How To Customize OpenOffice.org To A Dedicated XML Authoring Tool

---

A Case Study Of China Patent Editor

Guang Gavin Lu
<Gavin.Lu@Sun.Com>

Sun China Engineering and Research Institute
10/F., Innovation Plaza, Tsinghua Science Park
Beijing, P. R. China
Tel: +86-10-82618200
Fax: +86-10-62780969

Openoffice.org Conference @ Berlin, Germany, September 2004

How To Customize OpenOffice.org To A Dedicated XML Authoring Tool

A Case Study Of China Patent Editor

Table Of Contents

How To Customize OpenOffice.org To A Dedicated XML Authoring Tool

A Case Study Of China Patent Editor

Abstract

As the Internet spreads greatly, patent applications worldwidely are moving onto electronic filing systems with defined patent XML file formats. Accompanied by patent offices of Europe, U.S., Japan and WIPO, State Intellectual Property Office of China (SIPO) also defined its patent XML file format, SIPO XML and started electronic applications. But one usable, stable, and cost-effective patent XML authoring software is still absent for Chinese patent applicants, attorneys, and examiners.

OpenOffice.org is a free productivity suite compatible with all major office suites. It's stable, crossing-platform, free, and usable enough. OpenOffice.org utilizes XML as its native file formats, and supports XSLT filters to import in and export out external, 3rd party, customized XML file formats. So OpenOffice.org is a good candidate to be customized to China Patent Editor.

This article mainly introduces the project of customizing OpenOffice.org to China Patent Editor.

Patent Applications In China

The Patent Office of China (CPO) was founded in 1980 to protect intellectual property, encourage invention and creation, help popularize inventions and their exploitation, and promote the progress and innovation in science and technology. In 1998, CPO was renamed State Intellectual Property Office (SIPO)[1]. In 1984, the Patent Law of China was promulgated and went into force in 1985, which was amended twice in 1992 and 2000.

Once the Patent Law of China was in force in 1985, patent applications grew greatly year by year. The annual patent application number in 1985 was only 14,372, but the total number reached one million in 2000. In 2002, the annual patent application number became 252,632, and exceeded 300,000 in 2003. In March 2004, SIPO got its two millionth patent application.

In the mean time, China joined the patent cooperation in the world. In 1980, China acceded to the Conversion Establishing the World Intellectual Property Organization. It's now member of various international treaties and agreements on patent protection. Just after China joined The Patent Cooperation Treaty (PCT)[2] in 1994, it began to serve as the Receiving Office, International Searching Authority and International Preliminary Examining Authority. Chinese also became one of the PCT working languages.

In order to improve the performance and achieve better efficiency and effectivity, cut down costs of applicants and patent offices, many countries are deploying or have deployed electronic patent filing systems. Japan Patent Office (JPO) has achieved great electronic application ratio, European Patent Office (EPO), United States Patent and Trademark Office (USPTO) and World Intellectual Property Organization (WIPO) are deploying their electronic filing systems too, whose DTDs are all PCT XML DTD[3] or its derivations. SIPO also started its electronic filing system[4] with its own much simpler probative patent XML DTD[5] on March 13, 2004.

But actually the electronic filing system could not solve all issues within one night. One is that, before the debut of electronic filing system of SIPO, there were already more than 2 million patent applications in paper. It had taken hundreds of employees about half a year to convert them into computer documents. But in this methodology they were just converted into pure text or word-processing files, not in patent syntax of SIPO yet.

Another issue is that the common authoring tool of SIPO patent applications is Microsoft Word. Neither it can directly save files into SIPO XML format, nor any external tools convert Word files to SIPO XML files. Microsoft Word is expensive too. So it's necessary and urgent to design and implement one authoring and converting tool for SIPO XML, especially for its patent claims and descriptions.

Overview Of SIPO XML

Including PCT, EPO, USPTO, JPO and SIPO, all XML DTDs of patent electronic filing systems are composed of patent application documents and other data type definitions of auxiliary documents or forms, such as fee sheets, package data, dispatching lists, receipts and so on. The most important and complex part is patent application documents.

In SIPO XML DTD, all the auxiliary documents are designed as forms, defined by a bunch of XML DTD files of simplified Chinese. Because they were estimated to be used only by Chinese and not share within WIPO PCT, these DTDs were not described in English at all.

It's not too difficult to make out system tools to implement SIPO's electronic filing system, and comply these simple auxiliary documents' definitions, whereas it's totally different for patent application documents.

SIPO XML DTD

Illustration 1 is the basic structure of patent application documents defined by SIPO XML DTD.

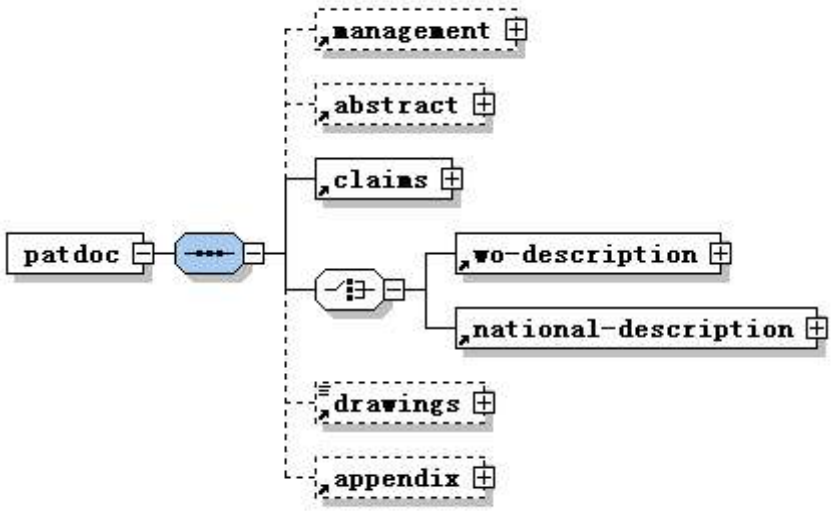

Illustration 1 SIPO XML DTD basic structure of patent applications

- <management> represents its management information such as application numbers, possible patent numbers, and page numbers of separate sections.
- <abstract> means the abstract of patent application description body, composed of one or more paragraphs. Before all these paragraphs, there could be one heading.
- <claims> is composed of one or more claim items, before which a heading could exist too.
- Either <wo-description> or <national-description> but not both, represents the body of

patent application.

- <wo-description> means PCT patent description if it's a PCT patent application, including six sections of technical field, background art introduction, disclosure of invention essential, brief description of appending drawings, best mode of application, and industry applicability, or just one prosaic description. One title of invention can exist before all these sections or prosaic description.
- <national-description> means Chinese patent description if it's just a national patent application of China, including its description and an optional title of invention before it.
- <description> is composed of headings, paragraphs, images and any spaces, such as page breaks, line breaks, tabs and blank spaces.

- <drawings> represents appending images, chemistry formulas, mathematical formulas, tables, and their brief unadorned captions.
- <appendix> is composed of appendix description and one optional heading before it.

Because SIPO had accepted above two million patent applications before it started its electronic filing system, and had executed for about ten years before it joined the PCT in 1994, SIPO has its own filing processes and designed its own XML DTD by itself.

Although that, SIPO XML DTD isn't totally irrelative to WIPO PCT XML DTD and others XML dialects. In SIPO XML DTD W3C MathML 2.0[6] is utilized to represent mathematical formulas, CML 1.0[7] utilized to represent chemistry formulas, and Exchange Table Model DTD, OASIS Technical Memorandum TR 9901:1999[8] (XML expression of the exchange subset of the full CALS table model DTD, and XML version of OASIS Technical Resolution 9503:1995) utilized to represent tables. These three modules are also utilized by PCT XML DTD of WIPO directly or indirectly.

Defects

Unfortunately, this SIPO XML is not a classic XML dialect just representing its logic data without presentation styles. SIPO XML defines both data to represent logic data and contents relative to patent application itself, and some basic representing layouts, such as various spaces, alignments, font styles, widths and heights and so on. But it doesn't define other elements and attributes of representing styles, such as indents, page styles, paragraph styles, so that SIPO XML can't determine completely the representing styles of its patent XML files. This way is similar to other patent XMLs, which creates the same issues. It's impossible to create a distinct representation for one SIPO XML file, because it contains certain so blurred sections that different tools can define them differently by themselves.

One object of the patent electronic filing systems is to manage patent data electronically and pay more attentions to patent contents and logic data and less to their representing layout styles. So it's a

better way to adjust SIPO XML to only defined logic data and provide converting XSL-FO and XSLT definitions and tools if it's necessary to put them into print unifiedly.

Another issue is that it's incomplete in certain elements. For instance that mathematical formulas can't be included as well as in W3C MathML 2.0, because <mathml> element in SIPO XML is just defined as a string.

Another issue is incoherent. Take patent claims and lists as an example. In SIPO XML's definition, patent claims include one or more claim items, but it's quite weird that claims also include lists. And for claim items, they could include one or more paragraphs, which's quite good. But for list items, they include not paragraphs but #PCDATA, accompanied by sub lists. So actually list items could not have several sequential paragraphs separated by sub lists, which's weird too.

Fortunately SIPO has realized these issues and decided to solve them in the future revision.

Requirements And Concepts

SIPO wants that its electronic filing system contains the following features:

1. It could edit patent descriptions and its claims in electronic filing system. Because SIPO XML DTD changes gradually, this software must be extensible;
2. It's implicative to assign elements and attributes of layout styles;
3. It's handy acceptably to assign elements and attributes of contents and logic data;
4. It could trace the XML tree during authoring, and inform users on invalidation against SIPO XML DTD if necessary, and help to create validated documents;
5. It could show the XML tree whenever asked;
6. It could define many represent styles, at least for patent descriptions and claims;
7. It could accept objects created by other programs, and change them into images;
8. It could cooperate with other authoring tools for specific XML dialects such as MathML;
9. It could convert those legacy Microsoft Word and pure text documents into SIPO XML files, while it's acceptable to interoperate with users by questions/answers and wizards.

This system is expected to run on users' familiar Microsoft Windows, and developed based on a mature word-processing software with similar user experiences to Microsoft Word, as an add-in or by customization.

Users

It's obvious that Microsoft Word, the accustoming word-processing software, has already framed its users greatly. Microsoft Word allows its users to write down their articles freely, but an XML authoring tool is definitely putting down more restrictions on its users. Every user of Microsoft Word perhaps has his or her own writing habits or styles, but there're probably only several limited ways or even one way to authoring XML documents in one XML authoring tool. So it's quite clear that it's impossible for every patent applicants to like and be accustomed to this new patent XML authoring software as soon as it's released.

Actually, the users of SIPO XML authoring tool can be classified into three groups:

- Inventors or applicants: They're quite familiar with relative technical areas and probably often write technical papers. They usually utilize various kinds of materials in their patent applications, such as texts, engineering graphics, mathematical or chemical formulas, programming codes, and biological deposit information.
- Patent attorneys: They're quite familiar with patent files as law documents, but much less familiar with the relative technical contents. In order to create the application materials,

they've to compose stuff from the patent applicants and other channels, and even translate them to Chinese for foreign applicants. So probably they'll misunderstand and misrepresent implied meanings in the materials, for instance, creating or conserving space-separated tables.

- Patent examiners: They receive and examine patent application materials from inventors, patent applicants and attorneys. They'll not author SIPO XML documents, but probably correct slight errors in those materials. They're also in charge of electronizing legacy paper-based patent achieves and converting old files into SIPO XML documents.

Relative Features Of OpenOffice.org

Fortunately OpenOffice.org owns almost all the necessary features to meet SIPO's needs.

From OpenOffice.org 1.1 on, There's an XSLT architecture[9] supporting to import from or export to external, user-defined or 3rd party XML file formats. This XSLT filter architecture actually came out from XMerge[10] project, which's to support editing rich format documents on small digital devices such as Palm and PocketPC, and provide the ability to merge these files back to OpenOffice.org. XMerge is implemented in Java, and also supports transferring between 3rd party XML files with customer-defined DTDs or schemas and OpenOffice.org XML[11] documents, by Xalan-Java.

Illustration 2 is a simple demo of XSLT based filter architecture of OpenOffice.org. During the importing process, external XML files are parsed by XML parser within importing filters, converted by transformers of XSLT into OpenOffice.org XML formats, and then imported by XML importers into internal documents models. The exporting process is almost reversed.

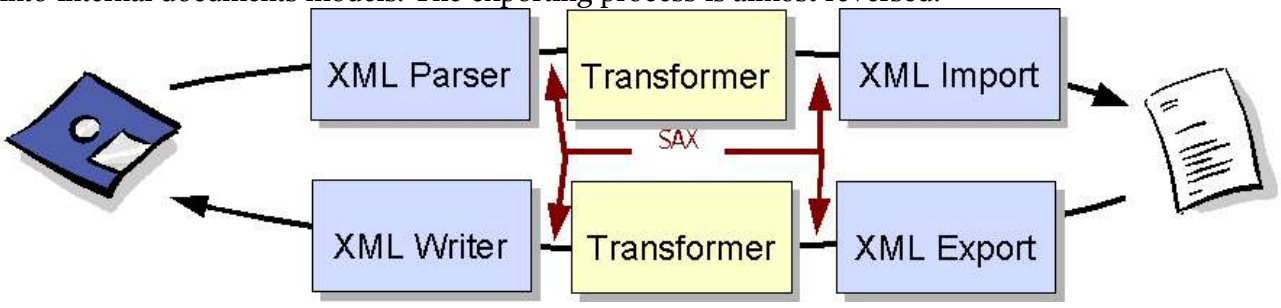

Illustration 2 XSLT based filter architecture of OpenOffice.org

If only the importing/exporting conversion logic, or the XSLT transformation scripts are implemented and necessary configurations of external XML file types and filters are registered, these XSLT based filters are finished. For instance, DocBook[12] is a favorite XML format of technical articles and books. OpenOffice.org has already integrated its importing and exporting XSLT based filters. One Docbook template of Writer also is combined because it's necessary to name some special styles for Docbook, by which the XSLT scripts can know how to map relative elements and attributes between 3rd party XML DTDs or schemas and OpenOffice.org XML DTD, perhaps even by complex iteration, calculation and conversion.

Although it's not easy for OpenOffice.org now to build concurrent XML trees of external XML file

formats while it's running, its SDK programs can make certain validation against external XML schemas or DTDs by itself. Here come out two totally different processes on the sequence of transformation and validation.

In illustration 3 there're two completely different workflows of XSLT transformation between SIPO XML and OpenOffice.org XML, and validation against SIPO XML DTD by OpenOffice.org SDK programs.

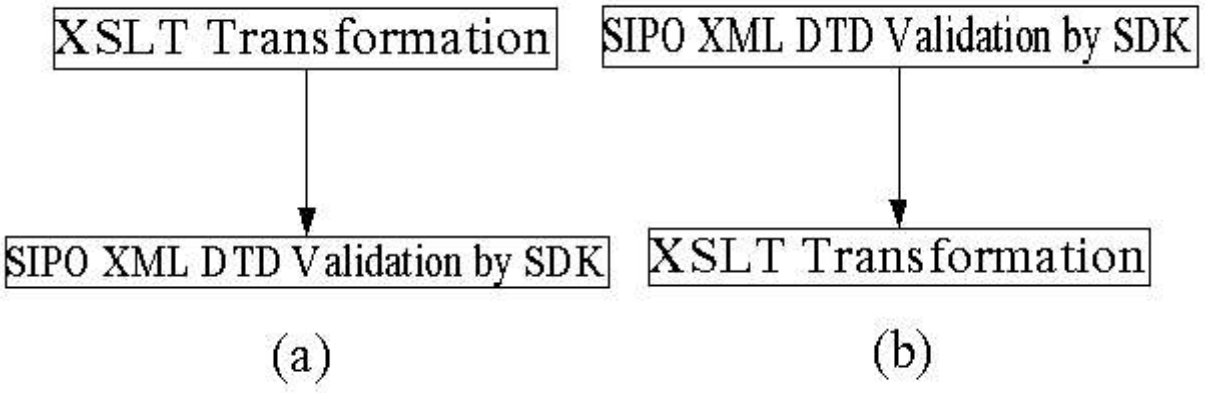

Illustration 3 Scenarios of XSLT transformation and Validation against SIPO XML DTD by SDK programs

(a) sequence shows that the XSLT transformation is before SIPO XML DTD validation by SDK programs, whereas (b) sequence shows that SIPO XML DTD validation by SDK programs is before the XSLT transformation.

Let's analyze these two sequences during both importing and exporting periods.

During the importing period of SIPO XML files, if (a) sequence is applied, XSLT transformation from SIPO XML to OpenOffice.org XML should be executed firstly. It's relatively simple. But in this way it's assumed that SIPO XML files are OK or any possible invalidation is just ignored because for XSLT scripts it's almost impossible to forecast, detect, or recover arbitrarily invalidated files. It's too burdensome for XSLT scripts to achieve acceptable performance even if they could detect certain errors. So these input files have to be assumed validated SIPO XML files. As a consequence, it's impossible for the following validation against SIPO XML DTD by SDK programs to find out any possible invalidation because the documents are already in OpenOffice.org internal document model after the former XSLT transformation, either filtered or ignored.

It's quite similar during the exporting period. As soon as SIPO XML files are exported out by the XSLT transformation, there isn't any invalidation which can be detected against SIPO XML DTD by SDK programs.

So far as for (a) sequence, it's improper for the combination of the XSLT transformation and validation against SIPO XML DTD by OpenOffice.org SDK programs.

So how about (b) sequence?

During the importing period, the validation against SIPO XML DTD by OpenOffice.org SDK

programs is before the XSLT transformation. So it's quite easy for SDK to validate external SIPO XML files against its DTD before they're imported into OpenOffice.org internal document model. After that, it's also a clear way to transform SIPO XML to OpenOffice.org XML.

Well, during exporting period it's a little more complex. If we'd like to make validation against SIPO XML DTD by SDK programs before the XSLT transformation, It's necessary to hardcode its DTD into SDK programs because at that time XSLT transformation hasn't executed and documents are still in OpenOffice.org XML format. Inherently it's because XSLT based filter architecture utilizes not DOM interface but SAX stream, so that it's very difficult for SDK programs to build one XML tree of external XML file format while it's still in OpenOffice.org internal document model. This difference makes it very difficult for current OpenOffice.org to support realtime validation against any external XML DTDs or schemas, except that an XML tree is created and maintained by OpenOffice.org SDK programs themselves. There's another benefit of (b) sequence: XSLT transformation and validation against SIPO XML DTD with SDK can be implemented by different engineers in parallel. Less risks will appear at the rigid timetable.

Finally it's decided to choose (b) sequence after serious consideration and comparison of these two options. But the practical process of customization to an XML authoring tool is much more complex than this prototype analysis.

Process Of Customization

Analysis of SIPO XML DTD

It's not very easy for 3rd party developers to understand and grasp the essentials of SIPO XML DTD, because it doesn't contain complete documents, it isn't a set of complete data type definitions, and it even isn't a set of coherent DTD either. But this kind of external XML formats probably often appear in projects.

The most effective way to understand it is discussion, discussion, and still discussion. During the analysis process of SIPO XML DTD, engineers and SIPO experts held many technical meetings, almost once weekly at first and if only necessary afterwards. Whenever we have any problems on it, we can connect with them immediately and solve the issues.

Another merit of our process is that we can develop XSLT transformation and validation against SIPO XML DTD with SDK programs in parallel. So we can discuss the DTD ourselves and make it clearer.

XSLT Transformation

XSLT transformation of SIPO XML includes three major parts: the importing and exporting XSLT scripts, OpenOffice.org Writer templates for SIPO XML, and the configurations.

Actually the importing/exporting XSLT scripts and templates could be packed into configured JAR package, which could be released separately and added by users themselves. This's beneficial for any upgrading of SIPO XML XSLT transformation later. But at first time, this XSLT transformation must be integrated into the installation set, so that users could easily install them once together.

Illustration 4 shows the one property page of the configuration of SIPO XML filter in simplified Chinese. The file type and filter name are both “SIPO XML”, the application is Writer rebranded with “China Patent Editor”, the file extension name is “patxml”, and its docType (in unshown transformation property page) is “patdoc” as declared root element of SIPO XML DTD. Because SIPO XML DTD is composed of four module files, and never used in this XSLT transformation directly, it isn't defined or included in the configuration at all.

In order to add SIPO XML transformation natively into OpenOffice.org installation set, three modules: filter, officecfg, and scp are modified.

- In filter module, a directory “sipoxml” is created and importing/exporting XSLT files, one template, and makefile.mk are added. The prj/build.lst is also modified.
- In officecfg module, TypeDetection.xcu in registry/data/org/openoffice/Office is modified to add configuration items for file type “SIPO XML” and filter name “SIPO XML”.

- In scp module, inc/gid.inc, source/office/dir.scp, source/office/javafilter/*_javafilter.scp are all modified to add new SIPO XML filters. Additionally, in order to make SIPO XML template visible as soon as installed (ordinarily templates of XSLT filters are invisible firstly because they're not in normal template directories), source/office/files.scp and modules.scp are modified to put this template into the normal template directory too.

Illustration 4 Configuration of SIPO XML filter

The SIPO XML template makes up a skeleton of SIPO XML filters, by mapping necessary SIPO elements and attributes into OpenOffice.org nodes and converting between them.

- Management ID, application number and patent number are mapped to user-defined meta data, “application catalog” is also defined in order to classify whether national application or PCT application.
- Pages of abstract, claims, description, drawings, and total application are declared as user variable input fields, and two other fields “doc_pic_dir_name” and “image_suffix_max” are also defined in order to meet image operations of SIPO XML files, mentioned later.
- Several specific fonts are defined and declared in according to abide SIPO's rules on how to authoring patent application materials. The representation of SIPO XML files by specific fonts can help to assure their exactly same appearance, which also locks itself to certain font vendors.

- For mapping SIPO XML contents such as paragraphs, headings, sections, graphical types, claims and lists, to OpenOffice.org XML contents, special paragraph, heading, page, header, footer, graphics, list styles and their relative attributes are defined. Default properties of OpenOffice.org XML styles are also defined for those undefined properties of SIPO XML styles, so called rules by SIPO operations. Users have to manually assign correct styles onto contents.

There're several other so difficult issues that it's impossible to simply map them to OpenOffice.org XML elements and attributes.

- <literal> and <program-listing> in SIPO XML represent raw-formatted texts, including spaces, tab-stops, new-lines, and so on. But these format controllers are already represented by special elements in OpenOffice.org XML file format. Regression and iteration templates of string processing for multiple keywords are quite complex and probably make stack overflow and program crash. It's not a good idea. We finally map them to paragraphs in OpenOffice.org XML, and convert them back to paragraphs in SIPO XML, with either one <literal> or <program-listing> and normal content elements in it. In this way at least we could save necessary data and keep valid against SIPO XML DTD, although a little redundant and error-prone.
- Vertical cell mergence in tables of SIPO XML is represented by the @morerows attribute of <entry> elements, which conflicts greatly with the representation of OpenOffice.org XML by sub table elements in cells. Because of the restriction of XSLT scripts in pure SAX, it's almost impossible to build one table data structure in memory and convert between these two table representations. So just horizontal cell mergence is implemented in SIPO XML tables and vertical cell mergence is left to later improvement, perhaps involved with Java extensions.
- In order to map those unordered and interincluded text properties of SIPO XML: <overscore>, <underscore>, <smallcaps>, <italic>, <bold>, <superscript> and <subscript>, these relative templates in XSLT scripts include regressions of applying their own templates. For example, a template for <bold> applies a template for <smallcaps>, which applies an template <underscore> again, which applies another template again and again, till finally a template for pure text is applied.
- All images in SIPO XML files are linked from ordered image files in a specific named sub directories, but OpenOffice.org can either link in or embed images. So embedded images must be transferred into linked images when SIPO XML files are saved in OpenOffice.org. These embedded images converted to linked images are named in order, as new files after those existing linked images. OpenOffice.org SDK programs help greatly to convert embedded and outside linked images into normal linked images. Automatically converting objects to images is left to improvement in the future.

SIPO XML DTD Validation And GUI Customization

As formerly mentioned, the validation against SIPO XML DTD is executed before the XSLT transformation, so that it's necessary to hardcode SIPO XML DTD into those validating SDK programs. Actually, these OpenOffice.org SDK programs are composed of two parts, the calling OpenOffice.org Basic macros and called Java programs as one UNO component defined by UNO IDL. In this case:

- OpenOffice.org Basic macros work as the entrance of validation process against SIPO XML DTD, and execute certain simple jobs such as looking for empty paragraphs and continuous blank spaces, checking whether correct styles assigned, and calling the XSLT transformation to export SIPO XML documents.
- Java SDK programs work for more complex jobs: checking word by word whether correct properties of styles are assigned, whether various SIPO rules are exactly abided, guiding users to correct errors in an intuitive way. All the validation work is based on the hardcoded SIPO XML DTD and undocumented rules, which're really hard jobs. In order to convert embedded images into linked images, image filters and Base64 decoding programs are also included, with temporary OpenOffice.org flat XML documents created, decompressed, and images fetched out.

One issue SIPO found is that, for a patent application description as long as dozens or hundreds of pages, it's unbearable to take several minutes or longer, just to make validation against SIPO XML DTD once. But SIPO XML DTD and default rules are quite rigid. For example, continuous blank spaces and empty paragraphs must not exist, and all the contents must be applied those designed styles. Whereas users often type continuous blank spaces or empty paragraphs, and forget to manually assign SIPO dedicated styles to contents. So once validation programs find these errors, warning messages will show out to users. If users don't carefully assign styles and edit their documents before they start validation progress, they'll meet with seemingly endless noisy messages.

By SIPO's suggestion, the progresses of checking blank spaces and empty paragraphs are separated from the main validation process and put before it, so that users may check them firstly and then the rest. In this way the average validation time is shortened greatly and user experience is improved.

Another problem SIPO brought out later is that, when users unmeantly delete invisible page breaks before every section titles, it's quite difficult for them to recover back to normal document structure. Because actually every section title is bound with one set of master page and page master styles for this section, with different titles, footers, headers, and customizedly restarting page number fields. This section structure can't be recovered by simply insert new page breaks from menu items again. Although the best and simplest way is to just undo the last operation by pressing “Ctrl+Z” combination key once this structure is broken, common users don't know much enough knowledge upon word-processing document structure and probably are scared by the disorder already.

So several dedicated buttons and menu items are created for users to reconstruct the whole document structure, section by section, from the beginning to the end.

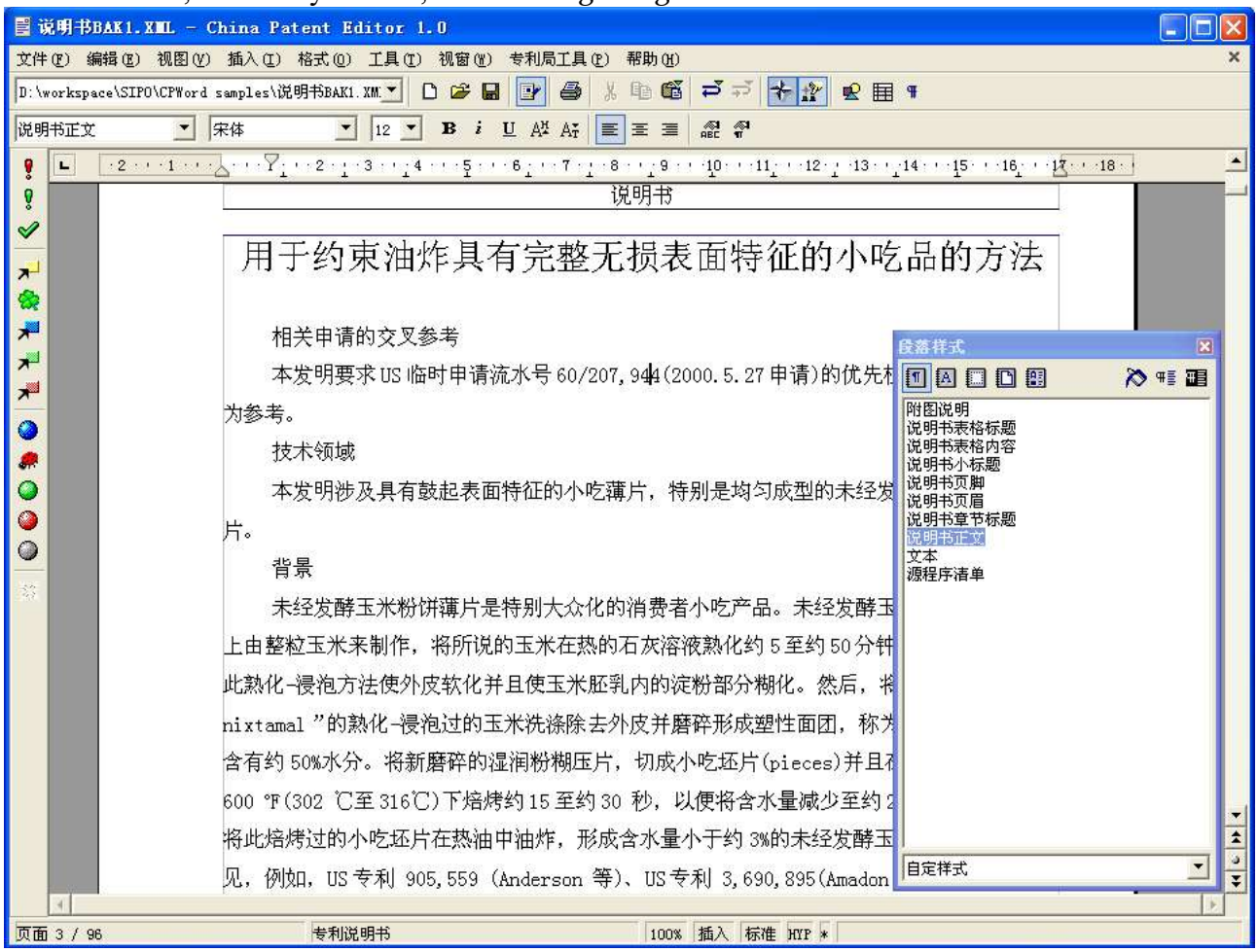

Illustration 5 Validation against SIPO XML DTD and GUI Customization

In Illustration 5 shows a screenshot of customized OpenOffice.org program. The menu bar, standard bar and formatting bar are configured to only show necessary and helpful items and widgets in order to avoid any possible misleading and complexity. The tool bar is designed completely for reconstructing the whole document structure, jumping to section headings, and validating documents against specific parts of SIPO XML DTD and default rules.

Rebranding Work

The rebranding work of China Patent Editor is relatively simple, because SIPO would like to cooperate with open source communities and show their interest in free and open source software and endeavor to cut down their cost of total ownership of software.

Under this friendly spirit, replacing OpenOffice.org icons is stripped off from the rebranding work, so that only four modules are necessary to modify:

- In order to change the installation, flashing introduction and about images, sipo_product.bmp in readlicense_oo/source/cinema/, sipoabout.bmp and sipointro.bmp in offmgr/res are created as SIPO's new images in the same sizes as in OpenOffice.org.
- In order to change the license and readme information, license.txt and license.html in readlicense_oo/source/license/wnt/, and readme.txt in readlicense_oo/source/readme/wntmsci8/ are replaced with corresponding information SIPO provide.
- In offmgr/source/offapp/intro/, ooo.src is modified to change variables like $OOO_VENDOR and $OOO_INTRO, and simplified Chinese information within OpenOffice.org help-about dialog.
- In scp/source/office/, config.scp is modified in order to replace the global variables: $ProductName, $ProductVersion and $VerdorName.

Illustration 6 is the flashing introduction image of rebranded OpenOffice.org, “China Patent Editor”.

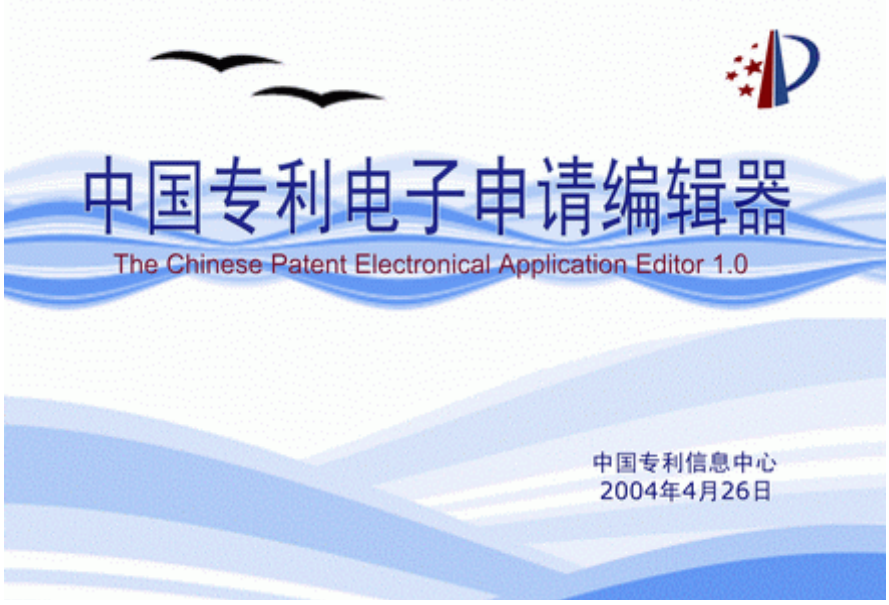

Illustration 6 Flashing Introduction Image of China Patent Editor

Conclusion

This project of customizing OpenOffice.org to China Patent Editor was successfully finished within three months. SIPO was satisfied that the assigned deadline was met and major features were implemented.

After the reflection of this project, it's clear that:

- OpenOffice.org is definitely capable to be customized to an XML authoring tool and a good basic platform for it because of its outstanding XSLT filter architecture. This project is just another implementation of this kind of applications.
  - It seems that certain data model of OpenOffice.org XML file format isn't good enough and need upgrade, such as the horizontal cell mergence model in Writer tables.
- If the validation process against external, 3rd party, and customized XML DTDs or schemas is necessary,
  - OpenOffice.org SDK programs are necessary to implement this validation with any languages OpenOffice.org supported or their combination.
  - Due to the restriction of SAX interfaces in OpenOffice.org, it's almost impossible currently to build realtime XML trees while editing. The practical methodology is checking periodically or nonperiodically.
  - Because OpenOffice.org doesn't support enough restrictions all over its document module, it's necessary to make good templates and checking various stuff as much as possible. More restriction properties of OpenOffice.org XML file format or powerful controls and forms seem beneficial for it.
  - The program process is better to be validation against external XML DTDs or schemas, and then XSLT transformation for both importing and exporting. Sometimes it's acceptable to remove the validation step of importing files, just assuming they're validated.
  - Although OpenOffice.org SDK API is quite powerful already, its improvement is necessary and more relative documents and examples are beneficial too.
- In order to implement great XML authoring tools based on OpenOffice.org XSLT filter architecture, it's better that external, 3rd party and customized XML DTDs or schemas are well defined: complete, consistent, and reasonable.
  - If they focus on logic data contents, it's better to just include data contents only without any styles. If they also want to define certain representation styles, it's better to make complete XSL-FO and XSLT definitions for publishing separately.
- It's quite easy to rebrand OpenOffice.org.

Team work spirit is essential to the success of this project. Issues are almost always solved at

frequent joint meetings of engineers, managers and customers. Discussion, discussion, and still discussion are all our killer processes. Due to the tight timetable, we actually implemented eXtreming Programming (XP) in certain degree.

Acknowledgment

It's great to work with great colleagues. In this project to customizing OpenOffice.org to China Patent Editor, Xu Robert Chen and Jian-guo James Meng worked with me together to implement all the codes. Robert and James worked on the validation against SIPO XML DTD and GUI customization, while I focused on the XSLT transformation and rebranding work. Li William Wan, our engineering manager, and Jing Sophia Zhang, the marketing manager, involved greatly in the early communications with SIPO customers and engineering meetings afterwards. It's critical to have their leadership along and help to solve all technical and nontechnical issues. Our customers from SIPO also provided beneficial ideas and advices during the whole period, made probationes and reported bugs once the first working milestone was built out, which's quite helpful to accelerate the engineering progress, greatly appreciated too.

1 State Intellectual Property Office of China, http://www.sipo.gov.cn/
2 The Patent Cooperation Treaty, http://www.wipo.int/pct/
3 WIPO PCT XML DTD, http://www.wipo.int/pct-safe/epct/schemaDocs/1.2/schemaDocs.htm
4 Electronic Filing System of SIPO, http://www.cponline.gov.cn/
5 SIPO XML DTD, http://www.sipo.gov.cn/sipo/ztxx/zldzsqxt/t20040408_27707.htm
6 W3C MathML 2.0, http://www.w3.org/TR/MathML2/
7 CML 1.0, http://www.xml-cml.org/
8 XML Exchange Table Model DTD, http://www.oasis-open.org/specs/tm9901.html
9 OpenOffice.org filters using the XML based file format, http://xml.openoffice.org/filter/
10 Document Editing on Small Devices – XMerge, http://xml.openoffice.org/xmerge/
11 OpenOffice.org XML File Format Specification, http://xml.openoffice.org/xml_specification.pdf
12 DocBook, http://www.docbook.org/