IADIS International Journal on Computer Science and Information Systems
Vol. 6, No.1, pp. 94-109
ISSN: 1646-3692

FORMAT-INDEPENDENT SEGMENTATION AND CONVERSION OF DIGITAL DOCUMENTS[1]

Angelo Di Iorio, Antonio Feliziani, Luca Furini and Fabio Vitali
Department of Computer Science, University of Bologna

ABSTRACT

Enterprises need to produce a lot of digital documents: from internal reports to official documentation, from confidential data to public advertisements, from customers' letters to worksheets, and so on. The process of editing and publishing typographically sophisticated documents is difficult and very often outsourced. The production of such documents, in fact, still requires competencies and resources that only professional typesetters can provide. In this paper we present a flexible architecture designed either to improve professional publishing processes or to make it easier authoring and publishing for inexpert users: combining automatic content extraction with advanced formatting techniques, we automatically produce high-quality documents, in particular PDF files. Actually, both input and output documents can be expressed in different data formats. IsaPress is a java implementation of such architecture, based on a set of format-specific smart parsers, XSLT meta-stylesheets and an improved version of FOP.

KEYWORDS

Digital publishing, multi-channel publishing, format conversion, ISA, IsaPress.

1. INTRODUCTION

The production of *professionally formatted documents* is a complex task. In fact a heavy duty and cost intensive back office intermediates between the output of the author (most likely, MS Word files) and the input to the press (most likely, PDFs). Some of the activities of the editorial staff impact on the actual content of the book (such as proof reading, grammar check, etc.), some is legal/commercial, but many activities have much to do with the preparation of

[1] This paper is an extended version of: A. Di Iorio, AA. Feliziani, L. Furini, F. Vitali, "A Cross-format Architecture for Professional Publishing", IADIS International Conference on Applied Computing 2010, Timisoara, Romania.

the final deliverable ready to go to the presses through pagination applications such as InDesign (InDesign 2011) or XPress (XPress 2011).

The output requirements in these situations are precise, high quality and imperative. This makes it impossible to use widespread word-processors to directly produce books ready for printing and distribution, although authors still prefer to write content with these tools, that are very simple, intuitive and not expensive (or even free). The presence of converters that produce PDFs from MS Word files does not solve this issue. Several of these tools are available, either commercial (CutePDF 2011) or free (Bullzip 2011). This list could go on and on. However, the output generated by such tools does not meet the requirements of a professional publishing house: the conversion models currently adopted are crude and approximate (Di Iorio et al. 2006).

The challenge is to provide a fully automatic conversion engine that generates high-quality results from arbitrary sources, removing the need for intermediaries to convert the input and revise the converted result.

In this paper we present a general framework that makes this process take place, focusing on the context of professional digital publishing. The overall framework relies on three main components: (i) a generic intermediate data format, called IML, (ii) a generic conversion architecture called ISA* and (iii) a sophisticated pagination engine. The paper is actually focused on the first two components with the intent of showing either their generality or their applicability to real scenarios. In fact, we go into details of IsaPress a cross-format architecture for professional publishing currently in use by an Italian Publishing House to produce high-quality books. Further details about the pagination engine are in (Di Iorio et al. 2006).

The paper is then structured as follows. Related works are investigated in section 2. Then we introduce our segmentation model and IML in section 4, and discuss ISA* in section 5. Section 6 is devoted to IsaPress, before conclusions in section 7.

2. RELATED WORKS

A basic principle is well accepted by the document engineering and markup languages community: the separation between content and format. This idea is so embedded and shared by the community that providing a complete list of references is impossible (canonical references are (Coombs et al. 1987) (SperbergMcQueen et al. 1997)). On the opposite side, we found very interesting ideas about the impossibility of separating content and presentation and actually segmenting documents reusable subcomponents. Hillesund (2002) argued that there is no way of separating content and presentation but they are strictly interconnected and mutually dependent. He consider the paradigm of XML “one input – many outputs” basically wrong and claims that is can be only substituted by a weaker “many inputs – many outputs”. Indeed it is practically impossible to reuse content fragments and merge them from different sources into a good composite one. Walsh (2002) replied that position by stating that separation is possible and by holding DocBook as example of the success of such distinction.

Unlike the global critique of Hillesund, Piez (2005) argued that in some context it could be useful and profitable to write documents taking in mind both content and presentation, and managing them as a whole unit instead of separated sub-components. According to the author, “there is no reason to fear or disdain presentation-oriented design, but designers need only to discriminate when they want and need an isolated layer for our information capture, and when they want to work more directly with the ’hot lead’”.

An approach to segment and recombine content and presentation may consist in defining a rich document model, that includes structure information and presentation styles together with the document data; in this scenario, the application handling this kind of documents could be something very lighter and simpler, as it should only apply the logic contained in the document. An example of this “decentralized” approach is the Personalized Print Markup Language format [34], supported by the Print on Demand Initiative. PPML supports Variable Data Printing jobs: a PPML document defines a page model with both fixed content and “copy holes” placeholders, where the actual variable data will be placed. The advantages, besides the VDP capabilities that represent surely the main feature, include the possibility of using different formats together, as each copy hole could accept a different kind of content, while overcoming the deficiencies of each format.

The Document Description Framework (Lumley et al. 2005) follows the same approach, increasing the expressive power by adding programmatic behavior to the documents. A DDF document is composed of three different sections: data, logical structure and presentation. Structure and presentation are defined using templates, transforming respectively the data into a structured XML format, and the structured document into a suitable presentation format. This approach allows and encourages the re-use of data, structure and presentation, as there could be DDF documents defining just one of these aspects, that are merged together in order to obtain a completely instantiated document where a particular set of data is bound to a specific structure and some styles.

Particularly interesting for an environment still heavily dependent on the circulation of physical copies is also this research (Norrie et al. 2005) about a content publishing framework for interactive paper documents, describing the use of special devices such as "digital pens" to interact with digital contents by means of printed materials. In other words, a particularly generated printed version of a document could be seen not just as a one-way output but as an input channel too, adding the advantages of digital information storage to the versatility of a paper sheet.

3. DOCUMENT SEGMENTATION

To decouple the authoring process from the actual production of high-quality output, we propose a document segmentation model that expresses the most relevant constituents of a document, and upon which we have implemented advanced applications of document conversion.

3.1 The Pentaformat Model

Documents are traditionally segmented into *content* and *presentation* and, although opposite opinions exist (as discussed in the previous section), researchers and professionals agree on advantages of such approach. We refine this distinction by identifying five components that can be extracted from *any* document, regardless of its actual layout and presentation.

Table 1. The pentaformat model

Dimension	**Description**
Content	The plain information made of text and images (we mainly focus on these elements, and leave out audio and video for the moment).
Structure	The labels used to make the meaning and the logical organization of the content explicit. Both structure and content constitute the basic information written and organized by the author.
Presentation	The set of visual features added to maximize the impact of the document on human readers. Presentation is built over the structures and aims at strengthening what is inherently expressed by structured content.
Metadata	The set of information that make a document searchable, indexable and manageable within wider contexts.
Behavior	The set of dynamic actions triggered by events on a document.

Our model is called “Pentaformat” and summarized in Table 1. Our claim is not only that *any* document can be considered as the integration of those five dimensions, but also that they are clearly distinguishable from each other, and can be interchanged and reformulated to obtain different documents. In order to better explain the nature and impact of our segmentation model, some properties of these dimensions are discussed below:

- *Logical separation*: we consider each dimension as a partial perspective onto the same document. Each dimension provides specific information (orthogonal to all others), is created through the help of specific competences and has a specific role for the overall meaning of the document itself. Note that talking about *logical separation* does not mean these components are always created separately and by different users (on the contrary it is very common to find them intermixed); rather, it means they can be *abstracted* and separated *a posteriori* to express different kinds of information about the same source document.
- *Mutual connection*: from a different point of view, these dimensions are strictly connected. They are built on the top of each other, and they “work together” for the overall meaning of the document. No dimension makes much sense when examined in isolation.
- *Context-based relevance*: no hierarchy is imposed *a priori* over these dimensions, but they are equally important from a theoretical point of view, although the content can be probably granted some primality since it is the basic information upon which everything else is built. It is the context that determines the relevance and replaceability of the other dimensions: for instance, presentation is very important for professional publishing but can be discarded in data mining applications, while metadata can be neglected in pagination processes, or dynamic information such as behavior is useful only in a few contexts.
- *Context-based interchangeability*: depending on the context of use of the document, these components can be replaced with new ones. For instance, we can use the structure to fit the content into a completely different presentation, or express a set of metadata into a completely different vocabulary, or transfer a set of dynamic behaviors onto a completely different platform, etc.
- *Language independence*: information captured by each dimension can be expressed in different languages. However, the actual instantiation into a specific format does not influence the meaning of that information. Yet, the capabilities of

a specific language limit what can be encoded with that language because of syntactical details independent from abstract specifications. For instance, structural elements (such as paragraphs, lists, tables, etc.) can be translated into HTML tags, TEI [36] constructs or any other encoding language, while presentational information (such as box, lines, colors) can be translated into SVG constructs, XLS-FO primitives, formatted blocks and so on. The same considerations can be obviously extended to behavior and metadata. In conclusion, a cross-dimensional property is necessary to complete our model: the *language* each dimension is expressed in.

So the real point of our work is to be able to separate and extract all constituents of a document so as to reformulate a few of them, or to reuse some of them in different contexts. To this end we look at producing *generic formats* that describe the specific constituent elements of each document, so as to facilitate understanding and reuse. A generic format is therefore a set of elements describing the relevant bits of the documents in terms of content, structure, presentation, behavior and metadata, although in this paper we will only refer to the first three dimensions.

3.2 IML: Intermediate Markup Language

From the multitude of languages, formats and documents we daily work with, we might conclude that a huge amount of complex and diversified structures are needed. In (Di Iorio et al. 2005) we proposed and discussed some patterns for descriptive documents, concluding that by adopting these and only these patterns, authors can write well-structured, complete and unambiguous documents easily. Table 2 summarizes these patterns.

Table 2. Patterns for expressing structures of digital documents

Pattern	**Description**
Markers	Empty elements, whose meaning is strictly dependent on their position
Atoms	Units of unstructured information
Blocks and inlines	Blocks of text mixed with unordered and repeatable inline elements that have the same content model
Records	Lists of optional, heterogeneous and non-repeatable elements
Containers	Sequences of heterogeneous, unordered, optional and repeatable elements
Tables	Sequences of homogeneous elements

That work helped us to demonstrate that a small set of structures is sophisticated enough to express most users' needs: it is always possible to write arbitrarily complex documents or to normalize existing ones into simplified versions by using only such a limited set of structural objects, yet still expressing the same information. These patterns can be adopted to design an abstract language to express the structured content of *any* segmented document. We created such language and called it IML (Intermediate Markup Language).

IML is syntactically a very simple language that maps these patterns into specific XHTML structures, and uses attributes to express extra properties. This approach is similar to the microformats (Khare 2006), which embed semantic information within texts, by using a set of simple and open tags and attributes. Unlike micro-formats, which use specialized tags for a specific context, IML is a general language that let users to express any kind of information by simply using few attributes. Instead of having a pre-defined and rich set of names which capture the meaning of a text, it proposes a flexible mechanism which can be used to model any content. At the end, few tags only compose IML: P (for blocks), SPAN (for in-lines), TABLE, UL, LI and DIV (for different containers) and few more, all characterized by the @class attribute.

Since the objective of IML is expressing only *structured content* (i.e., expressing the role of each text fragment), with no presentational information and with no information about behaviour or metadata, IML documents are simply a sequence of content objects that simply specify which pattern each object respects (e.g., whether the object is a block text, a container, a table or an inline) and which specific class it belongs to (which kind of blocks it is, which level of nesting it has, and so on).

Consider for instance a document containing an "important" paragraph. Each format has a different way to express that information, but it is clear that, for instance, a paragraph when style is important in MS Word, a fragment `<p class='important'>An important paragraph</p>` in HTML, and the fragment `<important>An important paragraph</important>` in XML are all conceptually equivalent. IML expresses that semantic meaning in a simil-XHTML syntax (`<p class='important'>An important paragraph</p>`), but any other syntax would be equivalent.

Yet, some scenarios cannot be directly modeled with a so simple schema such as mathematical formulas, or graphical fragments, of forms, or fragments written in domain-specific syntaxes and so on. IML does not directly address them but can be easily extended for customized domains. Even adding some attributes and tags, the whole structure of the document does not change, and the basic pattern-based constructs remain unchanged. What change are only some *local* names and components.

The innovation of IML does not rely on its tags and attributes, rather in the fact that a minimal and rigorous set of objects and rules actually made possible to implement automatic conversion and advanced publishing systems. Yet, some scenarios cannot be directly modeled with a so simple schema such as mathematical formulas, or graphical fragments, of forms, or fragments written in domain-specific syntaxes and so on. IML does not directly address them but can be easily extended for customized domains.

4. ISA*: A FLEXIBLE ARCHITECTURE BASED ON PENTAFORMAT

By combining IML and the segmentation model described so far, we have designed a simple architecture that can be (and actually has been) repeated for very different scenarios.

We call it ISA*, since it generalizes some ideas developed for our previous project ISA (Vitali 2003). ISA is a web application designed to simplify and speed up the creation of web sites. Authors write content in MS Word (and specify the role of each text block by styles) and the system automatically converts such content into graphically advanced pages, by exploiting associations between the layout area names and the content styles, previously created by a graphic designer. ISA transforms such information into an XSLT that, in turn, will apply the selected formatting to the original content. ISA* applies a similar approach to heterogeneous domains and formats.

Basically, this architecture (shown in Fig. 1) separates all components of a document, then it works separately on each of them and then recombines them again for the final output. The whole system consists of bi-directional converters from and to any existing data format we want to support.

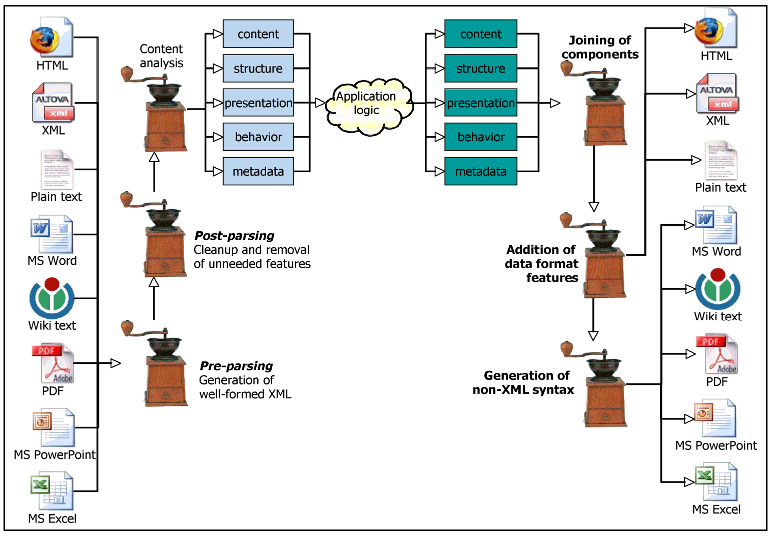


Figure 1. The ISA* architecture

Different models have been proposed for performing conversions between data formats (Mamrak & Barnes 1993): the *direct model* based on bidirectional transformations from a data format to another one, the *intermediate format model* based on a new format used as an intermediate representation of any format to be converted and finally the *ring model* in which data formats are virtually ordered in a circular structure and the transformation happens jumping from a format to the following one towards a pre-defined direction. The intermediate format model, a.k.a. *superior standard model*, has many benefits in terms of efficiency, quality and implementation facilities (Abiteboul et al. 1997) (Milo & Zohar 1998).

IML is therefore used as the intermediate data format that captures only relevant information of the input documents and ensures high-quality output by delegating the rendering to external powerful tools. Currently, the formats we manage include HTML, MS Word, ODF, PDF, LaTeX, plain text, as well as arbitrary XML. Note that IML is not meant to encode directly multimedia resources (such as EPS or JPG files, videos, etc.) that are referenced and attached to the document and need to be processed by external modules.

All document workflows using this approach follow the same general steps as shown in the picture: *content extraction* (on the left) and *high-quality post-production*.

4.1 Content Extraction

The first step consists of extracting all the constituents of a document and normalizing content in IML. That segmentation process can be further divided in three steps:

4.1.1 Pre-parsing

Since our architecture is intrinsically based on XML, our fundamental tools are converters from one XML format to another. While some of these formats are already based on XML, others require a further step. Thus, the first action is converting the source format into XML, before any further content accommodation or interpretation takes place. This operation, called *pre-parsing* is heavily dependent on the actual syntax being employed by the data format being converted and it is heavily different among data formats.

4.1.2 Post-parsing

After the pre-parsing step we read the resulting XML file, clean up the content and remove the parts that surely will not be needed for the smart component separation. These operations, cumulatively known as *post-parsing*, are also different across data formats, but for semantic rather than syntactical reasons. These include re-joining lines into paragraphs, or removing alternative variants of the same image. Although these operations are still dependent on the quirks and peculiarities of each data format, they are done on an XML source, and therefore they can be and are usually done via a XSLT transformation.

Note also that adding a new format to the architecture requires implementing a new pre-parsing and post-parsing processor. The overall approach is still scalable as only these two modules depend on the input data format while the actual conversion logic is generalized.

4.1.3 Content Analysis

Pre-analyzed content is then scanned to identify individual features and denominate the constituents. The output of this phase is the same XML document that was provided in input, with additional attributes specifying whether an element is content, presentation, behavior or metadata. The current engine is based on XSLT technologies.

The key point is that any document can be passed to the engine, without imposing constraints on its internal structures and styles. The approach follows a *"Garbage In, Garbage Out"* paradigm: no input file is rejected, but the better structured is the source, the finer the final output will be. Note also that the analysis is fully automated and no user action is required.

We experimented different possibilities to ease and improve the quality of this information extraction. One extreme is to impose strict rules onto the authors, possibly enforced with macros that verify if they are following them. In this case, editing is not free of hassle, but the conversion is perfect, simple and straightforward. The opposite extreme is to have the system accept just any document and do its best to extract the actual content; in this case, the complete freedom in writing has heavy impact on the sophistication/complexity of the converted result.

An intermediate solution, that we have tested in the e-learning context described in (Di Iorio et al. 2006b), consists in giving the users a set of guidelines about how to use styles and input macro, and then in implementing the appropriate transformations. Therefore, all documents can be processed by the system but the more compliant they are to the guidelines, the better will be the final result and the correct reformatting.

ISA* implements a hybrid approach. Whenever no guarantee of correct input is available, ISA* applies a set of heuristics and analysis techniques that allows users to provide even unformatted documents to produce well-formatted output (Vitali et al. 2004). These heuristics, expressed as parametric conversion rules, can be adapted so as to make this approach flexible for different scenarios and levels of complexity.

4.2 High-quality Post-Production

In the second step of ISA* architecture the perspective changes radically: what is an abstract description of content has to become an actual file, in a specific format, with specific formatting and layout. That process involves two clearly distinguished sub-processes: *application logic* and *high-quality rendering*.

4.2.1 Application Logic

Once the five constituents of the document are separated, each specific application can act on them independently. Operations can vary considerably, from simple ones to rather complex ones, depending on the purpose of the application itself.

A very short and incomplete list is shown below:

- Simply repackaging them in a different format in order to convert a document from one format to another
- Substituting the presentation constituent with a new one in order to reformat a document, regardless of its source, with a completely different layout and final aspect
- Analyzing the structure constituent looking for specific types of content in order to filter it out (e.g., removing advertisement from a web page)
- Adding specific ontology-driven elements to the metadata constituents so that the document can be correctly placed in a workflow process regardless of its source format.

It can obviously go on and on. It is worth noting, though, that all these operations are independent either of the input format or of the final one, since all files used by the application logic are all IML documents.

4.2.2 High-quality Rendering

Finally, all ISA* tools take care of re-generating a final document ready to be delivered to the final application. These processes follow a sequence absolutely symmetrical to the initial one: the new IML document is enriched with data format-specific information, and then converted via XSLT stylesheets into an XML format which can either be the final format, or the input to a converter to some kind of binary format, assuming we have the correct converter from XML to binary (such as a XSL-FO formatter).

Particularly important in this stage is the quality of the final conversion. The final rendering step takes in input both the converted document and configuration parameters that express the quality requirements to be met. By applying adaptive models, the renderer transforms the IML content into a ready-to-publish output, for instance a reusable and accessible learning object based on SCORM, or a sophisticated XSL-FO/PDF file.

Our model suggests then using external and format-dependent applications that are smart enough to take sophisticated decisions in order to generate high-quality results. The complexity inherent in such high-quality results depends also on the sophistication of the formatter that actually produces the final artifact. The more powerful and reliable is the renderer, the lesser is the effort required to produce high-quality products. However, in many cases the results that can be obtained with existing tools are not sufficiently sophisticated for professional use. For this reason, we often need to improve or re-implement renderers.

5. ISA* FOR PROFESSIONAL PUBLISHING: ISAPRESS

The abovementioned requirements are the milestones of IsaPress, an instantiation of the ISA* architecture for professional publishing. It is a system that automatically transforms unformatted content into ready-to-print and graphically advanced resources, in particular books. Assuring uniformity and high quality of their final products is not an easy and cost-effective task for publishing houses. Many manual interventions are still required to uniform source documents and make them ready to be "digested" by a (automatic or hybrid) conversion. The most widespread process to produce books involves different actors with different skills:

- *Authors*: they actually write content, ignoring the final formatting, and having few technical skills.
- *Publishers*: they decide the look&feel of the final product, in terms of formatting properties, dimensions, fonts, graphical choices and so on.
- *Pagination experts*: they transform the content provided by authors into a format ready to be processed and printed. They work with professional tools like InDesign (InDesign 2011) Quark XPress (XPress 2011) or PageMaker (PageMaker 2011) and perform some manual checks and corrections.
- *Typographers*: they actually print and bind the final books.

Note that we have omitted roles like proofreaders, reviewers, editors, etc. Our focus is not on the whole workflow of a publishing house, but rather on the semi-automatic conversion and publishing process.

In such a model, a leading role is still played by the pagination experts who are actually in charge of importing raw content in the system and verifying that they can be really transformed into a well-formatted book. According to the complexity of the final output, as well as the number of constraints to be fulfilled, such experts have even to manual intervene on the content and, when need, fix errors.

Fact is, software formatters are still limited and do not solve automatically the most complex issues, publishing houses require complex properties to be satisfied, content is very often unforeseeable and full of exceptions, but without the work of pagination experts many books would not be published.

The goal of our research is studying and implementing an automatic workflow that allows users to produce high-quality professional books from unformatted and raw text. The final goal is minimizing the manual effort of all the actors involved in the previous scenario, without sacrificing the quality of the results, up to completely dispose of the role of the pagination experts.

We started by defining a set of requirements that such a system needs to meet. In particular, we wanted to design a system that:

1. *Performs the whole process automatically*

2. *Minimizes authors' effort*
3. *Guarantees high-quality results*
4. *Guarantees flexible results*
5. *Allows user to further modify documents with a little effort*

Several benefits for writers, readers and publishers can be obtained by taking into consideration even multiple formats for input and output. The advent of non-traditional devices to read documents, the increasing importance of alternative publishing model like printing-on-demand, the success of alternative formats for electronic resources are clearly showing that publishing cannot be limited and entrapped by only paper printing. On the other side, the increasing success of new languages and vocabularies for text encoding, the maturation of tools for importing documents and the habit of using different editors by different users are clearly stressing on the importance of multi-channels in input too. However, advantages of multi-channel publishing have already been widely-discussed and outlined by researchers and professionals. For this reason, we even need a system that:

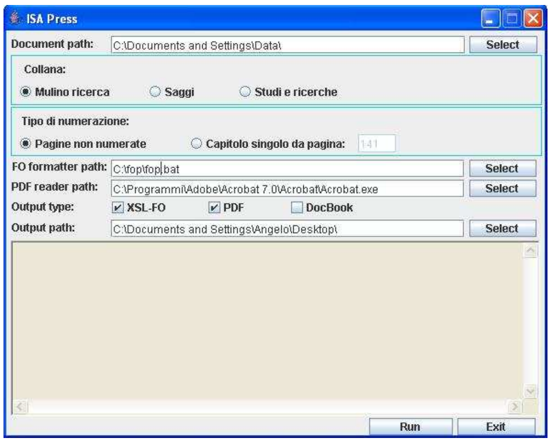
6. *Takes in input multiple data formats*
7. *Produces documents in different data formats*

A last point is equally relevant: since the lifecycle of a document does not end with its first publication (suffice it to mention further revisions, issues, translations and so on) we need to define a requirement that allows users to keep on working on the same document maintaining all the advantages of automatic conversion and publishing. Then, an ideal system should even:

8. *Produces documents in different data formats*

5.1 Improving the Publishing Workflow with IsaPress

IsaPress addresses issues and limits of a traditional publishing workflow, by completely automating the production of high-quality books. Fig. 2 shows the interface of the system.



Document path: C:\Documents and Settings\Data\
Collana: [✓ Mulino ricerca] Saggi Studi e ricerche
Tipo di numerazione: [✓ Pagine non numerate] Capitolo singolo da pagina: 141
FO formatter path: C:\fop\fop.bat
PDF reader path: C:\Programmi\Adobe\Acrobat 7.0\Acrobat\Acrobat.exe
Output type: [✓ XSL-FO] [✓ PDF] DocBook
Output path: C:\Documents and Settings\Angelo\Desktop\

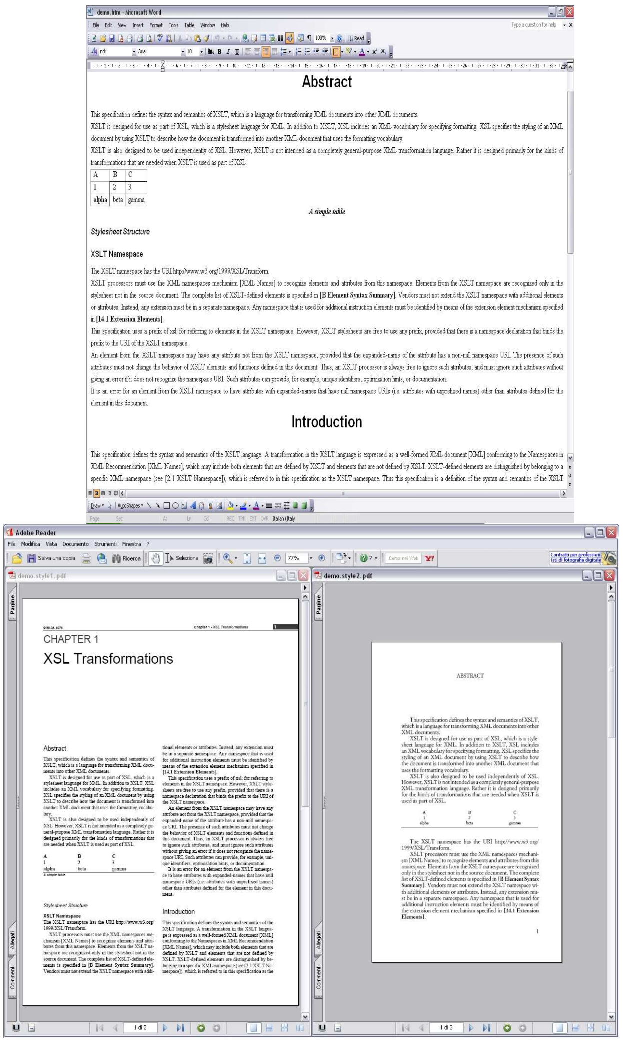
Figure 2. The interface of IsaPress (stand-alone Java application)

The current implementation can be actually deployed in different ways: a stand-alone Java application, a web application and a module of a legacy CMS. The internal conversion engine relies on the principles discussed so far: allowing users to write content by using their personal productivity tools, segmenting the content into abstract components among which an IML representation of content, and running an automatic templating process that produces the final result.

Although IsaPress currently supports different data formats either in input or in output, the initial conversion goes from MS Word or ODF (OpenOffice) documents into PDF files. Since it has been the main focus of our research for a long period (and the ground where we obtained the best results), as well as one of the most useful applications, we use this conversion to explain concepts, achievements and possible extensions of our work.

According to the ISA* architecture, the IsaPress process has two main steps:

1. *Content writing and extraction*: an author simply writes a document by using MS Word or OpenOffice. No particular plug-in is installed and no limitation is imposed over the tool features. The document is then processed by the IsaPress engine, which extracts its actual content and removes all presentational aspects, producing an IML file.
2. *High-quality post-production*: the intermediate IML document is then transformed into an XSL-FO file according to an XSL-FO template given in input. What is important here is the flexibility of the templating mechanism, which allows users to format the same content in very different ways without any further effort. It is enough to pass a different XSL-FO as input, and everything is automatically done by the system. At the end, the XSL-FO intermediate file is transformed into PDF, by exploiting the customized version of the FOP formatter described in (Di Iorio et al. 2006). The formatter in fact exploits some XSL-FO extensions and implements a revised Knuth algorithm (Knuth & Plass 1981) to automatically produce a ready-to-publish book.



Abstract

This specification defines the syntax and semantics of XSLT, which is a language for transforming XML documents into other XML documents.
XSLT is designed for use as part of XSL, which is a stylesheet language for XML. In addition to XSLT, XSL includes an XML vocabulary for specifying formatting. XSL specifies the styling of an XML document by using XSLT to describe how the document is transformed into another XML document that uses the formatting vocabulary.
XSLT is also designed to be used independently of XSL. However, XSLT is not intended as a completely general-purpose XML transformation language. Rather it is designed primarily for the kinds of transformations that are needed when XSLT is used as part of XSL.

A	B	C
1	2	3
alpha	beta	gamma

A simple table

Stylesheet Structure

XSLT Namespace

The XSLT namespace has the URI http://www.w3.org/1999/XSL/Transform.
XSLT processors must use the XML namespaces mechanism [XML Names] to recognize elements and attributes from this namespace. Elements from the XSLT namespace are recognized only in the stylesheet not in the source document. The complete list of XSLT-defined elements is specified in **[B Element Syntax Summary]**. Vendors must not extend the XSLT namespace with additional elements or attributes. Instead, any extension must be in a separate namespace. Any namespace that is used for additional instruction elements must be identified by means of the extension element mechanism specified in **[14.1 Extension Elements]**.
This specification uses a prefix of xsl: for referring to elements in the XSLT namespace. However, XSLT stylesheets are free to use any prefix, provided that there is a namespace declaration that binds the prefix to the URI of the XSLT namespace.
An element from the XSLT namespace may have any attribute not from the XSLT namespace, provided that the expanded-name of the attribute has a non-null namespace URI. The presence of such attributes must not change the behavior of XSLT elements and functions defined in this document. Thus, an XSLT processor is always free to ignore such attributes, and must ignore such attributes without giving an error if it does not recognize the namespace URI. Such attributes can provide, for example, unique identifiers, optimization hints, or documentation.
It is an error for an element from the XSLT namespace to have attributes with expanded-names that have null namespace URIs (i.e. attributes with unprefixed names) other than attributes defined for the element in this document.

Introduction

This specification defines the syntax and semantics of the XSLT language. A transformation in the XSLT language is expressed as a well-formed XML document [XML] conforming to the Namespaces in XML Recommendation [XML Names], which may include both elements that are defined by XSLT and elements that are not defined by XSLT. XSLT-defined elements are distinguished by belonging to a specific XML namespace (see [2.1 XSLT Namespace]), which is referred to in this specification as the XSLT namespace. Thus this specification is a definition of the syntax and semantics of the XSLT

Figure 2. A source Word document and two very different PDFs from the same file

Figure 2 shows an example of conversion performed by IsaPress: a MS Word file has been transformed into two very different PDF files, both ready to be printed.

As requested by the list of requirements discussed above, such a full automation is the core of IsaPress. Requirement (2) is even satisfied since users can upload not only MS Word but any kind of them. Allowing users to write content with his/her preferred productivity tools carries important benefits: firstly, authors do not need to learn using a new tool or editor, as simple it is; second, they can import and re-use existing materials with little effort. Yet, content verification and polishing could be required in order to make content ready for a book or a different output, but they are out of the scope of this paper. The automatic conversion from MS Word to PDF makes IsaPress meet even requirement (5): whenever a user needs to change something in the final resource he has to simply modify the original one and, then, start again the conversion process. Finally, by using XSL-FO templates and formatting them with an improved version of FOP, that guarantees high-quality pages, IsaPress meets even requirements (3) and (4). Details about the pagination and formatting of PDF files are provided in (Di Iorio et al. 2006) where we have shown how those requirements can be actually satisfied and how the whole process can be fully automated. Going on through the list of requirements, we stressed on the support for multi-format input and output.

In fact, we have generalized the original process of IsaPress in order to take in input even InDesign, DocBook files and produce even DocBook files and HTML pages, and new formats will be included in the near future.

5.2 Real-life use of IsaPress

IsaPress is not a prototype, but a working system used by an important Italian academic publishing house, called “Il Mulino”, in order to officially publish books. More than one hundred books and journal issues have been published in the last three years, covering different subjects (economics, law, etc.) and including different objects: from statistical and tabular data to plain text, from pictures to complex tables, from hierarchical subsections to boxes and footnotes.

The system has been also used to produce paper versions of e-learning material. Content is extracted from MS Word files and re-flowed into high-quality PDFs.

6. CONCLUSIONS

In this paper we have presented a general conversion architecture based on a simple, yet powerful, principle: segmenting documents into five components, working on them separately, and recombining them in order to obtain high-quality output.

In particular, content is normalized into an XML format, called IML, which provides a very small number of constructs, that are versatile enough to represent the relevant content information of any document. IML files can be obtained from a wide range of input formats through specialized and sophisticated conversions, mainly implemented using XSLT stylesheets, and can be later converted (adding the necessary presentation information) in a wide choice of output formats. In this "many input -> IML -> many output" wide picture, we presented with more details a specific end-to-end conversion, from (even bad-formatted) Word files to high-quality PDFs. IsaPress integrates a customized XSL-FO formatter presented in (Di Iorio et al. 2006).

Such approach will allow even inexpert users to produce high-quality material and, in conjunction with self-publishing systems like Lulu (Lulu 2011), will allow them to directly publish complete books from raw sources.

Yet our main focus remains on supporting professional publishers. A full integration of the traditional workflow relying on well-known commercial tools is planned. Furthermore, support for a wider set of input and output option are foreseen: not just only MS Word and ODF, but also DocBook, XHTML, InDesign, Quark XPress and others to be identified. The support for InDesign and XPress as both input and output formats for ISA*, in particular, will allow traditionally produced books to enjoy all IsaPress functionalities, i.e., it will allow completely automatic reformatting according to the rules of different book series to be applied to legacy pagination efforts, and it will allow pagination experts using commercial tools to receive and work on books generated via the IsaPress interface.

ACKNOWLEDGMENTS

We would like to thank the master and undergraduate students in computer science at the University of Bologna who contributed with their theses at the development of IsaPress. Furthermore we would like to thank the “Il Mulino” publishing house, and in particular Andrea Angiolini, Paolo Casarini, Stefano Manfredi and Lisa Zanarini for their contribution and interest in the development of the IsaPress application. We finally thank Paolo Marinelli and Gioele Barabucci for their valuable contribution to this research and to the development of IsaPress.

REFERENCES

Abiteboul, S. Clouet, S, Milo T. 1997. "Correspondance and Translation for Heterogeneous Data", Proceedings of ICDT'97.

Barnard, D. T. and Ide, N. M. 1997. The text encoding initiative: flexible and extensible document encoding. J. Am. Soc. Inf. Sci. 48, 7 (Jul. 1997), 622-628.

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

Di Iorio, A., Feliziani, A. A., Mirri, S., Salomoni, P., & Vitali, F., 2006. Automatically Producing Accessible Learning Objects. Journal of Educational Technology & Society, 9 (4), 2006, 3-16.

Di Iorio A., Furini L., Vitali F., 2006 "A Total-Fit Page-Breaking Algorithm with User-Defined Adjustment Strategies". In the *Proceedings of the IS&T/SPIE Annual Symposyum on Electronic Imaging*, January, 2006, San Jose CA, USA.

Di Iorio A., Gubellini D., Vitali F., 2005. “Design Patterns for Document Substructures”. In the Proceedings of Extreme Markup Conference 2005, August, 2005, Montreal, Canada.

Hillesund, T., 2002. “Many Outputs Many Inputs: XML for Publishers and E-book Designers”. *Journal of Digital Information*, 3, 2002.

Khare, R. 2006. Microformats: The Next (Small) Thing on the Semantic Web? *IEEE Internet Computing 10*, 1, 68-75.

Knuth DE, Plass MF., 1981. Breaking Paragraphs into Lines. *Software. & Practice Experience*. 1981; 11(11).

Lumley, J., Gimson, R, Rees, O. 2005. “A Framework for Structure, Layout & Function in Documents”, In *Proceesing of ACM Symposium on Document Engineering*, November 2–4, 2005, Bristol, United Kingdom.

Mamrak S.A., Barnes J., C. O'-Connell, 1993. Benefits of automating data translation, *IEEE Software*, July 1993, 82-88.

Milo T., Zohar S. 1998. Using Schema Matching to Simplify Heterogeneous Data Translation", *Proceedings of the 24th VLDB Conference*, New York 1998, 122-133.

Norrie, M.C., Palinginis, A. Signer, B., 2005. “Content Publishing Framework for Interactive Paper Documents”, DocEng’05, November 2–4, 2005, Bristol, United Kingdom.

Piez. W."Format and Content: Should they be separated? Can they be?: With a counter-example", 2005. In *Proceedings of the Extreme Markup Conference*, Montreal, Canada, 2005.

Sperberg-McQueen C.M. and Burnard L., 1997. A Gentle Introduction to SGML. In *Guidelines for Electronic Text Encoding and Interchange*, pages 13–36, 1997.

Vitali F., 2003. "Creating sophisticated web sites using well-known interfaces" in: *HCI International Conference*, Crete (Greece), 2003.

Vitali F., Di Iorio A., Ventura Campori E., 2004. "Rule-based Structural Analysis of Web Pages". In *Document Analysis System VI, Volume 3163 of Lecture Notes in Computer Science*, pp. 425-437, Springer Verlag, Berlin 2004.

Walsh, N., 2002: "One Input Many Outputs: a response to Hillesund". *Journal of Digital Information*, 3, January 2002.

Tools and Web Resources

Adobe InDesign, http://www.adobe.com/products/indesign/, last visited 14 March 2011.

Adobe PageMaker, http://www.adobe.com/products/pagemaker/, last visited 14 March 2011.

Bullzip, http://www.bullzip.com/products/pdf/info.php, last visited 14 March 2011.

CutePDF, http://www.cutepdf.com/, last visited 14 March 2011.

FOP: Formatting Objects Processor. Available at http://xmlgraphics.apache.org/fop/, last visited 14 March 2011.

Lulu, http://www.lulu.com/, last visited 14 March 2011.

Quark XPress, http://www.quark.com/,last visited 14 March 2011.