Extreme Markup Languages 2007®

Montréal, Québec
August 7-10, 2007

From Word to XML to Mobile Devices

Paper Submission **Extreme2007-xxx**

David Lee
Epocrates, Inc.

Abstract

Epocrates provides clinical references on mobile devices. Key constraints for storing content on mobile devices are small memory footprint and limited CPU power. Key attributes of clinical content is 'rich data' requiring structural and presentation markup to be displayed. We have standardized on an efficient form of binary XML (XText) which is designed for efficient parsing and display natively on mobile devices while allowing authoring in any environment that can produce XML.

A problem arises because clinical content authors are familiar with authoring using Microsoft Word and are not comfortable authoring in native XML editors. Furthermore, content is not only authored by Epocrates employees, but also by third parties who we do not have the ability to require provide us with XML. The vast majority of content authors, both internal and external, are only trained and comfortable authoring in Microsoft Word. Even when provided with other methods of data entry, such as web enabled applications or native XML editors, the authors initially create content using Word and then copy and paste into the application. This 'impedance mismatch' causes much frustration, inefficiency and errors (both human and machine).

Rather then fighting against this installed base of skills and habits, we have created a workflow and tools to enable authors to create content in their environment of choice (MS/Word) and then translate these documents into our XML schema which is then used to create the binary XML content which is deployed to mobile devices.

This paper is a case study of our experiences to develop a workflow to handle smooth translation from MS/Word formats to XML and then to device formats. This includes lessons learned in both the human and technical realms and can serve as a design pattern for others attempting to achieve similar goals. Automated portions of the workflow include an XML pipeline based on XQuery. Human and usability consideration is given to what types of editing within Word people are most comfortable with and what kinds of errors and mistakes are common. The entire workflow is implemented using no commercial components except for Microsoft Office, and is based on Java, scripting and the Saxon implementation of XQuery.

From Word to XML to Mobile Devices

Table of Contents

From Word to XML to Mobile Devices

David Lee

§ Introduction

Who Is Epocrates ?

Epocrates provides clinical references on mobile devices to over 475,000 active subscribers. Our core products are a subscription based clinical publishing reference.

Common Terminology

There are a few terms which are difficult to avoid, so they are defined here.

PDA [Personal Digital Assistant]	In this paper refers to handheld devices, running a general purpose operating system, specifically the Palm/OS or Pocket PC OS (Win/CE).
Palm	A PDA device running the Palm/OS operating system
PPC	A PDA device running Microsoft's Pocket PC or Windows Mobile operating system.
PDB	From the Palm/OS terminology, a "PDB" is a "Palm Database". A very simple variable length record format with a single 16 bit key index.
Syncing	The process of synchronizing a server's database with a PDA's database. New records on the server are inserted into the PDA database, modified records updated and records which no longer exist on the server are deleted from the PDA.
Sync Server	The on-line servers in the Epocrates data center responsible for managing syncing to the PDA's.
KOL [Key Oppinion Leader]	From clinical publication terminology, refers to a person who is considered an expert in their field and who's comments and opinions can be trusted.

§ Core Application

The Core Application supporting the technologies and workflow desribed in this paper is the MRC [Mobile Resource Center] application which provides a constantly updated mobile reference to clinical publications. See figure 1

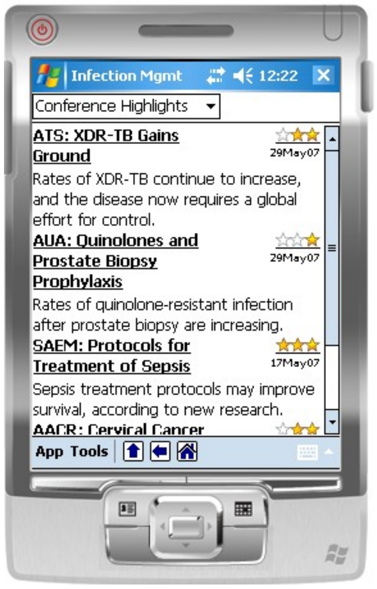
§ Background "The Problem Space"

Often, the more difficult problems are not technical problems, they are human problems. Somtimes forcing people to work differently so they fit well within your ideal technical model simply doesn't work.

The "problem" attempting to be solved is the 'impedance mismatch' between clinical authors and XML content. Our experience is that the authors of clinical content authors are familiar with authoring using Microsoft Word and are not comfortable authoring in native XML editors. Furthermore, content is not only authored by Epocrates employees, but also by third parties who we do not have the ability to require provide us with XML. The vast majority of content authors, both internal and external, are only trained and comfortable authoring in Microsoft Word. Even when provided with other methods of data entry, such as web enabled applications or native XML editors, they initially create content using Word and then copy and paste into the application. This 'impedance mismatch' causes much frustration, inefficiency and errors (both human and machine).

Rather then fighting against this installed base of skills and habits, we have created a workflow and tools to enable authors to create content in their environment of choice (MS/Word) and then translate these documents into our XML schema which is then used to create the binary XML content which is deployed to mobile devices. This workflow leverages the common skills and experiences of the authors by allowing them to continue to use Word as their authoring software, but constraining them in ways which makes the translation of Word to our XML schema fairly simple.

Figure 1: Home screen of the Mobile Resource Center Application



ATS: XDR-TB Gains Ground 29May07
Rates of XDR-TB continue to increase, and the disease now requires a global effort for control.
AUA: Quinolones and Prostate Biopsy Prophylaxis 29May07
Rates of quinolone-resistant infection after prostate biopsy are increasing.
SAEM: Protocols for Treatment of Sepsis 17May07
Sepsis treatment protocols may improve survival, according to new research.
AACR: Cervical Cancer

Characteristics of the application

The characteristics of the mobile application, with respect to XML usage, help drive the design requirements for XML processing on the device and in content creation. The application runs on mobile devices which have very limited memory and typically slow CPU. Parsing XML on the device is not generally fast enough to provide a good user experience, instead an efficient binary XML encoding of the document is deployed to the device which can be parsed and displayed quickly and with minimum memory requirements. This means that all XML parsing, and conversion to device formats needs to be implemented on the server or desktop, not on the device, thus allowing the use of tools which are not portable to devices.

Workflow Outline

An intermediate XML format is used that represents the full structured data. This intermediate format can be generated from any number of sources, not only Word documents and then itself serves as the source of possibly multiple output formats. The relevant workflow involves editing content in Word, then translating first to the intermediate XML format (MRC [Mobile Resource Center] schema) then finally to a display format (HTML, PDF, XText).

Human Workflow

The human workflow is involves many people in the publishing process, the conversion to word to XML is only a small part if the process. See Figure 2.

XML Conversion workflow

The automated part of the workflow converts Word or other document content into the intermediate format (MRC XML) and then into an output display format. For mobile devices this is an XML schema called XText then binary encoded to an efficient format for the devices. See Figure 3.

Deployment workflow

The deployment process is initiated after the final QA of the content. Device formated content is pushed to the database server and made available to the "Sync Servers" in the form of both complete data (full record images) and incremental data (changed data since the last update). When devices connect to the data center during synchronization, the appropriate content is deployed to the device to bring it up to date. See Figure 4.

Figure 2: Human Workflow

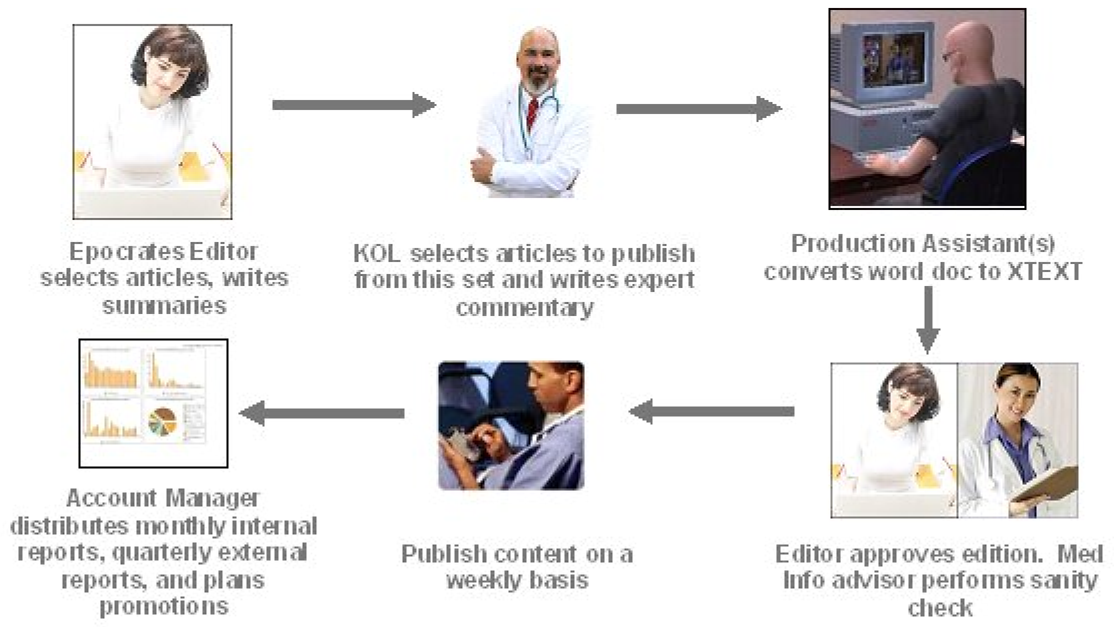
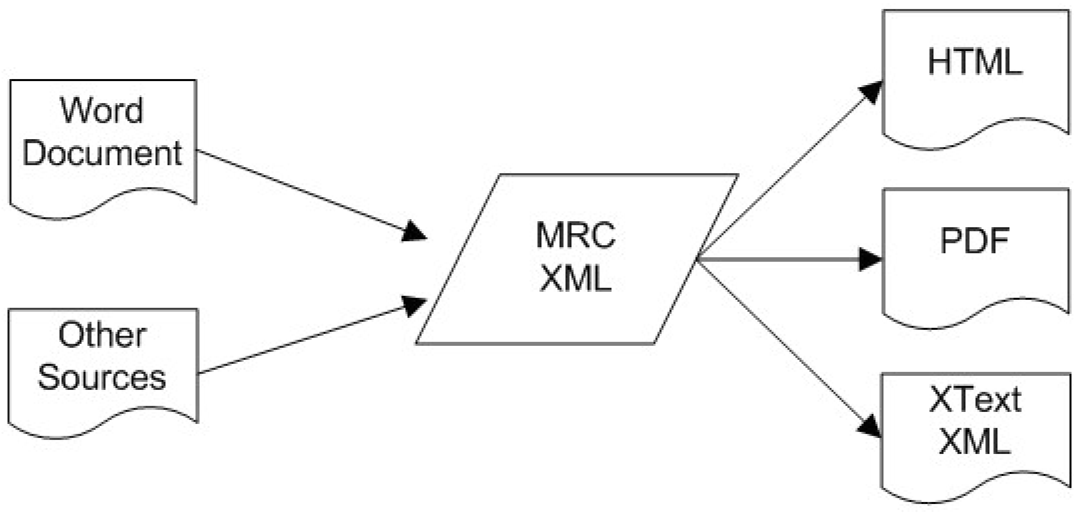

Epocrates Editor selects articles, writes summaries
KOL selects articles to publish from this set and writes expert commentary
Production Assistant(s) converts word doc to XTEXT
Editor approves edition. Med Info advisor performs sanity check
Publish content on a weekly basis
Account Manager distributes monthly internal reports, quarterly external reports, and plans promotions

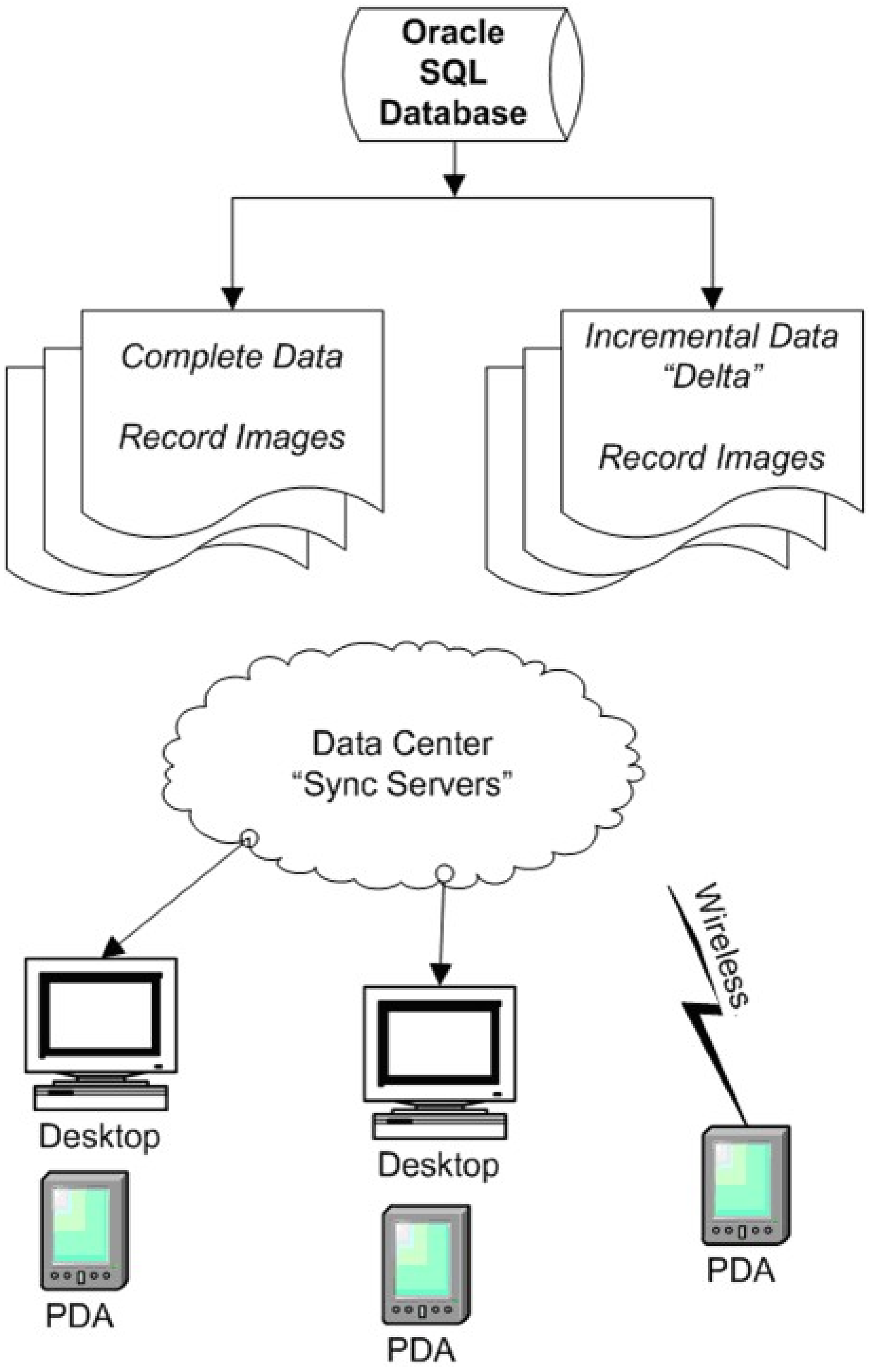
Figure 3: XML Conversion workflow



§ MS Word Authoring

At Epocrates we have had years of experience with how typical content authors, at least in the clinical publishing field, work within the MS/Word environment. Somewhat surprising, to us at least, is that many of the sophisticated techniques used by the popular Word to XML conversion tools actually work extremely poorly in the 'real world' of clinical authoring. We have learned about what features of Word can be exploited to aid in XML translation, and what features are too easily abused.

Figure 4: Deployment workflow



Word Styles

Early attempts at enforcing structure within word documents was heavy use of Styles. The idea is to use word styles to convert directly to XML elements or to tag locations in a document corresponding to sections in an XML document. In theory this seems like a perfect match between Word and XML, and is the technique used by several major commercial Word to XML conversion applications

However in practice we have found this to be an unsuccessful strategy. It is very difficult to enforce consistent use of styles in the Word environment even if we supply the template (Word DOT files). Word does not have much ability to constrain use of styles and which styles are allowed where in the document. Authors frequently will apply the wrong styles, copy styles from other documents, delete or re-arrange the positions of tagged markers in template documents. Also common practice is authors choosing to not use styles at all but rather use the font characteristics to achieve the right "look" without actually marking the document with styles.

Tagged Sections

A common goal in extracting structure from word documents is to identify sections of text which belong within XML elements. We created template documents and marked sections with either styles or bookmarks. For example, we create an "Abstract" section, and mark it by using the word "Abstract" using the "Abstract" style or a bookmark "Abstract", expecting authors to enter the abstract text directly after this.

Unfortunately our experience is that authors would edit the tagged sections, sometimes deleting them entirely, reorder them, or copy and paste sections from other documents making parsing of the document impossible.

Form Fields

We experimented briefly with word Form Fields. The idea is that these are the natural candidates for entering tabular data. Surprisingly people had a great deal of problems with them. The form fields only allow "plain" text (no markup), this caused some people to actually delete the form fields and paste in structured text. Furthermore, parsing the data of the form fields from the Word Object Model was tedious. In general we found form fields to be of very limited usefulness.

Formatting and symbol characters

A very common problem is authors using word formatting, and 'smart' features such as "Smart Quotes", auto conversions to symbol characters such as copyright, em-dash, en-dash) and manually inserting Unicode characters such as units (micro), Greek characters, and mathematical symbols. The set of symbols which are portably supported on mobile devices is very limited and these 'special' characters are a constant cause of problems. It is difficult at the time of authoring to prevent people from inserting unsupportable characters.

Since it is impossible to stop insertion of any text into a word document, special care must be used to strip out or translate unacceptable characters, especially the "smart" characters that Word is so insistent on using

Word Macros

The word macro facility (Word VB Script) is a very powerful feature of Word. When used carefully it can assist in the authoring process, scan documents for errors, correct common mistakes, and even save out an XML representation of word documents. However when overused it can cause problems for the author by getting in the way too much, hitting unanticipated runtime errors or not being activated properly. The macro language itself, while powerful, is cumbersome to program complex operations and difficult to maintain and deploy consistently.

It is difficult, and sometimes impossible, to enforce the use of macros. Security settings on individual workstations can completely turn off macros. Use of multiple versions of Word can cause compatibility problems. Unanticipated copy and paste of content across documents can strip documents of attached macros. Keeping a distributed authoring environment up to date with code changes is also a challenge.

Our experience with real world suggests that deployment of macros to an authoring staff is reasonable for some purposes as long as some basic design rules are followed.

- Macros should never be required for authoring. They can provide useful error checking help during the authoring phase but should not be an absolute requirement. If a user somehow disrupts the functioning of a macro in a document they should be able to complete their tasks, even if it means that some errors are undetected at authoring time.

- It is acceptable for the production staff to be required to use macros because their work environment is more controlled and can expect timely assistance from IT if problems occur.
- Macros should be kept as simple as possible due to complexities of writing, debugging and maintaining macros.
- Macros should not interact with every keystroke, but rather should be invoked at the end of an editing session to help catch and fix common errors.

§ Getting XML out of Word

We evaluated many methods of extracting XML from Word documents including several commercial products. We found no single 'off the shelf' method suitable to our needs for either cost, functionality, or usability reasons. The free products were not featureful enough for our needs, and the commercial products have very restrictive licensing agreements not appropriate to our distributed work-flow (very high per seat license fees) and still required a large development effort to customize the products which diminish their "off the shelf" advantages. Furthermore, we found the core techniques used by commercial products often run counter to our own usability experience.

We decided to implement in-house our Word to XML conversion.

Word to XML Conversion strategies

There are multiple strategies for converting Word to a usable XML format.

Convert through RTF

Many commercial and free software convert word documents to XML by first storing as RTF. Then a post processing phase parses the RTF and generates XML in some predefined schema.

Convert through HTML

This strategy involves "Save as HTML" then post processing the HTML to XML. Visual fidelity is good but structural elements are lost (such as bookmarks, styles tags).

Native Word 2003 XML editing

Word 2003 can save natively as Word XML. This is a very powerful feature but it requires use of Word 2003 or later and stores a very complex XML format which is tedious to parse.

Convert through Macros

Word Macros (VB Script) can access both the Word Document Model, and the MS XML COM object. This allows a word macro to store Word documents as XML in any schema. However, doing a good job of this is very difficult due to the complexity of the Word Document Model, and the use of VB Script. The advantage over "Save As XML" is that it does not require Word 2003 and can interactively present the user with error messages as problems are found.

Selected Input Strategies

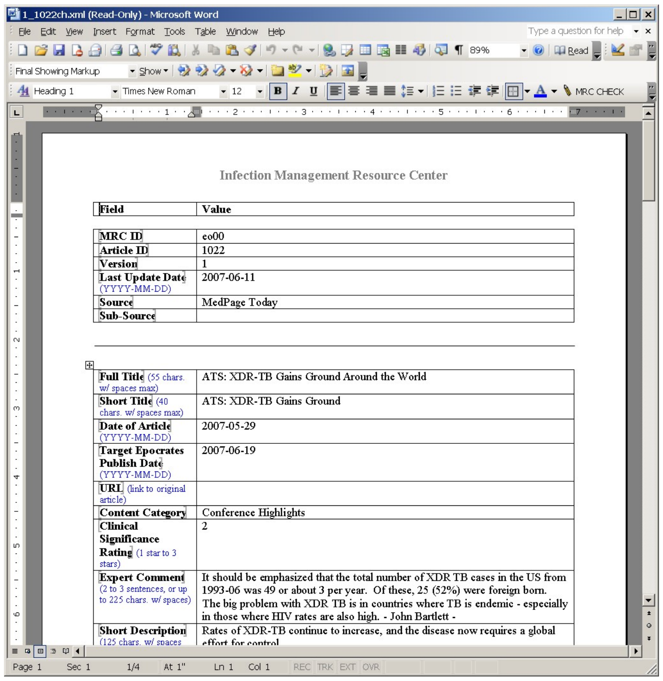
We have two separate products which currently need conversion from Word to XML, developed by different teams. One project, which has been in production for several years, used the "Convert through Macros" technique to extract and produce the final XML format. This project is where we uncovered many of the early mistaken assumptions about common editing usage. It also uncovered problems attempting to convert a Word doc through macros to a final XML format. A mixture of conversion and business logic all in one Word Macro created very complex code that was difficult to maintain and enhance, and impossible to re-use for other projects.

A new project gave us the opportunity to start over and reexamine our design. For both projects we standardized on a Word usage "design pattern" for minimizing data entry problems.

Word Tables

We have found to-date that the most error resistant and easiest to use design pattern for entering data into Word which is to be extracted as XML is to make use of word Tables. For both projects we construct one or more 2 column tables in the 'template' document for distribution to content editors.

Figure 5: Example word document



1_1022ch.xml (Read-Only) - Microsoft Word
File Edit View Insert Format Tools Table Window Help
Type a question for help
Final Showing Markup
Show
Heading 1
Times New Roman
12
MRC CHECK
Page 1 Sec 1 1/4 At 1" Ln 1 Col 1 REC TRK EXT OVR

Infection Management Resource Center

Field	Value
MRC ID	eo00
Article ID	1022
Version	1
Last Update Date (YYYY-MM-DD)	2007-06-11
Source	MedPage Today
Sub-Source	

Full Title (55 chars. w/ spaces max)	ATS: XDR-TB Gains Ground Around the World
Short Title (40 chars. w/ spaces max)	ATS: XDR-TB Gains Ground
Date of Article (YYYY-MM-DD)	2007-05-29
Target Epocrates Publish Date (YYYY-MM-DD)	2007-06-19
URL (link to original article)	
Content Category	Conference Highlights
Clinical Significance Rating (1 star to 3 stars)	2
Expert Comment (2 to 3 sentences, or up to 225 chars. w/ spaces)	It should be emphasized that the total number of XDR TB cases in the US from 1993-06 was 49 or about 3 per year. Of these, 25 (52%) were foreign born. The big problem with XDR TB is in countries where TB is endemic - especially in those where HIV rates are also high. - John Bartlett -
Short Description (125 chars. w/ spaces	Rates of XDR-TB continue to increase, and the disease now requires a global effort for control

In column 1 we enter field header names or descriptions like "Title" or "Article Date", and create a word bookmark with an internal field name. In addition we add descriptive text such as the date format 'YYYY-MM-DD' or other suggestive text such as "Enter the title up to 40 characters".

In column 2 we either leave the cells blank, or include sample data such as "Put the article title here". We have found this very simple structure is highly resilient to a variety of common human errors and editing practices. People very rarely corrupt tables in Word like they do in free-form or style tagged text. Even copy and pasting from other documents does not corrupt the controlling structure. Changing style tags doesn't effect things. Even moving or inserting rows doesn't break the structure as long as the entire row is moved. See Figure 5 for a sample word document in this format.

Furthermore this structure is relatively easy to parse and extract meaningful XML consistently.

While in this project, the schema is fairly flat, we have experimented with more complex schemas which have a deep structure and have found that nesting tables within tables to represent nested XML elements works just as well in practice.

The biggest difficulty using this approach is handling of repeating elements. Our experiments having authors insert blank rows to tables have shown we cannot count on the authors ability to do this correctly. Fortunately in our case, the use of repeating elements is limited to a small number of occurrences so we pre-create blank cells for the maximum number of occurrences. We have not yet found a good solution to handling of an arbitrary number of repeating elements.

Word Macros

We have found that careful use of Word Macros can provide valuable and early error correction which is difficult to provide further 'downstream' in the pipeline. Once data is extracted from a Word file its almost impossible to backtrack that data and provide meaningful errors to the author about where the data came from. By using a Word macro many errors can be detected at point of entry and meaningful feedback given such as positioning the cursor at the error. We also made use of some automatic corrections including translating smart quotes to ascii quotes, and converting unsupportable common Unicode characters.

There are still problems with using macros that are difficult to eliminate completely such as security settings, users copying data into other files without copying the macros, installing into the "Normal.DOT" template on a users system etc. These issues are difficult or impossible solve perfectly so its important to consider use of word macros as an "aid" but not a requirement for the editing process, especially when the authors are remote, in unmanaged environments, or especially when outside of the organization.

Selected XML Extraction strategies

The original project used Word Basic to extract XML from word documents, for the new project we went with Word 2003 'Save as XML' and store only the XML version of the word document. In both cases we defined an intermediate XML schema to represent the basic structural elements we needed to extract from the Word documents. This schema (we call it 'Epocrates Word Schema') is composed of the following very simple element set. This is intentionally simplistic and is designed to be the bare minimum needed for our applications.

Elements in the Epocrates Word Schema

Tables	Captures the structure of table, row, and cells.
Paragraphs	Captures paragraph breaks including the word Style associated with the paragraph
Line breaks	Encodes hard line breaks.
Formatting	Captures basic character formatting including bold, italics and underline.
Bookmarks	Capture bookmark sections. Words bookmark-start/bookmark-end parse are converted into a single bookmark tag which identify a point in the document.
Markup tags	We needed the ability to 'pass through' XML-like markup. To accommodate this we recognize freeform text that conforms to simple XML syntax and encode it as TAG_START and TAG_END tags.
Plain text	Everything else is plain text. Some word special symbols are translated into more usable Unicode representation.

Word Basic

With one project we use Word Basic to parse the Word document and write it out to the intermediate XML format. The main advantage to this strategy is that it works with versions of Word prior to Word 2003. Furthermore the XML extraction can be part of the same Word Macro which performs the error checking.

The disadvantages are that Word Basic is a tedious language to develop complex software, and that it is actually quite complicated to parse through the Word Object Model to extract the necessary data.

Word XML

In the new project we made use of the Word 2003 feature which allows you to store a word document with full fidelity as an XML file. You can do this with "Save As" (either manually or in a macro) or by changing the default file type to xml. The resulting file can be opened within Word and re-saved as a .doc file with no loss of information. We use an XQuery script to convert the Word XML file to the same "Epocrates Word" intermediate XML format.

By converting first to an intermediate format, it not only decoupled the input from direct dependency on Word XML (and hence an architectural dependency on Word 2003), but also allowed vastly simpler code further down the pipeline.

§ XTool Application

An "XTool Application" is an application targeted for mobile devices which is entirely specified as a set of XML files. These XML files compose the configuration (meta data), indexing, and the content. The Content is the bulk of the data and is represented in "XText" schema XML Files.

XText is a schema designed for describing documents optimized for mobile devices. Its a schema similar to XHTML in concept but much simpler. It removes most of the device assumptions inherent in XHTML (such as font sizes, RGB colors, pixel metrics) as well as simplifies many of the core elements for efficient rendering, and in particular rendering appropriate to the characteristics of mobile devices. There is also inclusion of some concepts particular to the Epocrates mobile architecture, such as inter-application linking and enforcement of consistent look and feel.

On the device, due to severe resource constraints (CPU, RAM and storage), a binary encoded and compressed XML representation is used. This improves resource usage significantly and allows for quick real-time parsing and rendering even on the most constrained devices.

On the server, the serialized text representation of the XML is encoded into the binary format and packaged into the device specific database format ("PDB" files). The application is deployed to the devices as part of a synchronization.

§ XML Pipeline

The entire process of converting a set of Word documents, to the final output "PDB" format was implemented as an XML Pipeline, primarily in XQuery, but with some java code as well. The end point of XML in this pipeline is an XML file describing the binary layout of the PDB file, which is then converted to binary PDB format by using java code to produce a text file used by an off-the-shelf tool "pdbc" which generate the palm database files. The content of these PDB files is primarily binary encoded XML documents (in XText format) designed and optimized for presentation on mobile devices, along with some indexing information.

The pipeline makes heavy use of intermediate formats. This design is intended to allow multiple insertion points along the pipeline to accommodate content originating from various sources, not only Word documents. In addition, the intermediate formats allow document generation at various stages where appropriate.

The whole pipeline looks something like this Figure 6.

The implementation of the pipeline infrastructure was written in java, making heavy use of Saxon XQuery implementation for all XML transformations. Various pipeline implementations were considered but discarded because none were a good fit for this process due to the necessity of passing in some computed runtime values. The pipeline process could have been easily implemented as a sequence of individual processes in a scripting environment.

XQuery was used instead of XSLT as a proof of concept that XQuery is well suited for general purpose complex XML transformation in addition to query. Furthermore, I found that the XQuery language and syntax to be much simpler write, read, and debug then XSLT.

The following outlines the steps in the pipeline.

Doc Conversion

Input files in DOC format are converted to WordXML format using a word macro. Files already in WordXML file are imported without conversion.

MRC Compiler

The MRC [Mobile Resource Center] Compiler component is responsible for converting content in "MRC" format into "XTool" format. "MRC" format consists the files required to define a "Mobile Resource Center", which is a collection of articles and supplemental information surrounding a single clinical topic.

The output of the MRC compiler is the collection of files in "XTool Format". "XTool" format is the a set of XML files which constitute a single deployable application to the device. The XML files include the documents, indexing, and meta data which describes a single deployable application.

Epoc Word Creation

An XQuery script parses each WordXML format file and produces the "Epoc Word" XML format.

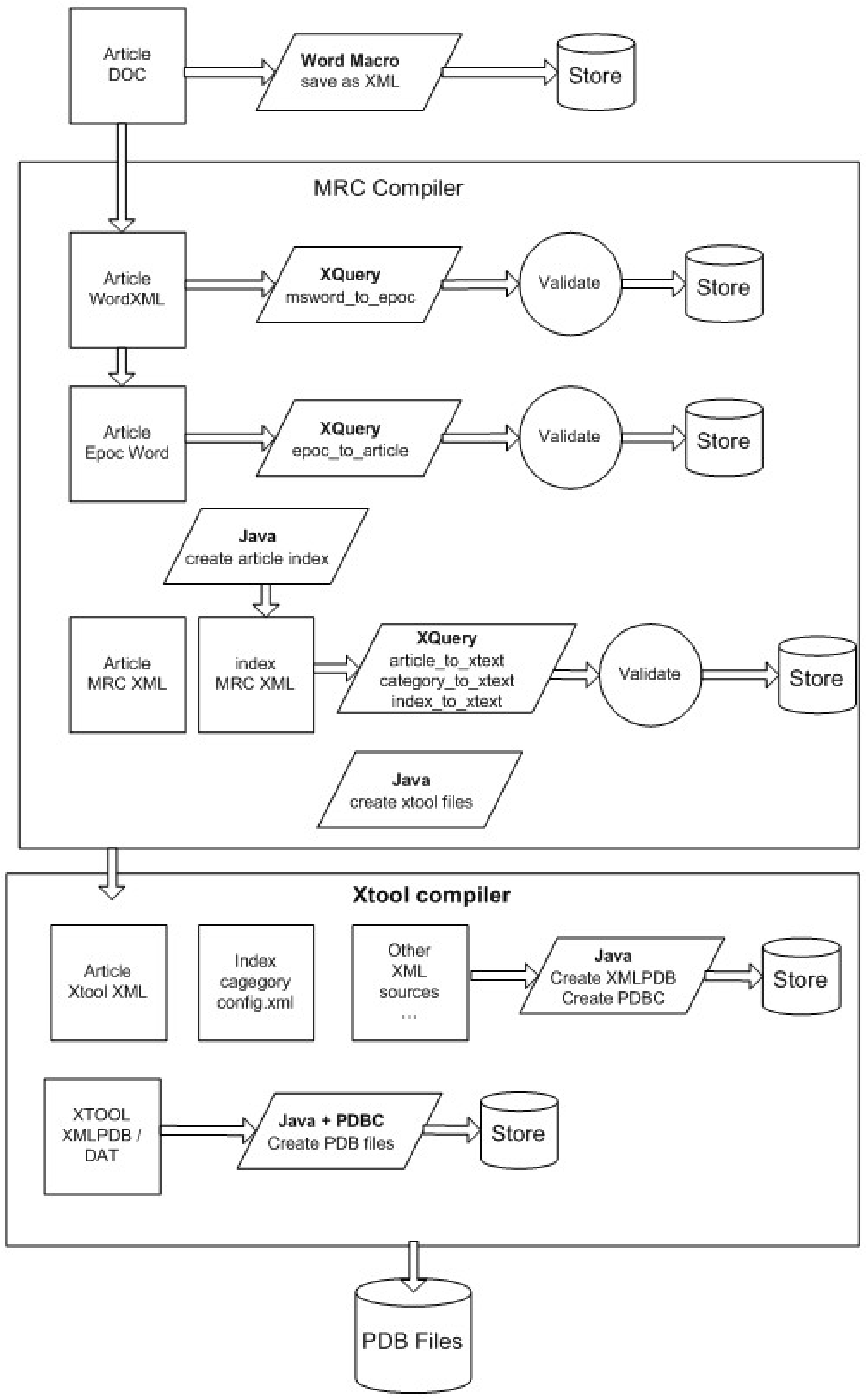
The resulting files are validated against the epoc word schema.

Documents already in epoc word schema may be imported at this step.

Epocrates Word Schema

The following is the Epocrates Word Schema

Figure 6: XML Pipeline



```
<?xml version = "1.0" encoding = "UTF-8"?>
<xsd:schema xmlns:xsd = "http://www.w3.org/2001/XMLSchema">
    <xsd:group name = "text">
        <xsd:choice>
            <xsd:element ref = "B"/>
            <xsd:element ref = "LINK"/>
            <xsd:element ref = "BR"/>
            <xsd:element ref = "BOOKMARK"/>
            <xsd:element ref = "TAG_START"/>
            <xsd:element ref = "TAG_END"/>
        </xsd:choice>
    </xsd:group>
    <xsd:element name = "DOC">
        <xsd:complexType mixed = "true">
            <xsd:choice minOccurs = "0" maxOccurs = "unbounded">
                <xsd:element ref = "P"/>
                <xsd:element ref = "TABLE"/>
            </xsd:choice>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name = "P">
        <xsd:complexType mixed = "true">
            <xsd:sequence minOccurs = "0" maxOccurs = "unbounded">
                <xsd:group ref = "text"/>
            </xsd:sequence>
            <xsd:attribute name = "style" type = "xsd:string"/>
            <xsd:attribute name = "list" type = "xsd:string"/>
            <xsd:attribute name = "number" type = "xsd:string"/>
            <xsd:attribute name = "level" type = "xsd:integer"/>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name = "TABLE">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref = "ROW" minOccurs = "0" maxOccurs = "unbounded"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name = "ROW">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref = "COL" minOccurs = "0" maxOccurs = "unbounded"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name = "COL">
        <xsd:complexType mixed = "true">
            <xsd:choice minOccurs = "0" maxOccurs = "unbounded">
                <xsd:element ref = "P"/>
                <xsd:element ref = "BOOKMARK"/>
            </xsd:choice>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name = "B" type = "xsd:string"/>
    <xsd:element name = "LINK">
        <xsd:complexType>
            <xsd:simpleContent>
                <xsd:extension base = "xsd:string">
                    <xsd:attribute name = "href" use = "required" type = "xsd:string"/>
                </xsd:extension>
            </xsd:simpleContent>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name = "BR">
        <xsd:complexType/>
    </xsd:element>
    <xsd:element name = "BOOKMARK">
        <xsd:complexType>
            <xsd:attribute name = "name" use = "required" type = "xsd:string"/>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name = "TAG_START">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref = "ATTRIBUTE" minOccurs = "0" maxOccurs = "unbounded"/>
            </xsd:sequence>
            <xsd:attribute name = "name" use = "required" type = "xsd:string"/>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name = "TAG_END">
        <xsd:complexType>
            <xsd:attribute name = "name" use = "required" type = "xsd:string"/>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name = "ATTRIBUTE">
        <xsd:complexType>
            <xsd:attribute name = "name" use = "required" type = "xsd:string"/>
            <xsd:attribute name = "value" type = "xsd:string"/>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
```

This is an example of the word document in Figure 5 after it is converted into the Epocrates Word Schema.

```
<?xml version="1.0" encoding="US-ASCII"?>
    <DOC>
        <TABLE>
            <ROW>
                <COL><P style="Heading1">Field</P></COL>
                <COL><P style="Heading1">Value</P></COL>
            </ROW>
        </TABLE><P/>
        <TABLE>
            <ROW>
                <COL><P><B>MRC ID</B><BOOKMARK name="mrc_id"/></P></COL>
                <COL><P>eo00</P></COL>
            </ROW>
            <ROW>
                <COL><P><B>Article ID</B><BOOKMARK name="article_id"/></P></COL>
                <COL><P>1022</P></COL>
            </ROW>
            <ROW>
                <COL><P style="Heading1">Version<BOOKMARK name="version"/></P></COL>
                <COL><P style="Heading1">1</P></COL>
            </ROW>
            <ROW>
                <COL><P style="Heading1">Last Update Date<BOOKMARK name="last_update_date"/> (YYYY-MM-DD)</P></C
                <COL><P style="Heading1">2007-06-11</P></COL>
            </ROW>
            <ROW>
                <COL><P style="Heading1">Source<BOOKMARK name="source"/></P></COL>
                <COL><P style="Heading1">MedPage Today</P></COL>
            </ROW>
            <ROW>
                <COL><P style="Heading1">Sub-Source<BOOKMARK name="sub_source"/></P></COL>
                <COL><P style="Heading1"/></COL></ROW>
            </TABLE>

            ....
```

Article creation

An XQuery script converts each article file from Epoc Word format into the MRC Article format. The resulting files are validated against the MRC Article schema.

Article Schema

The Article schema represents a single MRC article in a direct structured format. This schema is fairly simple and allows for allows for a minimum of mixed text, in addition to the structural markup

The following is the Article schema

```
<?xml version = "1.0" encoding = "UTF-8"?>
<xsd:schema xmlns:xsd = "http://www.w3.org/2001/XMLSchema">
    <xsd:group name = "para">
        <xsd:choice>
            <xsd:element ref = "P"/>
            <xsd:element ref = "M"/>
            <xsd:element ref = "L"/>
        </xsd:choice>
    </xsd:group>
    <xsd:group name = "text">
        <xsd:choice>
            <xsd:element ref = "B"/>
            <xsd:element ref = "LINK"/>
            <xsd:element ref = "BR"/>
        </xsd:choice>
    </xsd:group>
    <xsd:element name = "ARTICLE">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name = "FULL_TITLE" type = "xsd:string"/>
                <xsd:element name = "SHORT_TITLE" type = "xsd:string"/>
                <xsd:element ref = "EXPERT_COMMENT"/>
                <xsd:element ref = "SHORT_DESC"/>
                <xsd:element ref = "PDA_TEXT"/>
            </xsd:sequence>
            <xsd:attribute name = "article_id" use = "required" type = "xsd:integer"/>
            <xsd:attribute name = "clinical_significance_rating" use = "required">
                <xsd:simpleType>
```

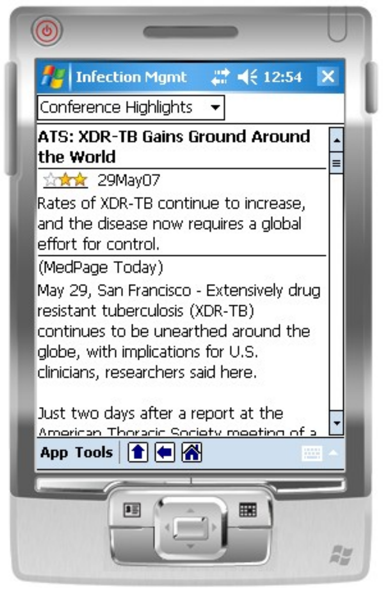
```
                    <xsd:restriction base = "xsd:integer">
                        <xsd:maxInclusive value = "3"/>
                        <xsd:minInclusive value = "0"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:attribute>
            <xsd:attribute name = "content_category" use = "required" type = "xsd:string"/>
            <xsd:attribute name = "date_of_article" use = "required" type = "xsd:date"/>
            <xsd:attribute name = "last_update_date" use = "required" type = "xsd:date"/>
            <xsd:attribute name = "target_epoc_publish_date" use = "required" type = "xsd:date"/>
            <xsd:attribute name = "mrc_id" use = "required" type = "xsd:string"/>
            <xsd:attribute name = "version" use = "required" type = "xsd:integer"/>
            <xsd:attribute name = "source" type = "xsd:string"/>
            <xsd:attribute name = "sub_source" type = "xsd:string"/>
            <xsd:attribute name = "url" type = "xsd:string"/>


        </xsd:complexType>
    </xsd:element>
    <xsd:element name = "EXPERT_COMMENT">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref = "P" minOccurs = "0" maxOccurs = "unbounded"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name = "SHORT_DESC">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref = "P" minOccurs = "0" maxOccurs = "unbounded"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name = "PDA_TEXT">
        <xsd:complexType>
            <xsd:choice minOccurs = "0" maxOccurs = "unbounded">
                <xsd:group ref = "para"/>
            </xsd:choice>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name = "P">
        <xsd:complexType mixed = "true">
            <xsd:sequence minOccurs = "0" maxOccurs = "unbounded">
                <xsd:group ref = "text"/>
            </xsd:sequence>
            <xsd:attribute name = "list" type = "xsd:string"/>
            <xsd:attribute name = "number" type = "xsd:string"/>
            <xsd:attribute name = "level" type = "xsd:integer"/>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name = "B" type = "xsd:string"/>
    <xsd:element name = "LINK">
        <xsd:complexType>
            <xsd:simpleContent>
                <xsd:extension base = "xsd:string">
                    <xsd:attribute name = "href" use = "required" type = "xsd:string"/>
                </xsd:extension>
            </xsd:simpleContent>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name = "BR">
        <xsd:complexType/>
    </xsd:element>
    <xsd:element name = "M"/>
    <xsd:element name = "L"/>
</xsd:schema>
```

The following is a sample of the same article from Figure 4 after conversion to the Article schema

```
<?xml version="1.0" encoding="US-ASCII"?>
<ARTICLE
    mrc_id="eo00"
    target_epoc_publish_date="2007-06-19"
    clinical_significance_rating="2"
    article_id="1022"
    content_category="Conference Highlights"
    date_of_article="2007-05-29"
    version="1"
    last_update_date="2007-06-11">
    <FULL_TITLE>ATS: XDR-TB Gains Ground Around the World</FULL_TITLE>
    <SHORT_TITLE>ATS: XDR-TB Gains Ground</SHORT_TITLE>
    <EXPERT_COMMENT><P>It should be emphasized that the total number of XDR TB cases in the US
        from 1993-06 was 49 or about 3 per year.  Of these, 25 (52%) were foreign born.
        The big problem with XDR TB is in countries where TB is endemic -
        especially in those where HIV rates are also high. - John Bartlett - </P>
```

Figure 7: Sample MRC Article page on the mobile device



ATS: XDR-TB Gains Ground Around the World

29May07

Rates of XDR-TB continue to increase, and the disease now requires a global effort for control.

(MedPage Today)

May 29, San Francisco - Extensively drug resistant tuberculosis (XDR-TB) continues to be unearthed around the globe, with implications for U.S. clinicians, researchers said here.

Just two days after a report at the

American Thoracic Society meeting of a

```
    </EXPERT_COMMENT>
    <SHORT_DESC><P>Rates of XDR-TB continue to increase, and the disease now requires a global effort for contro
    </SHORT_DESC>
    <PDA_TEXT><P>(MedPage Today)</P><P>May 29, San Francisco - Extensively drug resistant tuberculosis

    ...
```

XTool Compiler

The XTool compiler takes a set of XML files describing an "XTool Application" (an XML config file, and XML XText content files) and creates the binary device "PDB" files. It does this through a pipeline that first encodes the content files from the XText schema into a binary encoded XML format, which is then output as XMLPDB format. The XMLPDB format describes the exact binary record layout for the PDB files. This format is then converted to "pdbc" format which is the text format required by the pdbc compiler, and then compiled into binary PDB files.

XText content

Each page of the XTool application data is specified in XML using the XText schema. The following is the XText XML content of the same article in Figure 4 after conversion from the Article format to the XText format used by XTool. Figure 7 shows what this page looks like when deployed to the mobile device and rendered by the MRC application.

```
<?xml version="1.0" encoding="US-ASCII"?>
<XTEXT lr_hpad="0" tb_vpad="0">
  <HEADER lr_hpad="0" tb_vpad="0">
    <COMBOBOX name="topnav">
        <OPTION value="Resource Center Home" page="index"/>
        <OPTION value="Conference Highlights" selected="1" page="conference"/>
        <OPTION value="Clinical News" selected="0" page="news"/>
        <OPTION value="Scientific Abstracts" selected="0" page="pubmed"/>
        <OPTION value="MobileCME Links" selected="0" page="cme"/>
        <OPTION value="Resources" selected="0" page="resources"/>
    </COMBOBOX>
    <HR/>
    </HEADER>
    <PAGE lr_hpad="1" tb_vpad="0">
        <P align="left"><FONT color="black" bold="true">ATS: XDR-TB Gains Ground Around the World</FONT></P><HR
        <P align="left"><IMGBUTTON popup="star" src="two_star" name="star"/>&#160;29May07</P>
        <P>Rates of XDR-TB continue to increase, and the disease now requires a global effort for control.</P>
        <HR/><P>(MedPage Today)</P><P>May 29, San Francisco - Extensively drug resistant tuberculosis (XDR-TB) c

    ...
```

PDB Compiler

The XMLPDB files are transformed into "pdbc" format, a text format used by the pdbc compiler. Then pdbc is executed to create the binary PDB files from the pdbc text files. An application configuration file (app.xml) is also created to pass on the meta data required for deploying an XTool application.

§ Postmortum - Lessons learned.

The design outlined in this paper were evolved as much by our failures as our successes. Our failures were both human and technical.

On the human side were incorrect assumptions about how authors really use word processing tools. The best technical solutions failed miserably when the people don't use them in the expected manner. Adopting the technical solutions around how people actually work, rather then trying to get them to change their work habits has been the most successful strategy.

On the technical side, using complex word macros (VBScript) turned out to be a big mistake. Especially when the macros are a required part of the authoring environment. Attempting a monolithic single program which converted Word to the final XML format was also very challenging, difficult to debug and left no room for future expansion. Implementing the process as a pipeline, instead of a single transformation, resulted in a much better solution. The use of XQuery exclusively for the XML transformations worked very well. I was tempted in many occasions to use the Saxon feature of calling into native Java code in the XQuery scripts, but found that it was unnecessary. XQuery alone provided 100% of the needed functionality for every phase of the pipeline. The java and script code used to integrate the pipeline steps was only needed to gather input from the environment, and to call the processes necessary for the final conversion from XML to the device format.

§ Conclusion

Epocrates has successfully adopted an XML oriented pipeline architecture bridging two which are frequently difficult or inefficient to integrate with XML - authoring in Microsoft Word and rendering rich content on mobile devices. Effectively addressing two "ends" of the pipeline are critical for a successful XML implementation, as fundamental arguments against using XML technology are difficulty in creating the content in XML format, converting content to XML and rendering XML data in formats specific to the application. In the problem space for Epocrates, allowing content to be authored in Word solves the problem of content authoring and conversion, and efficient encoding and rendering of XML on mobile devices addresses the output concerns. This allows the entire workflow to be based on an XML pipeline architecture which has proven to have great advantages.

§ References

- XML encoding techniques for storing XML data on memory limited (mobile) devices
  David A. Lee, XML2006 conference
- XMill An Efficient Compressor for XML
  www.liefke.com/hartmut/xmill/xmill.html
- From Word to XML
  http://www.xml.com/pub/a/2003/12/31/qa.html
- XML in Microsoft Office Word 2003
  http://msdn.microsoft.com/msdnmag/issues/03/11/XMLFiles/
- Microsoft Office Word 2003 XML Software Development Kit
  http://msdn2.microsoft.com/en-us/library/aa223586(office.11).aspx

The Author

David Lee
Epocrates, Inc.
727 Poplar Lane
Jasper
IN
47546

David Lee has over 20 years experience in the software industry responsible for many major projects in small and large companies including Sun Microsystems, IBM, Centura Software (formerly Gupta.), Premenos, Epiphany (formerly RightPoint), WebGain. As senior member of the technical staff of Epocrates, Inc., Mr. Lee is responsible for managing data integration, storage, retrieval, and processing of clinical knowledge databases for the leading clinical information provider. Key career contributions include Real-time AIX OS extensions for optimizing transmission of real-time streaming video (IBM), secure encrypted EDI over Internet email (Premenos), porting Centura Team Developer, a complex 4GL development system, from Win32 to Solaris (Gupta, Centura), optimizations of large Enterprise CRM systems (Epiphany), implementation of ecommerce systems for on-demand digital printing and CD replication (Nexstra).

Extreme Markup Languages 2007®
Montréal, Québec, August 7-10, 2007
This paper was formatted from XML source via XSL
by Mulberry Technologies, Inc.