education, projects like Wikibooks or Wikiversity are taking advantage of the community to produce textbooks, course materials or curricula to support and organize educational activities.

The impact of Wikibooks in the textbook industry is described in [2] in the context of community processes. It is stated that communities working on Wikibooks are currently only loosely connected and mostly unorganized; nevertheless these communities are capable to produce reasonable results under these conditions.

Especially with hyperlinked articles and semantic links between these Wikis are a significant step forward from collections of printed material, although traditional Wikis only provide the direct view on one article. Often a user wishes to combine several articles or include articles in his own content and produce a document with a consistent layout: A project manager wants to compile a dossier with several CVs available as individual articles, a tourist wants to create an individual sightseeing guide of several points of interest available in Wikipedia or an author who wants to include a Wiki article in his document or compile a document from several Wiki articles. In general there are several situations where users want to make a snapshot for a collection of articles with a specified revision, e.g. a release version of a software documentation or report and provide this as a printout.

After deciding to create the document containing several Wiki articles, authors are confronted with the question how to do this effectively. The simplest way of integration ("Copy & Paste") is sufficient for single articles. For content structures basing mainly on Wiki sources with dozens of articles, this becomes a cumbersome task, even more if this must be repeated after changes of content in the future. This paper introduces a module of the cross media publishing framework openFuXML [3] doing this automatically by converting Wiki articles into XML and utilizing the framework to create high quality textbooks (PDF), web pages (HTML) or ebooks (EPUB) from the same content repository.

The remainder of this paper discusses the related work in Section II. Section III introduces the cross media publishing and authoring framework openFuXML and is followed by discussion of integration Wiki content from a technical point of view in Section IV. Section VI presents results and ends with an overview on future work.

II. RELATED WORK

While we are not aware of other work extracting content from Wiki to XML with the objective to apply cross media publishing systems for ebook or high quality textbook production, several approaches exist to access and process content stored in Wikis.

General work in extracting information from unstructured content [4], extracting lexical knowledge [5] from Wiki articles as well as using natural language processing systems for Wiki content [6] has been carried out.

A wide range of bots or bot frameworks exist for automatic or semi-automatic content editing of Wiki text. This includes tasks such as spell checking, link checking or finding duplicates or disambiguation in the database.

Plugable rendering engines are also related to our work. E.g., the Wiki markup rendering engine Radeox is proposed in [7], this project aims to provide a generic translation of wiki markup written in Java. The engine VersoWiki is implemented in PHP and allows the transformation of Wiki markup to HTML and back. Common features of other PHP based engines like PmWiki or TextWiki are the usage of regular expressions and HTML as the primary output format.

Notable work was undertaken in [8] to create a data interchange format for page content of MediaWiki. A Document
Type Definition (DTD) is proposed (and implemented) to describe the page content, however this is only done for meta information like revisions, dates etc. The content itself is not transformed into XML.

Also notable is the built-in function “Printable Version” or “Add Page to Book” of MediaWiki. While the first function creates merely a HTML page without navigation elements, the second function allows users to create a PDF containing several articles. These generated PDF documents lack of customized tables of content, and other editorial elements like lists of figures, tables or index registers. The image quality is equal to the screen resolution only and formulas are created (with poor quality) as images. While these shortcomings may not be recognized by inexperienced users, they will soon realize that the content cannot be modified, reordered or extended. These serious restrictions for the creation of different output format are addressed in this paper.

III. THE CMP FRAMEWORK openFuXML

A. Cross media publishing

Cross media publishing (CMP) describes a publishing process that creates different types of media (e.g. books, cd-rom, web pages, e-books) from one single source (“single source publishing”). The content is managed and stored in a media independent manner, text elements are often managed using the Extensible Markup Language (XML). Images are available often in the source format, openFuXML maintains two different formats: (i) a scalable vector format (Encapsulated Postscript (EPS) or Scalable Vector Graphics (SVG)) optimized for printing or further processing and (ii) a raster image (Portable Network Graphic (PNG)) for screen output.

Using this concept of a single source, a cross media publishing process is able to render the content automatically for different types of target formats. This results in both a flexible and cost efficient way to store and maintain content repositories, since only a single repository has to be managed – thus avoiding redundancies of the same content for different media types.

B. History

The cross media publishing framework openFuXML became accessible in February 2007 and was initially presented at ED-MEDIA 2007 in Vancouver [9]. The development process started at the University of Hagen, Germany, in 2003, when a new development of a genuine cross media publishing system was agreed upon following extensive evaluation of existing systems. None of those was assessed suitable for publishing by experienced users, they will soon realize that the content cannot be modified, reordered or extended. These restrictions for the creation of different output format are addressed in this paper.

IV. THE openFuXML WIKI ENGINE

A. Wiki Markup Transformation

We worked together with professors from different faculties to be able to offer special subject environments for particular target groups such as mathematicians, lawyers, social scientists, engineers and computer scientists. The available didactical elements cover a wide range of concepts, e.g. examples, hints, exercises and solutions, author information, prerequisites and learning objectives.

Listing 1 shows a simple Wiki article only containing a few headers on different levels. Knowing the rules of applying different levels of headers with the symbol “=”, one directly understands the structure.

Listing 1. MediaWiki markup

==The openFuXML Wiki Engine==
==Wiki Markup Transformation==
====Wiki Example====
====openFuXML Example====

The transformation of this Wiki markup to XML is shown in Listing 2 and described in detail in [10]. The hierarchical and semantic representation in XML is much more complex, but its advantages can easily demonstrated. If one wishes to
include the Wiki article in Listing 1 into a pre-defined subsection, all header depth levels need to be modified manually. In XML (Listing 2) the section element can directly be inserted (or referenced) within another section.

Listing 2. MediaWiki markup transformed to XML

```xml
<section>
  <title>The openFuXML Wiki Engine</title>
  <section>
    <title>Wiki Markup Transformation</title>
    <section>
      <title>Wiki Example</title>
      <section>
        <title>openFuXML Example</title>
      </section>
    </section>
  </section>
</section>
```

The Content Transformation process includes text and media objects. If source images are available as Scaleable Vector Graphic (SVG), this format is converted lossless into EPS and used for the PDF output rather than just copying low quality bitmap images. Mathematical expressions are extracted from the “alt” tag and thus available in Latex notation for further processing. The output of this step are different content objects, mostly articles (ofx:section) with text, images, links and references.

B. Template and Macro Transformation

The definition of templates and environments is a sophisticated way for users and operators of Wiki servers to customize the layout of the pages. On the one side this is the worst-case for external rendering or transformation engines, because users are free to define whatever they want in these templates and the actual presentation is often done in conjunction with CSS. On the other side, there is semantic information in the form of key-value pairs available. Obviously, the rendering engine needs to understand these information to be able to render it in the right way.

Beside a generic Template Transformator mapping the key-value pairs to a simple table, users a free to decide how a template will be processed. In the configuration file a unique transformation class can be chosen for each template. The outcome of this transformation class may either a valid XML document with the namespace of the authoring framework or a intermediate XML format with an arbitrary namespace which is translated in further processing to a openFuXML valid document.

V. CONTENT COMPILATION

A. Structure Definition

All references to external Wiki content are defined inside the container element wiki:content. The most important elements are wiki:page and wiki:category:

- wiki:page This element represents an article (a Wiki page), which will be inserted at this position. The child element has to be a ofx:section, the optional attribute transparent="true" will insert the content of the article at this position and not create a own section with the complete article.
- wiki:category All articles tagged with the given category are summarized. Either as individual sections or compressed a synoptical table.

Document structures are defined in XML with the openFuXML authoring framework. Listing 3 (with partly omitted XML namespaces) shows an example of such a definition.

Listing 3. Example of XML structure definition

```xml
<xml version="1.0" encoding="UTF-8"/>
<fx:ofoxdoc>
  <fx:metadata>
    <title>HelloWorld</title>
  </fx:metadata>
  <fx:content>
    <section source="text/introduction.xml"/>
    <section>
      <title>Use Cases</title>
      <wiki:content>
        <page name="Category:Use_Case" depth="0"/>
      </wiki:content>
    </section>
  </fx:content>
</fx:ofoxdoc>
```

The document has two top-level sections:

- The section Introduction demonstrated the inclusion of a XML document as a external source. The directive will be completely replaced by the content in the external document.
- The section Use Cases starts with the content of the Wiki page Category:Use_Case and is followed by all pages labeled with the category Use_Case as individual sub-sections.

B. Content Compilation

The structure definition document is the source for the process of Content Compilation. In a couple of steps, all external MediaWiki servers are contacted and the defined articles are locally stored for further processing.

1) Since all container elements can be defined as references (with external=true and a source attribute), the root document is parsed and all external content elements are merged into the root document.
2) The document at this stage contains the complete local content. In this step the document is parsed for external wiki:content Elements (see Listing 3) and the element is replaced by a local (external) content element pointing to a file.
3) Depending on the configuration (never, always or if the Wiki page is updated), the Wiki page is requested from the server and saved as Wiki markup.
4) The Wiki markup is transformed to XML, this step is explained in Section IV-A.
5) The newly created external elements are merged into the root document.
6) References are processed with customizable strategies for internal and external links.
The resulting XML document contains all content information and can be processed in the rendering engines for cross media publishing. Depending on the target format and its configuration, editorial elements like tables of content, figures or lists are available. Didactical and subject elements can be generated by individual Template Transformators. The process of content compilation and cross media publishing is outlined in Figure 2.

A textbook can be interpreted as a special case of a graph $G = (V, E)$ with $|V| = n$ nodes and $|E| = m$ edges called Tree. A node is equivalent to a section or subsection and an edge represents the relationship “is parent of”. The additional constraints (i) there is exactly one path between two nodes, (ii) $G$ is (minimal) fully connected with $m = n - 1$ and (iii) $G$ is (maximal) acyclic (i.e. the no cycle at all) describe a tree [11] shown in Figure 3.

C. Content Structuring

A general issue for all tools and algorithms trying to transform a collection of hypertext documents into a linear book is the ordering of different pages in the book. Let’s assume the root page $R$ of the hypertext collection points to pages $A$, $B$ and $C$, and they respectively point to pages $A_1$, $A_2$, $B_1$, $B_2$, $B_3$, and $C_1$. Should pages ordered in depth first fashion $R$, $A$, $A_1$, $A_2$, $B$, $B_1$, $B_2$, $B_3$, $C$, and $C_1$ or in a breadth first fashion $R$, $A$, $B$, $C$, $A_1$, $A_2$, $B_1$, $B_2$, $B_3$, $C_1$?

Event with this simplified example it’s easy to find arguments for and against each of the strategies, with additional links e.g. from $A_1$ to $B_2$ or $A_1$ to $B_2$ to $C_2$ to $A_1$ both strategies will fail.

One solution implicitly used in Figure 2 is a manual ordering. This is implemented for the wiki:page and wiki:category directives.

A collection of hyperlinked Wiki articles can be described as directed graph with an ordered pair $G = (V, E)$ with $V$ as a set of elements called vertices or nodes, and $E$ a set of ordered pairs of nodes called directed edges.

This theoretical descriptions reflect our intuitive understanding of the structure structure of a textbook and wiki articles and is summarized in Table I.

The required structure\(^1\) for a textbook and the graphical representation of a tree is outlined in Figure 3.

Figure 4 shows an example of linked hypertext documents. With the graph metrics in-degree and out-degree it’s possible to identify the node $A$ as a root node and $F$ as a leaf. But with this information it’s not sure if $F$ should be a child of $A$ or $D$. Cyclic links (shown with $B$, $D$ and $E$) must be detected and resolved during the transformation to a tree-like structure.

A couple of algorithms and heuristics are implemented to support users with the compilation of hyperlinked articles to textbooks. The directive wiki:page supports the attribute depth to specify the number of links which should be followed during content processing. Another metric is the distance to manual defined root elements $d_R$. This is demonstrated in

\(^1\)Of course this structure does not restrict the usage of internal references by authors “see section x.y”
Algorithm will attach it to $R_2$.

Figure 5: The node $C$ might belong to $A$ or $R_2$, but the algorithm will attach it to $R_2$ because $d_{R_2,C} < d_{R_1,C}$.

Wiki pages can be labeled with meta-information like categories. If this information is available, a clustering algorithm can be applied trying to cluster all pages with the same meta-information to the same parent. The user can specify an ordered list of key-words to give them a high priority for the algorithm.

Without semantic information, the local clustering coefficient $c_i$ [12] can be used. The connections of all neighbors of node $i$ to all possible connections among each other is represented by $c_i$, a high $c_i$ for hyperlinked articles suggests that articles with common neighbors tend to be neighbors, too.

VI. CONCLUSION AND FUTURE WORK

This approach combines the advantages of both environments: the simplicity of markup and authoring of Wikis and the flexibility of structured XML content objects for cross media publishing. The system is successfully tested to process a internal knowledge management Wiki, a Wiki for software development documentation and learning materials created in a distributed authoring process. It allows authors with basic IT knowledge to participate in the development process directly with the Wiki. Editors can work with the XML based authoring framework and directly include single articles, multiple articles or a selection of articles based on categories. The resulting document can be processed by the cross media publishing system openFuXML and rendered to HTML, PDF or EPUB. Based on XML, a transformation to other schemas, like DocBook or individual formats is possible.

Beside Wiki markup itself, the processing of custom templates is the most important step for reasonable results. Future work will focus on templates processing, creation of generic transformations and a repository for custom transformation classes, e.g. for templates used in Wikipedia or individual user generated templates.

REFERENCES


