An XML/SVG platform for document analysis

L. Pierron and A. Belaïd
INRIA-LORIA
Campus scientifique BP 239
54500 Vandoeuvre-les-Nancy, FRANCE
email : {Laurent.Pierron,Abdel.Belaid} @loria.fr

In this paper, a DTD (Document Type Definition) XmlLayout for document layout analysis is presented. The paper will focus on its finality and on the role that it can play in a document recognition framework. At first, we shall show how this DTD can interact in the communication between the system processes, and second, we shall explain its implementation into C++ , and finally we'll describe its use in the visualization methodology based on the SVG (Scalable Vector Graphics) DTD.

Document analysis platform

The DTD XmlLayout is the main format of a generic and opened OCR platform. The document "XmlLayout" is the common element shared by all the processing modules. It is seen as an information repository used for saving and getting the document analysis module information. A lot of attention is made in its definition allowing developers to dispose of a very precise document analysis language for programming. In a certain way, it acts as a protocol for document analysis tools.

Figure 1 shows the interactions between the human operator, a number of analysis modules and a XmlLayout document.

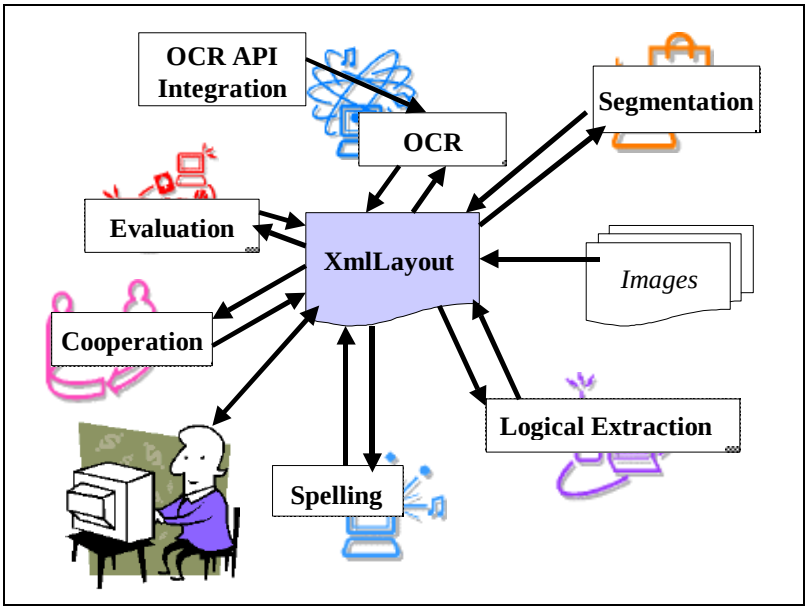

Figure 1 : Interaction between modules and XmlLayout

The human operator adds processing information in the XmlLayout document using a specific user-interface adapted to the job like segmentation, correction, spelling , etc. The modules read in the XmlLayout document the information needed for the processing. This interaction mode makes the direct interaction between the modules possible.

XmlLayout DTD

Figure 2 presents a document layout hierarchy showing the document composed of pages, frames, blocks, lines, words and glyphs. A frame is a rectangular area like a column containing one or several blocks. A block is a homogeneous medium containing either text, picture, table, formula or graphics.

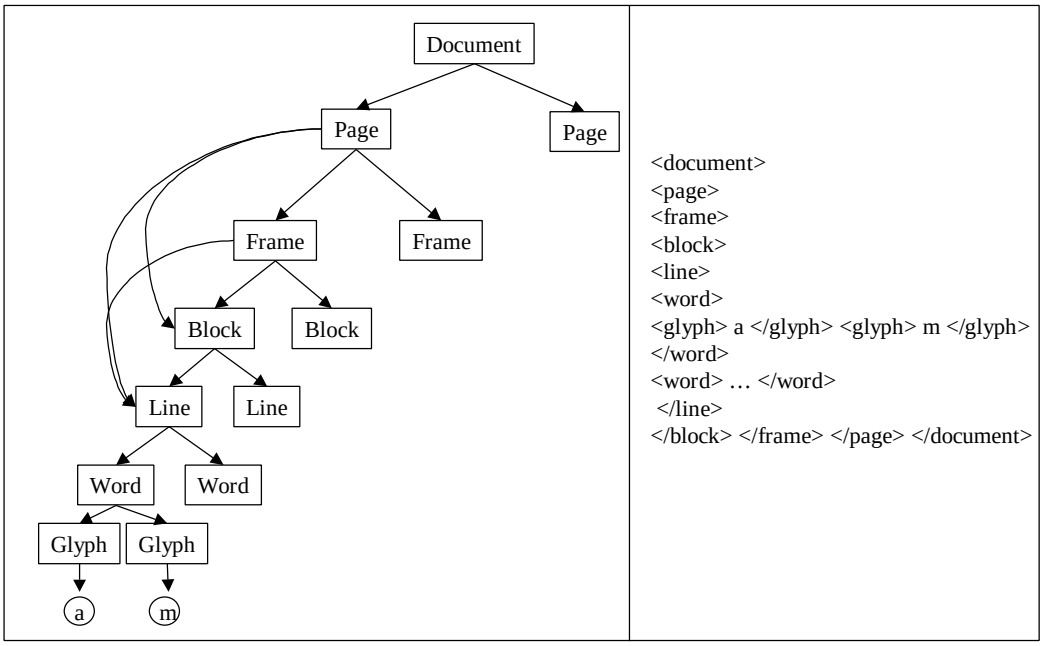

Figure 2: An example of a XmlLayout hierarchical structure

The table in annex describes the XmlLayout DTD. The main object of the DTD is “document”. A document contains at least one page, the element “document” attributes contain all the information on the document origin and information related to all the processing applied on it. There is a page per document even if one part only of the page is used. The page parts to analyze are defined by the Frame elements. If there is no frame, all the page is analyzed. The homogeneous segments such as text, table, graphics, etc. are represented by elements of block type. It is possible to not have blocks, leading to represent the total page by only one homogeneous block. Frame is a rectangular area defined before the OCR processing, in order to precise for the OCR the interesting parts to process. The block element is a rectangular area of a page or a frame which is identified as a homogeneous medium (text, graphics, picture, formula, table). If the block medium is text or table, then it is decomposed into lines. Line is composed of words which are composed of glyphs. Words and glyphs have an attribute related to the confidence score corresponding to the OCR recognition rate which can be numerical or symbolical. All the elements have an attribute giving their bounding box coordinates, and another one for font style description.

From XmlLayout to SVG

We use SVG for the document XmlLayout visualization. As far as we know, SVG is the only DTD which is able to take into account the physical objects with their absolute position in the 2D space. As the SVG DTD cannot represent more than one page, information about document and page are mixed. This DTD is chosen for a practical aspect due to the existence of visualization software (like Adobe SVG viewer) able to interpret this DTD and give a beautiful image. Documents in SVG can automatically be created from the original document into the XmlLayout DTD by using a sheet style XSLT. Table 2 gives the correspondence between Xmllayout elements and SVG elements. The elements of XmlLayout are put in a class attribute of an element of SVG. The bounding boxes are put in RECT elements, the groups of XmlLayout elements are put in a G element followed by a RECT element. Confidence score are put in the class attribute of the element TEXT or TSPAN. To put confidence score and element names in class attributes (XML allows more than one class by element) make us able to use CSS (cascading style sheet) to enhance visualization of documents in an SVG viewer, for example we can show low confidence glyphs in red.

XmlLayout element	SVG element
<DOCUMENT X,Y,W,H>>	<SVG viewbox>
<DOCUMENT imagefilename,language,ocrversion>	<DESC>"DOCUMENT :", "LANGUAGE :", "OCR :"
<PAGE x,y,w,h>	<RECT class="page" x,y,w,h>
<FRAME x,y,w,h>	<RECT class="frame" x,y,w,h>
<BLOCK media x,y,w,h>	<G class="block media"><RECT x,y,h,w>
<LINE>	<G class="line">
<WORD x,y,w,h>	<G class="word"><RECT x,y,w,h><TEXT class="word">
<GLYPH conf font size>	<TSPAN class="Cconf" style="font-size:size">

Table 1 : From XmlLayout to SVG

OCRAdapter Modules

The DTD XmlLayout was used for OCR platform construction by integrating several OCR engines. A C++ class named OCRAdapter is developed; its main function is to call an OCR engine and to return an XmlLayout document. This class is independent from the OCR used. This means that the user doesn't take care with the OCR used but only with what the OCR can bring. The class is also independent from any programming language. An interface was made to use OcrAdapter with scripting language Tcl and used in an application about FAX receiver identification. Another interface was built with VisualBasic to make an application to segment by line a document, this application is used in an industrial process to help intensive document human retroconversion. The figure 3 shows OcrAdapter integrated in a document recognition platform made of various OCRs.

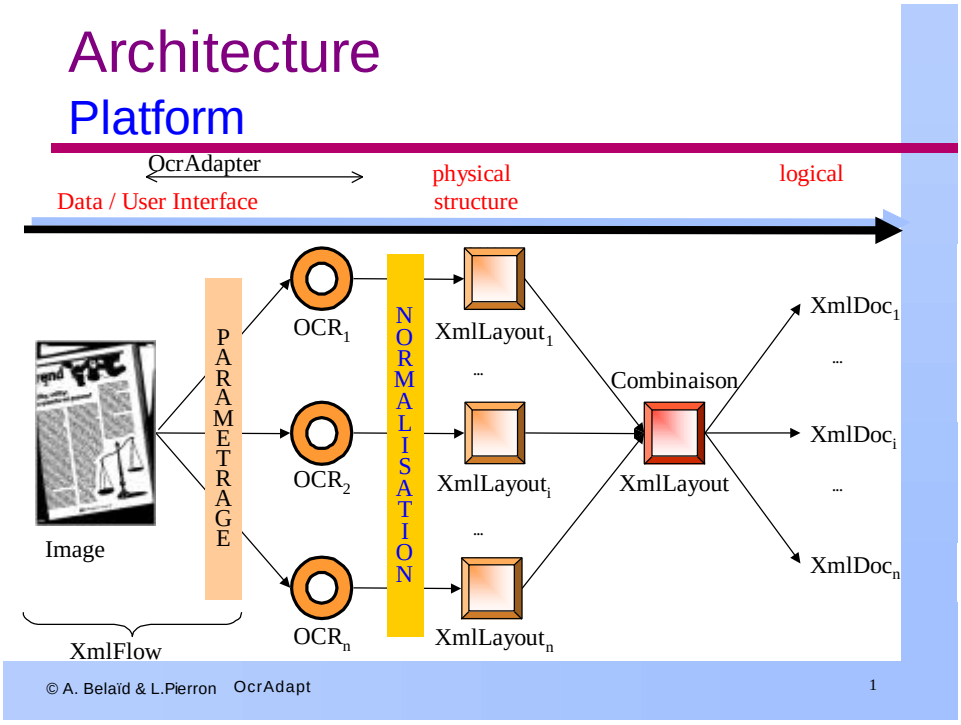

Figure 3 : Integrating OcrAdapter in platform

The idea of OcrAdapter is based on the *adapter* from the book Design Patterns. OcrAdapter is an abstract class which defines the skeleton of some methods. When you want to implement a real OCR, you derive a class from OcrAdpater, you implement the method from OcrAdapter by calling the real OCR and you put the result of the OCR in an XmlLayout document. Sometimes you have to convert the result from OCR specific format (like XDOC from ScanSoft) to XmlLayout or create XmlLayout document in memory with a DOM (Document

Object Model) API to copy result from OCR Engine internal structure to XmlLayout document. We proved the two methods by implementing : FineReader Engine (direct structure to XmlLayout), DevKit 2000 from Caere (direct structure to XmlLayout), GOCR (a free OCR engine, converter from result to XmlLayout) and textBridge API (converter from XDOC to XmlLayout).

Future Work

The W3 Consortium has defined the document edition process into two steps : first converting an XML logical document to a formatting object (FO) document with using of a style sheet (XSL), second from the FO document to the medium. As document image analysis is a reverse engineering process, this normally leads to use DTD to represent the inverse of the composers. So, FO DTD could be a good candidate to represent the document layout from the OCR output and the process to recognize the document logical structure could be made by reversing the style sheet.

References

Belaïd, A. and Brault, J.-J. and Chenevoy, Y.. Knowledge-based System for Structured Document Recognition. *In Proceedings MVA'90 IAPR Workshop on Machine Vision Applications.*(Tokyo (Japan)). 1990.
Belaïd, Abdel. Retrospective Document Conversion: Application to the Library Domain. *International Journal on Document Analysis and Recognition.* 1998. vol 1. n° 3. pp.125-146.
K. Taghva, A. Condit, J. Borsack and S. Erva : *Structural Markup of OCR Generated Text*, Technical Report 94-02, 1994, ISRI, University of Nevada, Las Vegas
DAFS : the Document Attribute Format Specification (http://www.dafs.org/)
XML : Extensible Markup Language (http://www.w3.org/XML/)
XSL : Extensible Stylesheet Language (http://www.w3.org/Style/XSL/)
FO : Formatting Objects (http://www.w3.org/TR/xsl/slice6.html#fo-section)
CSS : Cascading Style Sheets, a simple mechanism for adding style to Web documents (http://www.w3.org/Style/CSS/)
DOM : Document Object Model (http://www.w3.org/DOM/)
SVG : W3C Scalable Vector Graphics (http://www.w3.org/Graphics/SVG/)
Design Patterns : *Design Patterns: Elements of Reusable Object-Oriented Software*, by E. Gamma, R. Helm, R. Johnson, J. Vlissides, Addison-Wesley Publising Company, 1995. (http://www.ti.et-inf.uni-siegen.de/Entwurfsmuster/ArtikelImport/osefa/patterns/patterns.htm)

Annex XmlLayout DTD

Erreur! Nom de fichier incorrect.	<!ELEMENT frame (block|line)*> <!ATTLIST frame %commonattr; %styleattr;> <!ELEMENT block (line)*> <!ATTLIST block %commonattr; %styleattr; media (text|table|picture|graph|math) #REQUIRED> <!ELEMENT line (word)+> <!ATTLIST line %commonattr; %styleattr; alignement (right|left|centered|justified) #IMPLIED > <!ELEMENT word (#PCDATA|glyph)+> <!ATTLIST word %commonattr; %styleattr; %confidence;> <!ELEMENT glyph (#PCDATA)> <!ATTLIST glyph %commonattr; %styleattr; %confidence;>

Table 2 : DTD XmlLayout