Master's Thesis

University of Amsterdam
Faculteit der Natuurwetenschappen, Wiskunde en Informatica

November 2003

Section: Programmatuur

Under the supervising of:

In the Netherlands: Prof. dr. Paul klint
In New York at the United Nations: Sousa Jossai

A reengineered publication model based on: Reuse of a Document Management System And The presentation of an Enterprise Java Application

Mounia Belmamoune
mounia_bm@yahoo.com
mouniab@science.uva.nl

Acknowledgements

This Thesis represents a significant amount of work. It could not have been successfully completed without the help of many people. I would sincerely like to thank:

Prof. Dr. Paul Klint for inspiring me with many ideas and information about my master project.

Sousa Jossai for his great help and his generous suggestions.

My Family for being the most incredibly loving and supportive family.

My Friends for being terrific friends, giving me advise and support.

Abstract

Like many other large enterprises and organizations, the Treaty Section in the Office of Legal Affairs of the United Nations recognizes that content is an asset and that the ability to publish its products (CD-ROM's, books, etc) on the Internet accurately and cost-effectively can be critical for its success. However, in publishing its products the Treaty Section uses a very complex system requiring extensive human intervention, which leads to the following limitations:

- Published products are both badly-controlled and badly-maintained during their lifecycle.
- Within the actual publication system, the reuse of information and verification of it leads to enormous information delay.
- The presented data on the Internet is static; it is not presented dynamically in the view that is appropriate for the user.
- Many applications are involved during the production process, which causes system maintenance difficulties.

The objective of this thesis project has been firstly to *analyze the Treaty Section's publication system* in order to determine how the publication proceeds and what its limitations are and secondly *to present a reengineering model that will overcome all the limitations of the current publication process(es).*

Contents

Chapter 1

Introduction

The Treaty Section in the Office of Legal Affairs of the United Nations is entrusted, among others, with the registration and publication of international treaties and agreements.

The Treaty Section publishes the text of treaties and international agreements in all authentic languages and translations in English and French, as necessary and in different forms:

1- *Treaty series books*: a book, of about 500 pages, which is updated regularly. It includes text of treaties, subsequent agreements and actions.

2- *The publication of the status of multilateral treaties deposited with the Secretary-General, (called the "Bible").* This publication is updated daily and published on the Internet and as book at the end each year. There is an English and a French version of the "Bible." This publication is partly manually created and partly extracted from the Document Management System.

3- *Monthly statement*: This is a report on all original agreements registered in a given month or filed and recorded in a given month. It also publishes information on all subsequent agreements and actions to treaties registered, filed and recorded, registered with the League of Nations or related to the charter of the United Nations and statute of the International Court of Justice (ICJ) in English and French.

4- *Cumulative Index*: This is an index of all original agreements, subsequent agreements and actions published in a given range of Treaty Series volumes (typically 50 treaty series volumes). The publication is either in French or in English not in both languages.

5- *The depositary notification by the secretary general (also called CN), Journal, Cable, Letters, certificate of registration and the Checklist*: These are minor Internet and/or paper publications.

To publish the products mentioned above, the Treaty Section uses a very complex publication system that has served her needs for more than 6 years. During that time, it has been corrected, adapted and enhanced many times. People approached this work with the best intentions, but good software engineering practices were neglected. Now this system is suffering from many limitations such as:

- Many programs and applications are used during the different publication processes, which make the system difficult to maintain.
- With the actual publication system, the relationship between the content stored in the used Document Management Server and the published output is completely lost. The published content does not contain any relevant information that can link it back to the core in the Document Management Server in case of an update. This situation leads to data inconsistency between what is published and what is really stored in the server.
- The presented data on the Internet is static; it is not presented dynamically in the view that is appropriate for the user.
- When there is a change made on the data, the whole web page, affected by this change, has to be updated and reposted again, which is time and energy consuming.
- Bad search engine integration: the website has its own search engine. This situation leads to the situation of: ***what you see is not all what you have***.
- The workflow/Lifecycle not implemented**:** Workflow is missing in the actual system, which is necessary to decentralize content creation.

Because of those limitations, the publication system is unstable, it is still working but every time a change is attempted, unexpected and serious side effects occur. Yet, this system must continue to evolve. What to do?

A *good reengineering activity* seems to be the best solution for the Treaty Section in order to improve the publication processes and to overcome the different limitations of the actual publication system.

To realize a good reengineered model it was first necessary to analyze deeply the Treaty Section's publication system in order to understand its internal structure. The purpose of this project is to analyze the actual publication system and to present a suitable solution for the Treaty Section that will overcome the limitations mentioned above.

The word *reengineering* was and will be mentioned several times during this report, that is why it seems reasonable to understand first this activity.

Reengineering is a rebuilding activity that we can better understand if we consider an analogue activity: rebuilding a house. Consider the following situation [1]:

- Before you can start rebuilding, it would seem reasonable to inspect the house. To determine whether it is in need of rebuilding, you would create a list of criteria so that your inspection would be systematic.

- Before you tear down and rebuild the entire house, be sure that the structure is weak. If the house is structurally sound, it may be possible

to “remodel” without rebuilding (*at a much lower cost and in much less time*).

- Before you start rebuilding, be sure you understand how the original was built. Take a peek behind the walls. Understand the wiring, the plumbing, and the structural internals. Even if you trash them all, the insight you will gain will serve you well when you start construction.

- If you begin to rebuild, use only the most modern, long-lasting materials. This may cost a bit more now, but it will help you to avoid expensive and time-consuming maintenance later.

- If you decide to rebuild, be disciplined about it. Use practices that will result in high quality-today and in the future.

As in rebuilding a house activity, you will find in *Chapter 2* a detailed inspection of the actual publication system to determine which of its components can be kept and reintegrated in the new model. Then in *Chapter 3* a basic understanding of the most important technologies used in the Treaty Section’s publication system is given in order to deepen the knowledge about the internal structure and to facilitate the reengineering activity.
In *Chapter 4* an alternative solution of the actual system problems is presented and discussed.
Chapter 5, 6 and 7 contain my own reengineered model, which is presented with the most recent technologies.
Finally, a short conclusion is given in *Chapter 10.*

Keywords: Document/Enterprise Management System, Structured Documents, Meta-data, SGML and XML.

Chapter 2

Analysis

As a rebuilding activity of a house, it was necessary to analyze the existing system to determine whether it is in need of rebuilding before starting the reengineering process itself.

2. 1 Analysis of the existing system

From a business analysis perspective, the existing publication system has four logical processes:

1. *Content creation*
   - Integrated Authoring environment
   - Separation of Contents and Presentation
   - Content re-use
   - Cross-references between documents

2. *Content Management*
   - Version control
   - Workflow and lifecycle
   - Security
   - Integration

3. *Publishing*

4- *Presentation*
   - Books
   - Internet

Before giving a short description of the processes listed above, note that the publication system involves three important applications: A Document Management System (DMS), an authoring system and a conversion system to present the final products in the desired forms.

2.1.1 Content Creation

This is a process required by the content creators, who have to store the content and its properties in the DMS, where the content goes to the file system and the properties go to a local database *(see Chapter 3).* We will now briefly mention some important aspects of the *content creation process.*

- Integrated authoring environment

The DMS provides a good environment for content creators. The authors have access to the features provided by the DMS.

- **Separation of content and presentation**

It is not possible to publish to multiple formats without a strict separation of content and presentation. That is why the author stores the content in a central location (Docbase) and adds information (properties) to describe his content. The properties are also called *metadata* and are stored in a database.

- **Multi-user authoring**

The DMS can have many simultaneous users. Features such as record locking ensure that conflicting changes are prevented *(see Chapter 3).*

- **Content re-use**

A single content can often be used in different contexts.

- **Cross-references between documents**

Authors can create many cross links between documents.

2.1.2 Content Management

This is a process provided by the used DMS. The most important aspects of this process are given below.

- **Version Control**

The used DMS provides a version control feature, which is necessary for accountability, backup and disaster recovery *(see Chapter 3).*

- **Security**

The DMS provides security levels in order to protect the integrity of the content *(see Chapter 3).*

- **Integration with external systems**

The used DMS is only one of a number of other systems used to present the information on the Internet and as printed products.

2.1.3 Publishing

The publication of the different products follows *three distinct publication processes*. During those publication processes, all required attributes and the corresponding text,

when necessary, are extracted from the DMS and then manipulated with different applications before the final release of the required product.

Publication Process 1

The first described process is the publication process (Fig. 1) of Treaties Series books, Monthly Statement and Cumulative Index.

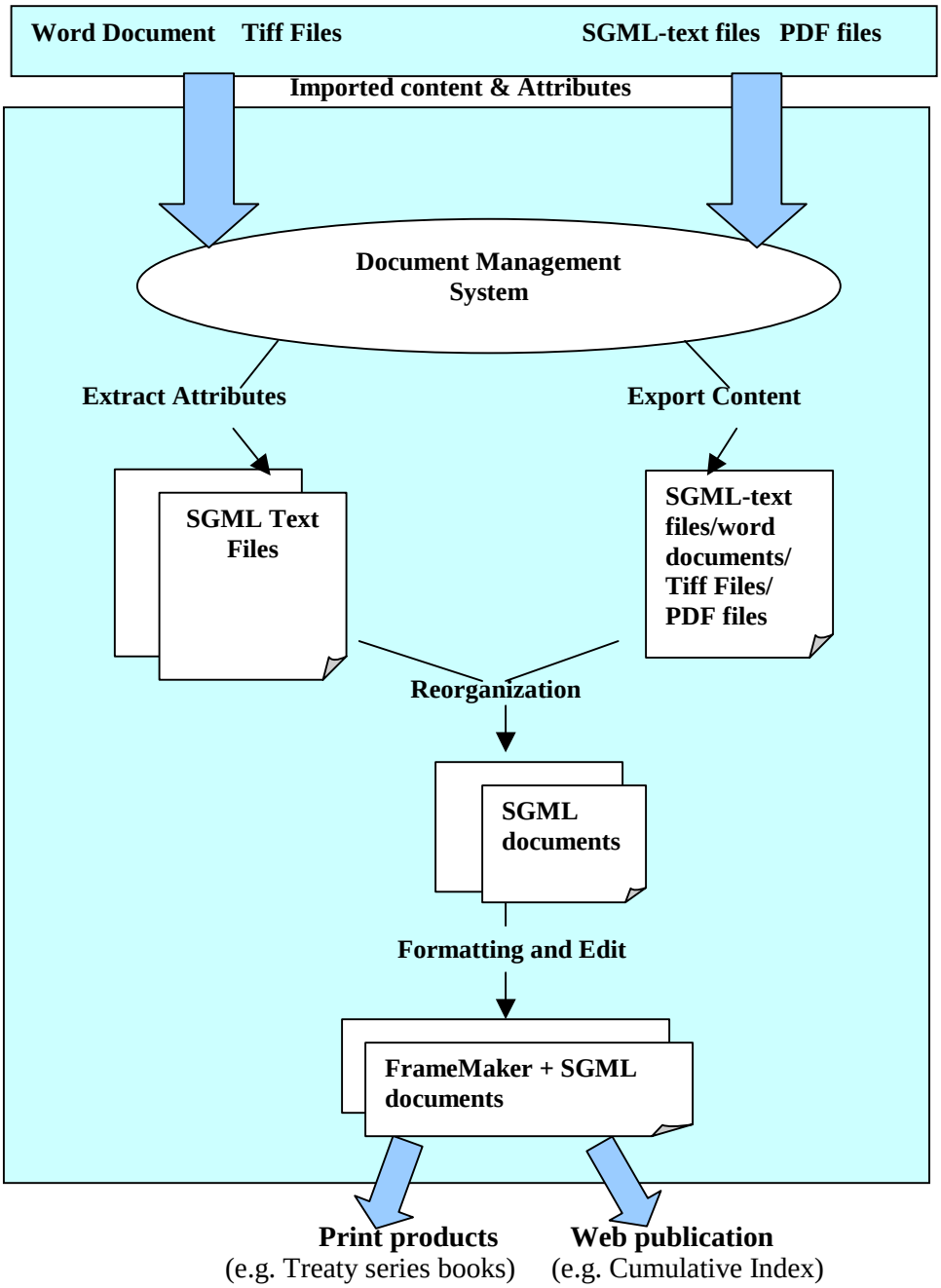


Fig.1 ***The functional diagram for publishing Monthly Statement, Cumulative Index and Treaty series books***

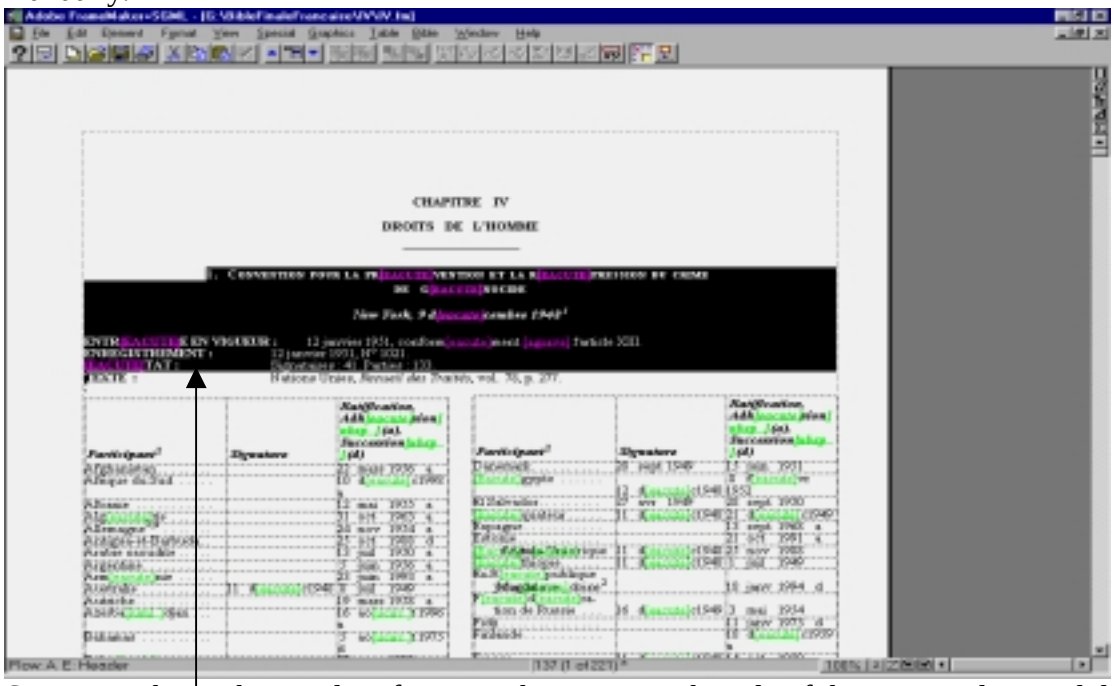
All the data, metadata and the content, are extracted from the Document Management System and formatted to SGML documents that are published.

This publication process is divided into four separate steps:

- The **extraction** of data from the DMS as SGML files using DocBasic[1] or Visual Basic.
- An optional **reorganization** of the SGML data using Omnimark[2] OMLE.
- **Formatting** and **Edit** of the SGML files into FrameMaker+SGML[3]. Each product has its own Document Type Definition (DTD).
- For Web **publications**: conversion of FrameMaker+SGML documents to PDF.

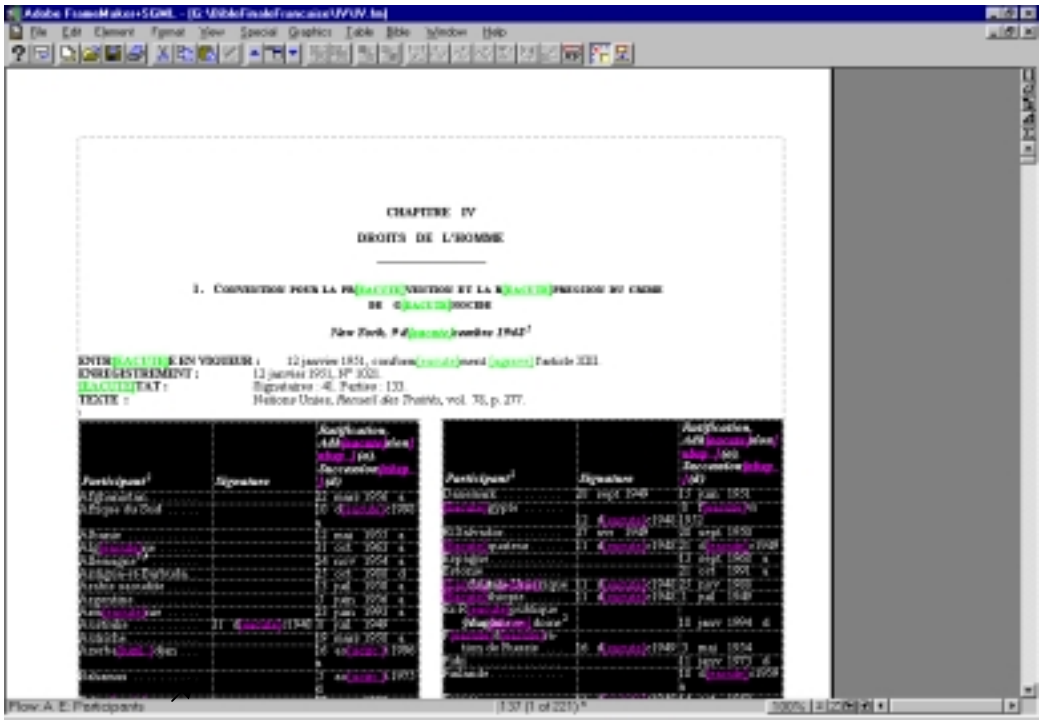
Publication Process 2

The second publication process concerns the "Bible" (Fig.2). In the "Bible", each treaty is composed of a header (*see Screen 1*), participant's table (*see Screen 2*) and other additional information (*see Screen 3*). The data of the header and the participant's table is extracted from the Document Management System, but other information is added manually.

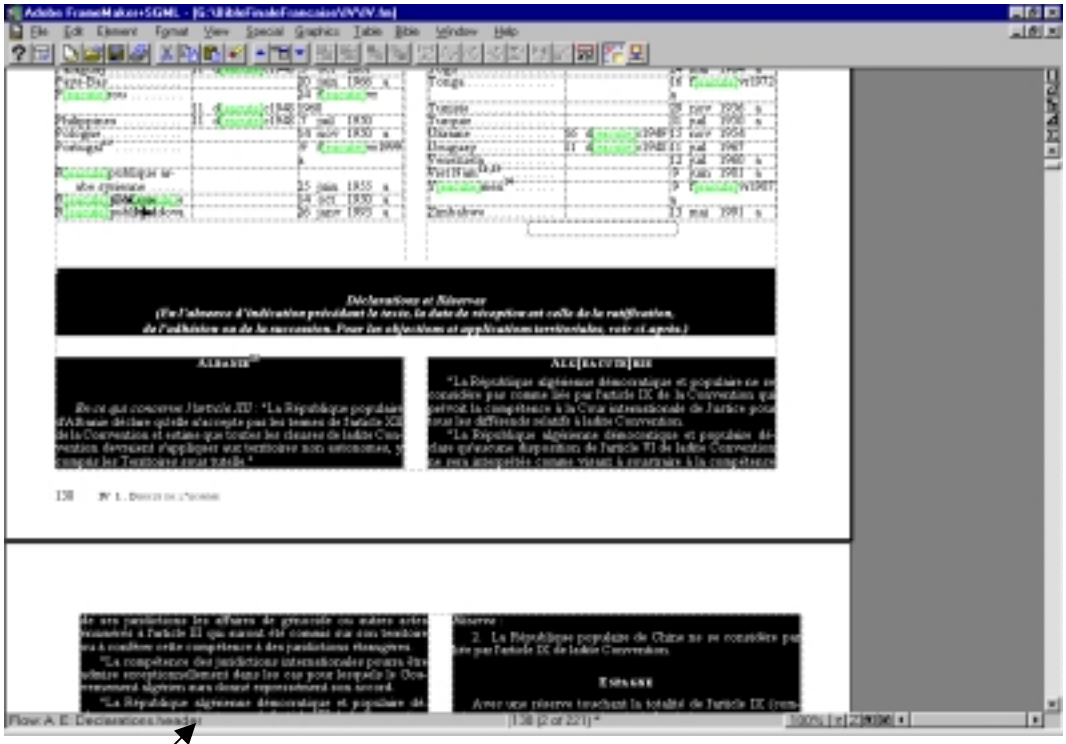


Screen 1: This is the Header of a treaty that contains the title of this treaty, place and date of conclusion and the registration information.

[1] Docbasic is a Documentum (the document management system used in the Treaty section) programming language integrated in Documentum. This language is similar to Visual Basic with few differences.
[2] OmniMark is a streaming programming language, for more information about this language, refer to http://developers.omnimark.com/
[3] Adobe FrameMaker+SGML is a powerful authoring system for publishing, for more information, refer to http://www.adobe.com/products/framemaker/



Screen 2: Participants table: a table of all participants that have signed this treaty.



Screen 3: Additional information, e.g. information related to some countries, this information has to be added manually via the authoring system.

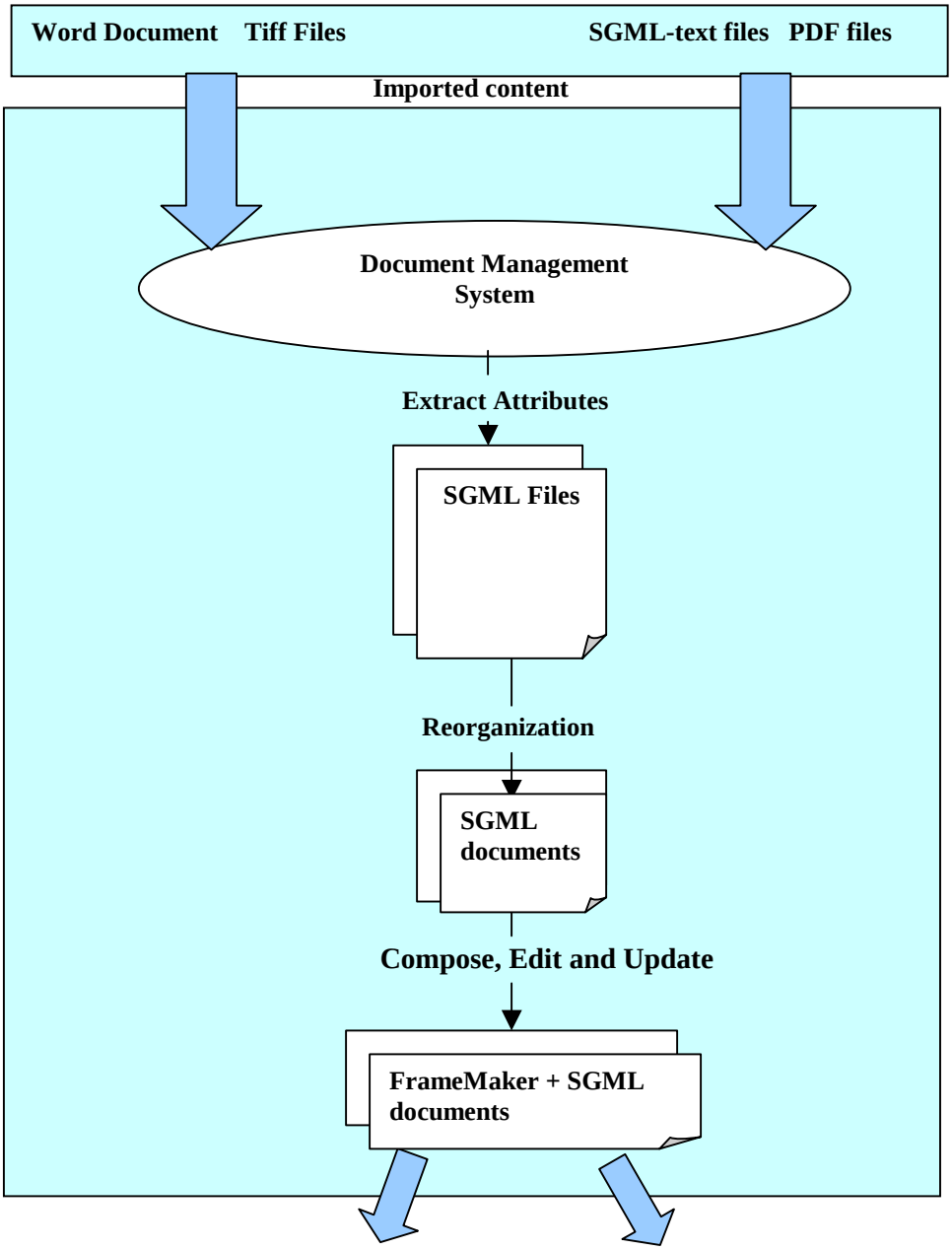


Fig. 2 The functional diagram for publishing the "Bible".

The publication process of the *Bible* is divided into the following separate steps:

Extraction: Functions developed in DocBasic or Visual Basic.
Omnimark programs provide functions for **reorganization**.

Compose and Edit functions are done through FrameMaker+SGML, at this stage the creator has to add manually additional information (*Screen 3*).
Update is done through customized menus in FrameMaker+SGML, developed with FrameMaker development kit (FDK) in C language in combination with Omnimark programs.
HTML **conversion** is provided by Omnimark.

Publication Process 3

This process (Fig.3) concerns the publication of Minor products such as CN, Journal, Cable, Letters, certificate of registration and the Checklist. Each document produced during this process is structured by a specific DTD so that it can be generated and composed automatically by programs as follows:

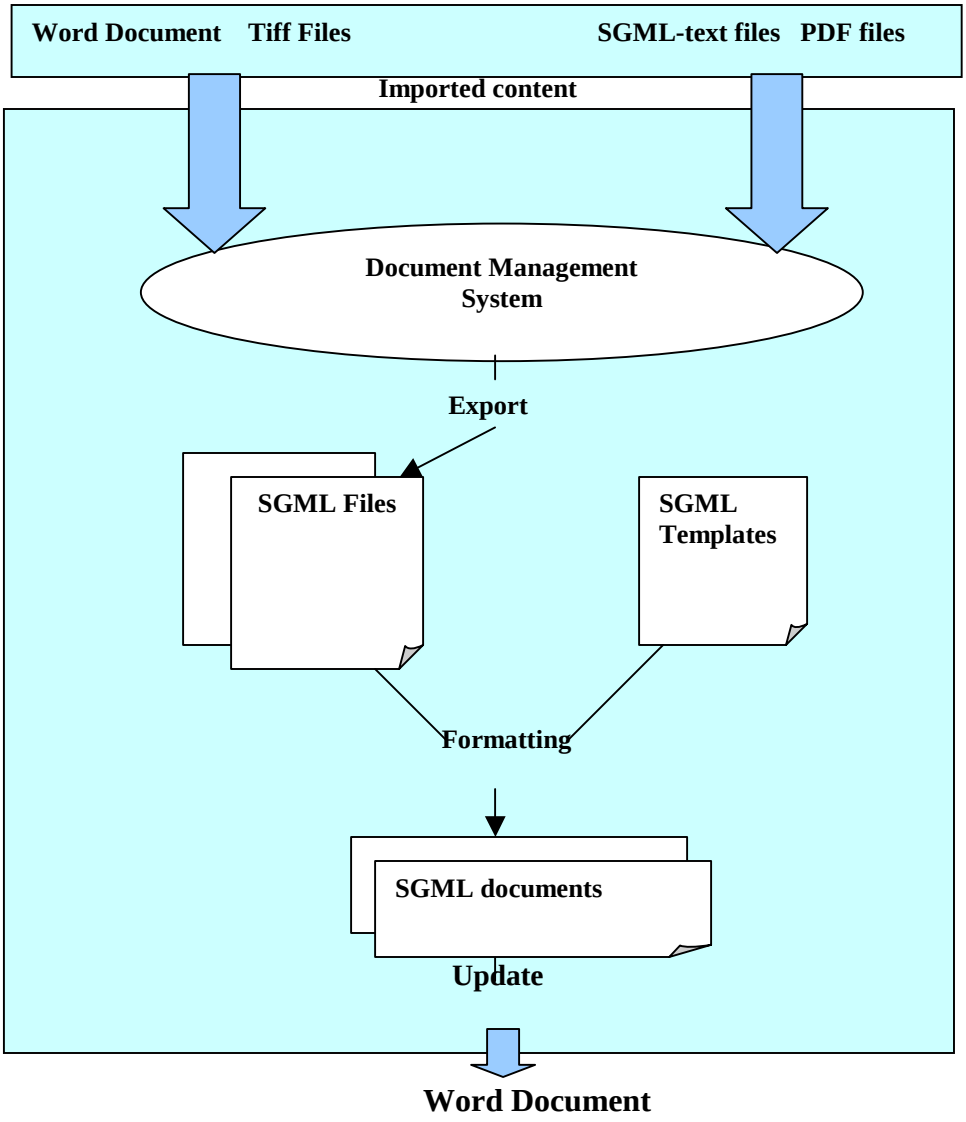


Fig.3 The functional diagram of the minor publications of the treaty section.

This publication process follows the following steps:

Export are developed in DocBasic.
Omnimark programs and a program in C language provide functions for **Formatting**.
Edit involves a batch file to launch WordPerfect.
Update is done by WordPerfect macros. Each DTD has its own macro.
For more information, refer to A1, A2.

2.1.4 Presentation

The information is presented in HTML and XML format for Internet publication and as SGML documents that are compiled and reorganized by an Authoring system for the paper publication.

Chapter 3

Publication System Technologies

The Treaty Section's publication processes are using a Document Management System (DMS), which is Documentum 4i, for managing content while the information is presented in the HTML, XML and SGML formats. Therefore, it seems reasonable to give first the theory behind those technologies before going into the working details of the publication system itself.

3.1 Theory about the involved technologies

3.1.1 Document Management System

3.1.1.1 Document Management History

A few years ago, information was stored and managed locally on individual desktops. This situation led to the creation of islands of information where information was unorganized, uncontrolled and duplicated in many locations. As a result, time was wasted and profits were lost. For those reasons and others, managing the content became a priority for many companies.
There are many forms of Content Management (refer to 3.1.1.5) and a Document Management System is one of them.

3.1.1.2 Document Management System definition

The DMS performs document imaging functions and extends the management to electronic files like word processing files, spreadsheets and reports. Some systems also allow library-like functions to manage where documents are, who has them, and the versions of the documents. Advanced full text search is usually available in DM offerings. Built-in workflow and compound document support are available with some systems.

3.1.1.3 Employment of Document Management System

More and more businesses are coming to realize that having tools, which manage the overall security and general access of documents, is not unique to the legal industry alone. Document management capabilities such as security, version control, check-in/check-out management, workflow, full-indexed text searching of all documents or selected subsets and specific case concurrency represent some of the major requirements expected in any DMS [3].

3.1.1.4 Document Management capabilities

Security

The DMS provides better security than a file system. With a file system, if someone has access to the hard drive, he can pretty much read, edit, and even delete any file on the hard drive. With the DMS, the owner can decide which individual user (or group users) can view, edit or delete the file.

In fact, the DMS controls who:

- Owns the content
- Can read the content
- Changes the content

Check-in/ Check-out management

If the user wants to make a change in the content, he has to check it first out for editing. The check out of content allows:

- The user to change or edit the content
- The placement of a lock on the content to ensure that only one person at a time may check out the content for editing.
- Other users (regarding their permission) to view the last version of the content that was checked in even if the content is currently checked out.

To let the other users see the changes, the owner of the content has to "check it back in" to release the lock and save the changes. By checking in the content, it is saved as a new version.

Version Control

When the content is checked in, the user can create a new version so that he can differentiate the new version from the old one(s). With this possibility, the user can go in time and access any old version. The version control system stores multiple versions of a source file/content, as well as record history information about the file such as:

- Creation date
- Who created it
- The version number or label
- A short description of the executed changes

Search Engine

When the user creates content, he can assign properties (also called meta-data or attributes) to that content, e.g. the treaty title, the participants, the date of conclusion, the place of conclusion, and any keyword to describe a particular content. The user can even specify the kind of the content e.g. a letter, a treaty, subsequent treaty, action, etc. The system adds to every created content specific attributes (also called *system attributes*)

such as the creation date, the modification date, the owner, etc. These properties are used to ease the search capabilities for contents.

Virtual Document

A virtual document (VD) is a document that contains components or child documents. A virtual document is similar to a folder. The main difference between a VD and a folder is that the first one has content and can be versioned while the second one does not.

The VD is useful in order to manage:

- The integration of multiple files formats e.g. PDF files, Word Documents, Tiff files (images), etc.
- The ordering of component files: the files appear in the order defined by the creator of the VD

Workflow

First, I have to mention that this characteristic is missing in the Treaty section's DMS. A workflow is a connected network of activities. It defines **who** performs what and **when.** A workflow helps a decentralization of the content creation.

3.1.1.5 Content Management Forms

DMS is not the only content management (CM) form that exists. There are many other forms for CM that are presented in *table 1*, in order to provide the interested readers with a brief overview.

Management Technology	***Primary Purpose***
Document Imaging	*Manage hardcopy paper by scanning into a DI system, allowing multiple user access by searching on predefined indexes.*
Content Management	*Extends document management to be more focused on a wider range of content. Some CM systems offer management and delivery of newer content streams as digital media, audio, streaming video.*
Web Content Management	*CM for the web. This technology is focused on helping organizations manage for websites. Depending on the system, web content management solutions can offer tools and processes to create, manage, personalize, and distribute (post) content (or information) to websites, PDA, cell phones, pagers, and other information delivery devices.*
Enterprise Content Management	*CM nirvana. Less a product, more of a complete end-to-end strategy for managing all of an organizations' content, no matter what it is and where it used. Ex: CM and web content management combined.*

Table 1: The different forms of Content Management [6]

After this brief summary of the functionality of a DMS, let's move to the other technologies involved in the Treaty Sections production system which are SGML and XML.

3.1.2 SGML

Standard Generalized Markup Language (SGML) is an international standard for the definition of device-independent, system-independent methods of representing texts in electronic form [4]. SGML is used for the description of marked-up electronic text. More exactly, SGML is a metalanguage, that is, a means of formally describing a language, in this case, a markup[5] language.

3.1.2.1 Characteristics of SGML

There are three main characteristics of SGML which distinguish it from other markup languages:

- Its emphasis on the structure, rather than the presentation of a document
- Its Document Type concept
- Its platform independence

These three aspects are discussed briefly below.

3.1.2.2 Descriptive Markup

A descriptive markup system uses markup codes (attributes or metadata), which simply provide names to categorize and describe parts of a document.

```
<SgmlDocument>
<Registration.year>1998</Registration.year>
<DocumentType>[Translation – Traduction]</DocumentType>
…..
</SgmlDocument>
```

(Extract from a SGML document)

From the encoded text above, even an inexperienced user can deduce that this document is an SGML document, the text is a translation (not an original text) and that the registration year of this document is 1998.

3.1.2.3 Document Type concept

[5] *Markup is defined as any means of making the interpretation of a text explicit, e.g. it instructs how a particular text fragment should be printed or formatted.*

The SGML is accompanied by the notion of Document Type, and hence a Document Type Definition (DTD). The type of a document is formally defined by its constituent parts and their structure. The definition of a treaty, for example, might be that it consists of a registration year, document type, title, followed by a sequence of one or more paragraphs. Any document lacking a title, according to this formal definition, would not formally be a treaty.

If documents are of known types, a special purpose program (called a *parser*) can be used to process a document claiming to be of a particular type and check that all the elements required for that document type are indeed present and correctly ordered. More significantly, different documents of the same type can be processed in a uniform way.

Rules, that specify the structure of a SGML document, are called Document Type Definition, usually abbreviated to DTD. Therefore, the DTD is a kind of an interpretation and validation tool for a document type.
A Treaty Series book or volume (element **TreatySeries** in SGML) may contain several sections as described below:

```
<!-- Structure of the TreatySeries -->
<!ELEMENT TreatySeries - - (CoverPage?, FirstPage?, TOC.eng?, TOC.fr?,
                            (PartTitle*,(TreatyHeader,(ImageDocument|Sgmldocument|TextDocument)*)+)*,
                            (AnnexTitle,AnnexEntry+)*)
                          +(topofpage,label,italic,footnote, footer,newline,registration.year, DeclarationHeader)>
<!ATTLIST TreatySeries Volume NUMBER #REQUIRED >
…….

<!ELEMENT (TOC.fr|TOC.eng) - - ((TOCPartTitle,TOCTreatyHeader+)*,
                                 (TOCAnnexTitle,TOCAnnexEntry+)*)
                               +(TOC.reference.format, Registration.year)>
<!ELEMENT TOCPartTitle       - O (TOCPartDates, TOCPartNumbers)>
<!ATTLIST TOCPartTitle       Number (I|II) #REQUIRED>
…….

<!ELEMENT TreatyHeader  - - (TreatyNumber,EngTreatyHeader,FrTreatyHeader,EIFTable*)
                           +(Topofpage, Label, Italic, Footnote, Footer, Samefootnotenumber, Newline,
                             Registration.year, Sscript)>
<!ATTLIST TreatyHeader     Multieif (T|F) F>

<!ELEMENT (EngTreatyHeader|FrTreatyHeader) - - (Participants,TreatyTitle,Attachments?,Conclusion?,
                            EIF,Languages,AttachLanguages*,Registration,Note?,LimitedPublication?)>
…….

<!ELEMENT AnnexTitle - O (EngAnnexNumber, FrAnnexNumber)
                      +(Topofpage, Label, Italic, Footnote, Footer, Samefootnotenumber, Registration.year)>
<!ATTLIST AnnexTitle Number (A|B|C) #REQUIRED>
……

<!ELEMENT Annexentry - -  (Annexentryheader, Separator?, ((Subentryheader | (Subagreementheader,
                            (Eiftable, Eiftable)?) | Actionheader | Subactionheader |
                            Declarationheader), (Imagedocument | Sgmldocument | Textdocument)*)+)
                          +(Topofpage, Label, Italic, Footnote, Footer, Samefootnotenumber, Registration.year) >
<!ATTLIST Annexentry       Multieif  (T|F)     F >
….

<!ELEMENT Languages        - - (#PCDATA)>
<!ELEMENT ActionParticipant - - (#PCDATA)>


……….
```

(An extract of the SGML DTD used for the Treaty Series book)

For more details about DTDs, refer to [4]

3.1.2.4 Data Independence

A fundamental goal of SGML was data independence. In other words, the structured documents should be transportable between different platforms with different software and hardware environments.

The two features of SGML discussed above, provide also a kind of data independency but at an abstract level (the documents are descriptive, and interpretable where ever they are), but here we are concerned with data-independency at the character level (byte level) of the structured document.

SGML provides a general-purpose mechanism for string substitution which is machine independent stating that a particular string should be substituted by another one when the SGML document is processed. The strings defined by this string-substitution mechanism are called *entities.*

One obvious application of this mechanism is to ensure consistency of nomenclature; another, more significant one, is to counter the notorious inability of different computer systems to understand each other's character sets, or of any one system to provide all the graphic characters needed for a particular application, by providing descriptive mappings for non-portable characters.

```
<!ENTITY ID8017f779 SYSTEM ".\Documents\Treaty-35808-partI\sgml-
25\file001.sgm">
<!ENTITY ID8017f77c SYSTEM ".\Documents\Treaty-35808-partI\sgml-
30\file001.sgm">
<!ENTITY ID8017f80c SYSTEM ".\Documents\Treaty-35809-partI\sgml-
25\file001.sgm">
<!ENTITY ID800dc113001 SYSTEM ".\Documents\Treaty-35809-partI\tiff-
64\file001.tif" NDATA tiff >
```

(Example of entities)

3.1.2.5 Using SGML

A variety of software is available to assist in the tasks of validating and processing SGML documents. *FrameMaker+SGML* is one of them used in the Treaty Section's publication system.

FrameMaker+SGML, is:

- An SGML *parser* that validates the SGML document according to its DTD.
- A *structured editor*, which is a kind of intelligent word-processor. It can use information extracted from a processed DTD to prompt the user with

information about which elements are required at different points in a document as the document is being created (Appendix B1)
- A *formatter* that operates on a tagged document instance to produce a printed form of it

For more information about FrameMaker+SGML refer to [9].

3.1.3 XML

XML is a specification for storing and exchanging data that the World Wide Web Consortium (W3C) created in 1996 to standardize information delivery across the Internet. The W3C defines XML as a subset of SGML. XML is not a new language; instead, you can think of it as a language specification [10].

XML is similar to HTML. XML uses tags and attributes to define data in the same way that HTML uses tags to define formatting. However, instead of having a fixed set of tags, as HTML has, XML lets the creator define the tags its XML streams use. Therefore, you can use tags to make content self-evident.

XML is a technology for marking up structured data so that any software with an XML parser can understand and use its content. Data independence, the separation of content from its presentation, is the essential characteristic of XML. XML documents are simply text files that are marked up in a special way, so XML is intelligible to both humans and machines. Any application can conceivably process XML data. That is why XML is ideal for data exchange.

Many software developers across the world are integrating XML into their applications to benefit from the various advantages of this language, which are:

- **Simplicity**: XML is simple because the author and provider can design their own document type using XML (vs. HTML)

```
<Treaty Language=”English”>
    <TreatyNumber>1234</TreatyNumber>
    <Participants>
        <Participant>The Netherlands</ Participant>
        <Participant>France</ Participant>
    </Participants>
    <TreatyTitle>Agreement for a co-operative housting program</TreatyTitle>
</Treaty>
```

(Example of XML presentation)

- **Intelligence**: XML is smart data. While the hypertext markup language (HTML) shows how the data should look, XML tells the user what the data means.
- **Simplified subset of SGML**: XML removes many of the underlying complexities of SGML in favor of a more flexible model; so writing programs to handle XML will be much easier than doing the same for full SGML.
- **Accessible and reusable information**: XML eliminates the chaos associated with HTML while allowing better management and reuse of structured documents. XML allows this by enabling Web teams to separate Web content held in a data repository from business logic (the rules that govern how content is delivered) and presentation (the layout templates that determine the graphical look of the Web site). By separating the definition of content from the way it is presented, XML makes it easier to use the content in multiple ways and for multiple presentation formats. The separation of content, presentation and logic also facilitates the content-delivery process. Web pages can now be constructed based on the visitor's actions or requests. When a request is made by the user, data that the user is permitted to see is pulled from the content repository based on the defined business rules *(fig 10)*. This type of on-the-fly page generation lowers the costs of re-purposing information and updating content [8].
- **Improvement of performance through granular updates:** XML enables granular updating. Developers do not have to send the entire structured data set each time there is a change. With granular updating, only the changed element must be sent from the server to the client. The changed data can be presented without the need to refresh the entire page or table.
- **Independence**: The tags in XML documents are not predefined. Any descriptive string can be used (as long as it uses the approved character set). Moreover, the data defined by these tags can be displayed in any way you like and on any platform.
- **Adaptation**: XML' s adaptation is infinite.
- **Maintenance**: When the DTD is provided, XML data become easy to maintain

Now that I gave the theory behind the most important technologies involved in the publication process let us discover how this latter behaves.

3.2 Treaty Section's Publication System in Details

The most important component of the publication process is the DMS. Documentum 4i is the Treaty Section 's DMS. How Documentum works, and how it is integrated with the other components of the publication process is given in sections below.

3.2.1 Documentum

Documentum 4i is a Document Management Software that provides a vault to store and control common content formats, with tight integration with the Microsoft Office suite,

word processing files types and such archival formats as Adobe's PDF, and SGML files. Documentum 4i provides the following features of the DMS:

- Security
- Version control
- Search Engine
- Virtual Documents

3.2.2 Documentum Input

When the owner wants to create content, he has just to save and store it in a central location. This central location is a repository in the server called Docbase (fig 4).

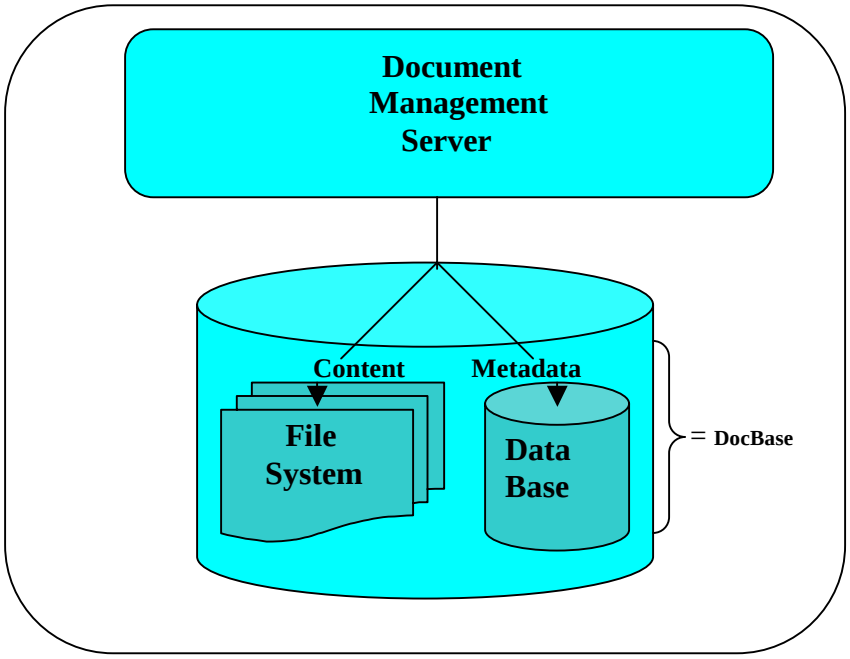


Fig 4: Document Management Server and the Docbase.

- The Docbase stores an "object's" content and any information, metadata, associated with this content.
- The content is stored in the file system of the server, but the corresponding metadata is stored in the database (Oracle 8i).
- The metadata facilitates the search for a particular content.
- The content can be changed, viewed, searched during its life cycle inside the Docbase.

3.2.3 Documentum Output

Documentum manages different content formats and in order to standardize its output, both from a structural standpoint and for a common look to its printed output, it makes use of SGML. This output can consist of SGML documents with relevant metadata, the content itself and log files containing additional information about each extracted content and its corresponding SGML document.
The authoring system, FrameMaker+SGML, compiles and edits this output before its publication in paper format or on the Internet.

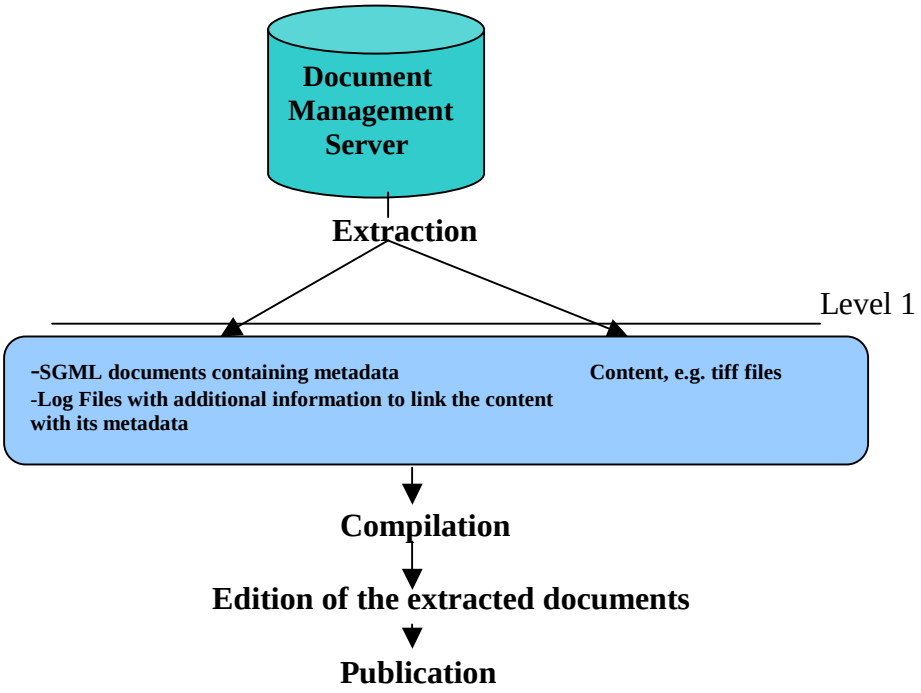


Figure 5:The Output of the DMS and its manipulation in the current System in case of Treaty Series books.

Figure 5 describes how the products are extracted from Documentum and how they are manipulated. This figure also clearly illustrates the problem mentioned earlier related to the absence of content control from end-to-end. On the one hand, the changes that occur after *level 1* cannot be reported back to the DMS and on the other hand, if changes occur in the DMS, there will be an inconsistency between what the DMS contains and what was extracted and published. An alternative model for the actual publication system must be found.

Chapter 4

An Alternative Model for the actual publication system

The treaty section is already making use of a DMS but the problem is that its DMS does not provide a full control of the publication process from the input to the final release because the output of Documentum is further manipulated by other systems before publication of a document that cannot be referred back to the DMS.

To solve this problem the Document Management Industry has developed a more comprehensive approach termed **Enterprise Content Management.**

4.1 What is Enterprise Content Management?

An Enterprise Content Management (ECM) provides a complete end-to-end strategy for managing all of an organization's content, no matter what it is an where it used [6].

An ECM ensures that all content to be published is sourced from appropriately managed repositories. This capability minimizes unnecessary duplication, provides sound version control and ensures that all information released to the public is authorized and current.

4.2 Description of Enterprise Content Management System

A description of Enterprise Content Management can begin with a look at what these words mean in isolation and in concert as described in [7].

"**Enterprise**" connotes a scope that is comprehensive, representing an entire organization's needs for the targeted services. The focus is on "core" services, that is, services at the heart of operations, services that departments have in common, regardless of differences between departments. An "Enterprise" system is one designed to gain the maximum benefit that can be derived from the advantages and economies of scale - in this case, scaled to extend across the entire enterprise.

"**Content**" refers to any work-product that can be created, modified, stored, or retrieved employing an organization's digital infrastructure. Content includes publicly accessible material published in an organization's web structure, plus all privately-held material not published in the web structure (but which nevertheless are assets of enormous value). Thus content refers not merely to the content of web pages, but extends to include all forms of digital objects; documents, designs, templates, data structures, data values, graphics, audio and video files, curricular material, etc. These various types of content, while dissimilar in terms of means of production, targeted mode of output, original purpose, etc, nevertheless share characteristics that enable them to come under the control of a unifying mechanism. They can be digitally stored. They can be transported over a common network infrastructure. They can be described in ways that permit search and retrieval across boundaries of content-type, medium or department of origin.

"**Management**" refers to the directed and purposeful employment of an organization's resources; the smooth and efficient handling of tasks related to the creation, transformation, storage, preservation and retrieval of the component elements that contribute to, and result from, an organization's efforts. The ECM approach is inherently robust, for it combines distributed storage with a centrally managed framework. The ECM approach is inherently scalable; a result of its innate capability to incorporate additional repositories (and new content types) into its organizational framework.

Taken together, the term Enterprise Content Management connotes a unified framework for managing, web-enabling and personalizing delivery of all disparate forms of content across the enterprise, regardless of their classical modes of creation, storage or presentation. ECM incorporates *and implements* the business rules that govern the processes needed to create, acquire, store, index, secure, search, export and transform these assets. Standards at the heart of the descriptive function of the ECM system enable federated searching, in which a single query can be directed toward multiple content repositories holding many different types of digital content. This reliance upon standards paves the way for a system that can grow in value over time as it brings an ever-increasing volume and variety of The Treaty Section resources "out of the silos and on to the grid." The transformation to the organization made possible by this is genuinely integrative in nature.

4.3 Advantages Of Having An Enterprise Content Management

Having an ECM system would provide [2]:

Content Life Cycle Management:

- Ability to manage content objects through their entire life cycle (creation, modification, archive and deletion)
- Requires library services as well as hierarchical storage management capabilities, backup capabilities and records management functionality (such as the ability to define and execute retention and disposition schedules)

Management of electronic objects:

- The ability to capture, control, and deliver content objects in a wide variety of formats
- Includes traditional object types (images, office documents and print streams)
- Also includes newer object types like e-mail, digital assets (video), etc.
- Requires a repository that provides traditional library services (including content profiling, check-in, check-out, version control, revision history, security, etc.)

Web Content Management:

- Web resources configuration and administration

- Web authoring and publishing using web-centric workflows
- Ability to deliver personalized content to users (based on user preferences, profiles or other data)
- Ability to separate business logic from content, with templates that handle the reformatting of content and dynamic presentation of content

Process Management:

- Ability to automate and manage business process and workflows
- Includes the ability to support ad hoc or collaborative approval workflows as well as higher-volume production workflows. This usually relates to documents and revisions, prior to web publishing

Integration:

- Ability to integrate with external content stores, line-of-business systems and user desktop environments
- Can be accomplished via object-level interfaces (such as EJBs), packaged connectors, APIs or enterprise application integration (EAI) technology

As an integrated capability, an ECM system provides a total information management solution for medium and large organizations. Thus, for the Treaty section the ECM will be the ideal solution.

4.4 Enterprise Content Management and the Treaty Section

An *Enterprise Content Management system* will play an important role in the future of medium and large organizations in the post -Millennial Web environment. However, it is not possible for the Treaty Section to take a proactive role in creating an environment in which these developments can mature. The reason is that the adoption of this system costs million of dollars, an amount that the Treaty Section cannot pay, at least in the next few years. In addition, a complete migration process from the actual situation to a totally new system will generate new problems related to user training and data migration which will take time and increases the total cost.
There is also a possibility to use a much known Enterprise content management system, Documentum, which is an extension of the actual document management system used by the Treaty Section. This option is advantageous in order to reduce the amount of time needed for data migration and personal training but this solution remains a very expensive one for the Treaty Section. Note that this option has not to be forgotten when the Treaty Section will have enough funds to adopt an Enterprise Content Management System. However, at this stage a long-term solution with a reasonable budget and which can solve all the Treaty Section’s problems has to be found.

Chapter 5

Proposed Model

The objective of this chapter is to define a reengineering model that could eliminate the restrictions of the present publication system and which will cost much less than to pursue a totally new ECM system.

5.1 Questions

Before starting to think about a solution to the Treaty Section's problems, I asked myself two questions about ***what I want*** and ***what I have.***

5.1.1 What I Want

What I want is clear. First and for most, I want full control of content during its life cycle, from its creation until its deletion (problem illustrated in *fig 10, Section 5.2*). I also want to implement measures that will *overcome the limitations of the actual publication system* and which will result in high quality-today and in the future.

5.1.2 What I Have

Because I have to find a solution based on *what I have* in order to limit the costs of the possible solution, I have to find which component of the present publication system I can keep and reuse in the remodeling process. From the analysis phase, we found that in the publication system, many applications and components are involved in publishing the Treaty Section's products. The most important of these components is the DMS. It is partly an *open-source* product which can be improved and reintegrated with other *new* applications to achieve good results.

In sections before *(See Chapter 3)*, we already sketched how the DMS of the Treaty Section works. What we still need to understand is its output in order to determine how this will be reused.

5.2 Description of the DMS's output

The output of the DMS *(see Fig. 5 and Fig. 6)* consists of SGML Documents with relevant metadata about each treaty (*see Fig. 7 and Appendix C1*).
The content itself also is extracted, when necessary, from the DMS, as SGML files, word documents or images *(see Fig. 7)*.
The output of the DMS is then compiled in the Authoring System, producing the desired document with the pre-defined SGML structure. After compilation, this document can be published in paper format or converted to HTML for publication on the Internet.
When necessary, with the extracted SGML documents we also get "log file(s)", which contain additional information used by the Authoring System to compile and edit the extracted products before their publication.

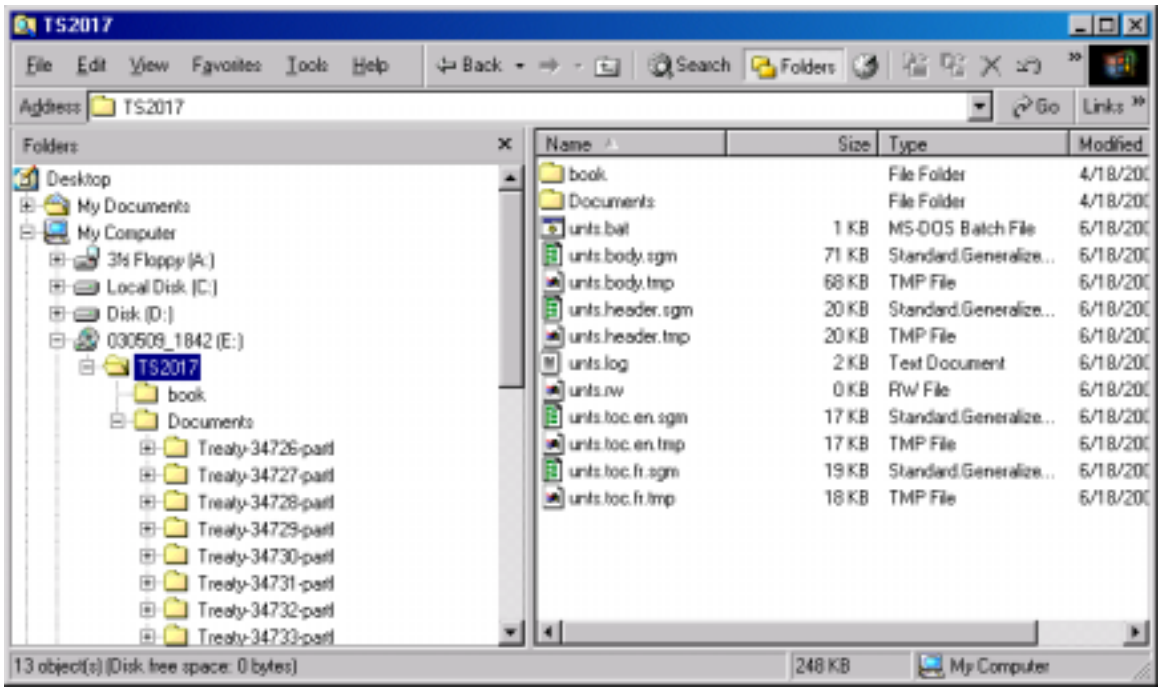


Fig 6:The Treaty Series book (The "TS 2017" Folder) that was generated by the DMS, where you can see, e.g. the produced Sgml and log files at the right side.

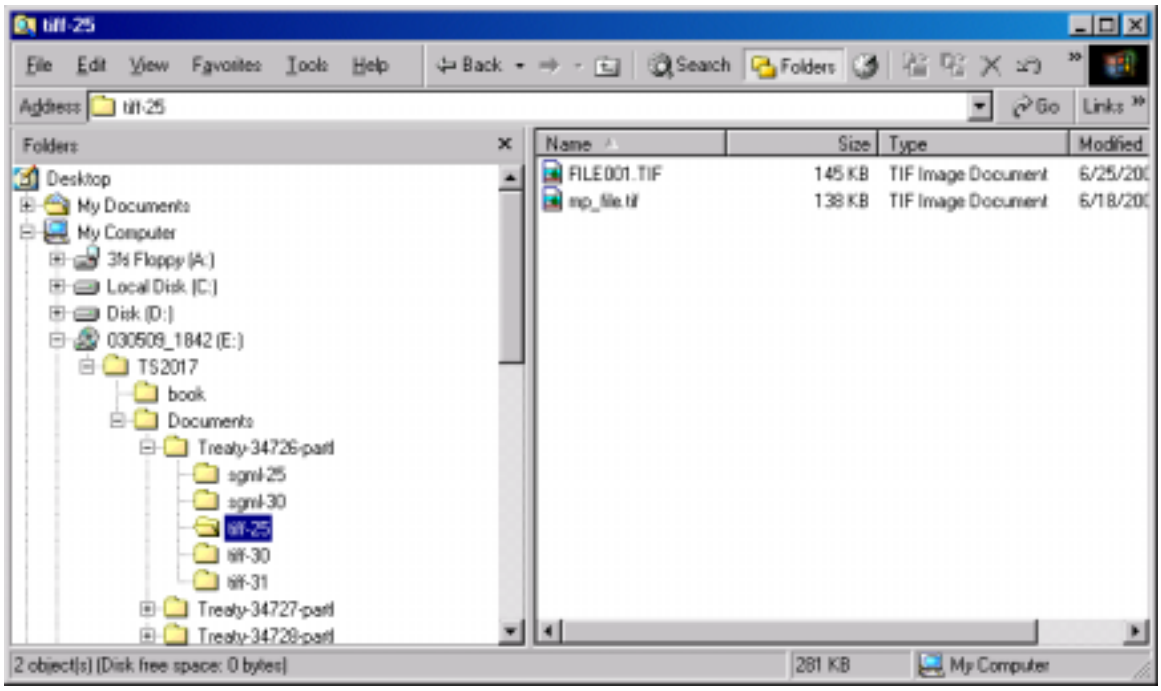


Fig 7: The extracted content from the DMS, e.g. Images (tiff files).

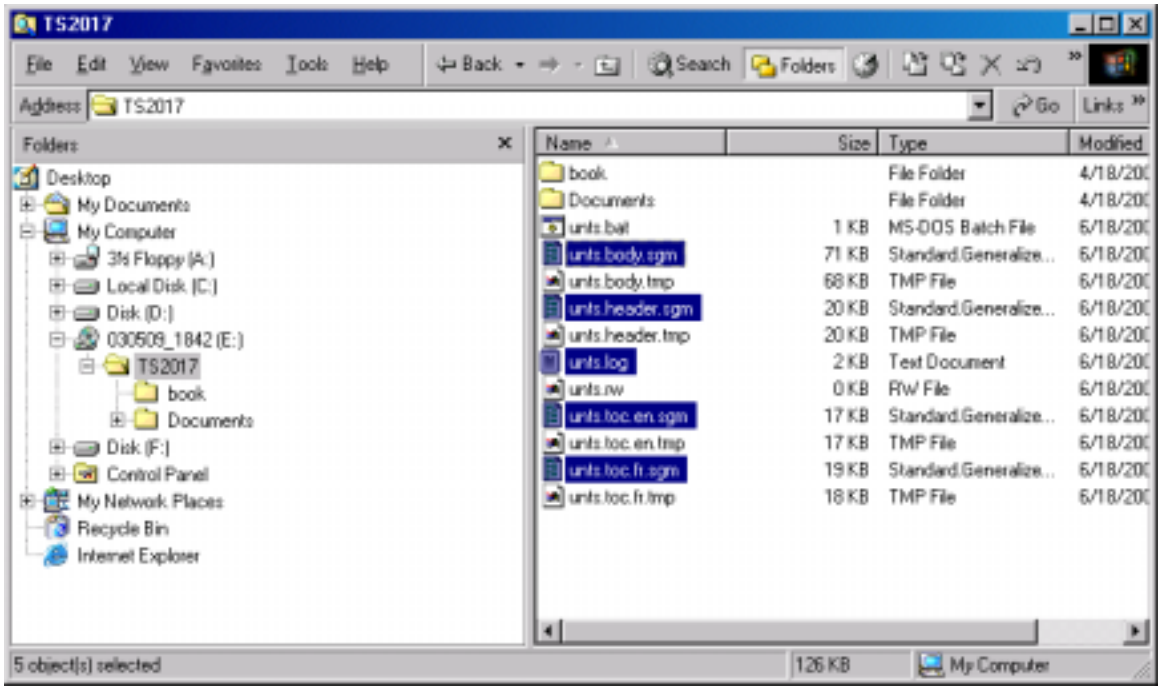


Fig 8:The extracted metadata comes out in SGML files and other relevant information that describes the extracted product is placed in the log file.

In the next sections, I will present my solution that will facilitate the publication processes and permit full content control. This solution is composed of an Enterprise Application that I called EPBOX and some modifications on the DMS and the authoring system.

5.3 Design of the 'EPBOX'

The model's name will be: 'EPBOX', which means: **E**xtraction and **P**ublication Box. EPBOX will take the DMS's output as input and its rules will be the following:

- **E**xtraction of the metadata from the SGML files
- Storage of the metadata in the application's, EPBOX, database
- Extraction and conversion of the content to PDF
- Storage of the converted content in the database
- Extraction of additional information from the log file, which was created for this purpose (*refer to Section 5.2.1*), and store its content properly in the database
- Finally, **P**ublication of the products in the appropriate way

5.3.1 Model Architecture

The EPBOX application will have a three-tier architecture, as shown in *Fig 9.*
The information tier consists of a relational database that will contain the extracted metadata, the contents and any additional information if necessary.

The middle tier consists of the server container that must be able to extract and store the metadata in the database. It must also convert the content to PDF before its storage in the information tier. The server also will handle client requests using the database. It selects the required information from the database then, when necessary, marks up this information as an XML document. The server applies then the Stylesheet Language Transformation (XSLT) transformation to the XML document and sends the resulting content to the client.
The client tier consists of client types such as Web Browser (WB) and PDA (**P**ersonal **D**igital **A**ssistant). Clients can interact with the server by sending requests and receiving responses.

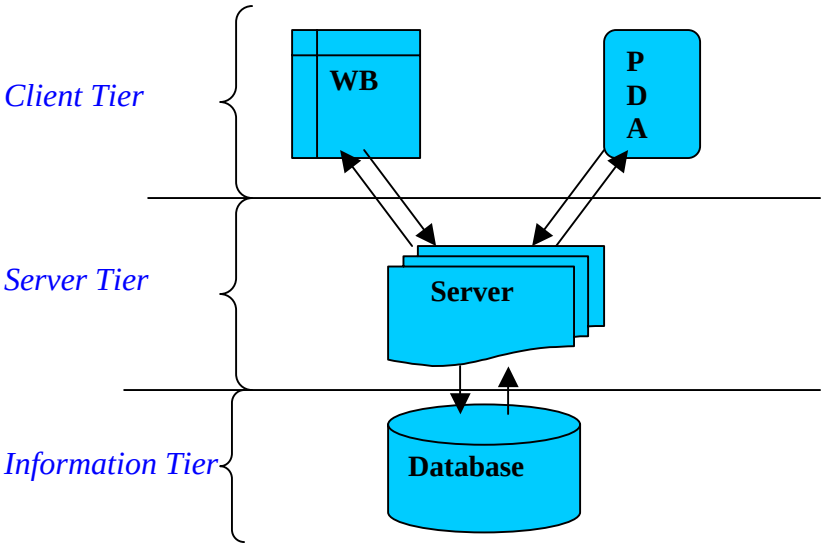


Fig 9: Three-tier architecture for EPBOX

Now let us discover how *EPBOX* will be integrated with the reused components of the publication system *(Fig 10).*

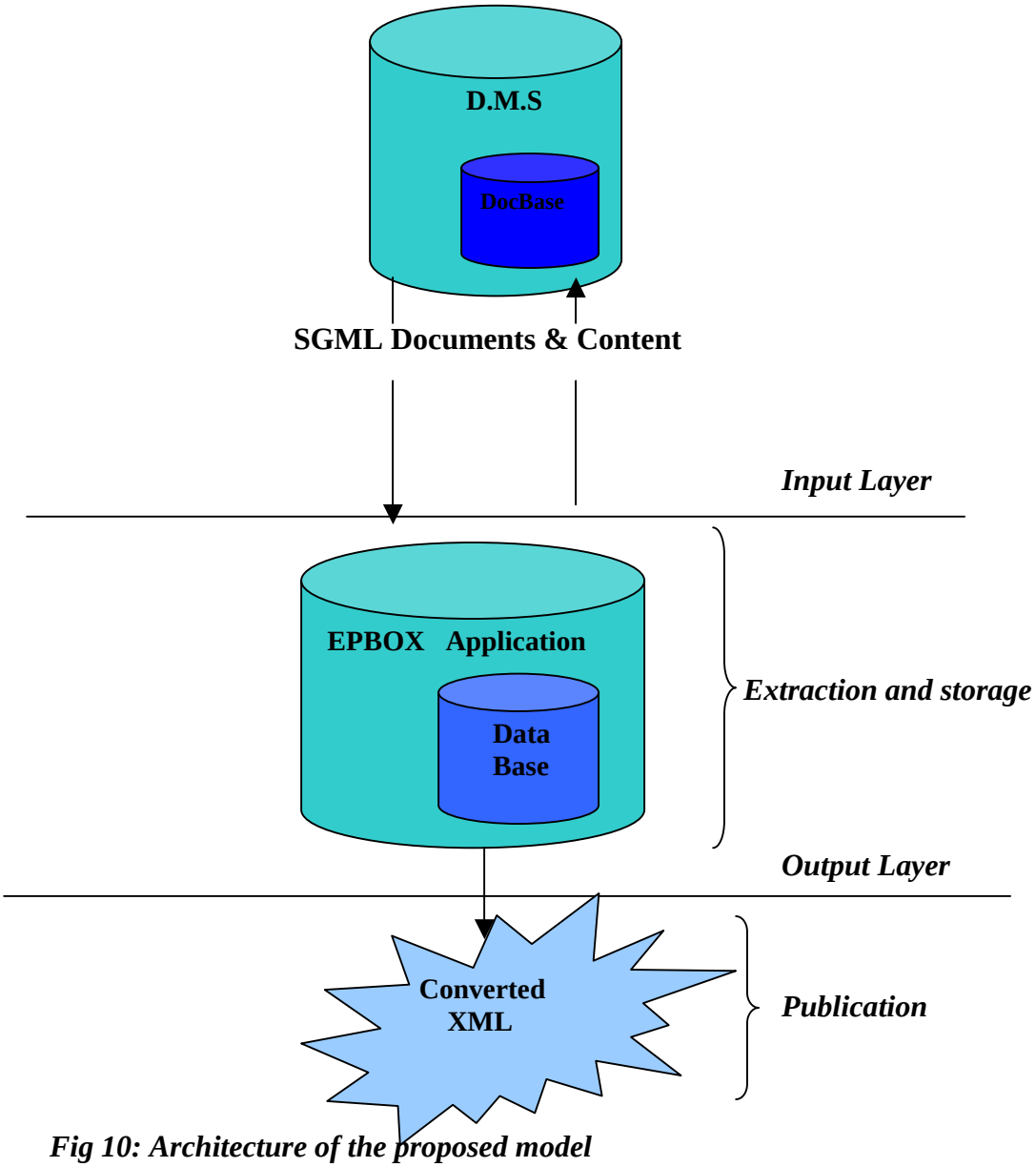


Fig 10: Architecture of the proposed model

In this proposed solution, the EPBOX is going to use the DMS's output. It has to extract the metadata from the produced SGML files that contain the Treaty attributes and then store it in its database. When necessary, the information contained in the report/log files is also extracted and stored in this database in order to link the treaty attributes to the Document Content, which in turn has to be *converted with relevant metadata* to PDF and then stored in the database.
In Sections 5.4 and 5.5, I will discuss in detail how the products will be published both on the Internet and in paper format once the EPBOX is integrated.

5.4 Publication on the Internet

Once the content *and/or* metadata are extracted with the EPBOX and stored in its local database, they can be made accessible to users all over the world.
When users send a request, the EPBOX will execute a program that in turn runs a query on the database and generates, when necessary, an XML document based on the query result.

Once the XML documents are generated, they have to be transformed in order to hand them over to the clients. In my model solution, I opted for Extensible Stylesheet Language Transformation (XSLT) to handle and format the XML documents.

As you can deduce from the solution architecture *(Fig 10)*, data is kept separated from its presentation and the role of the EPBOX application will be the assembly of the data before its presentation to the user. With this solution, the web pages can be produced *on-the-fly*, tailored to the clients based on visitor's actions, requests or client's type (refer to Fig 11).

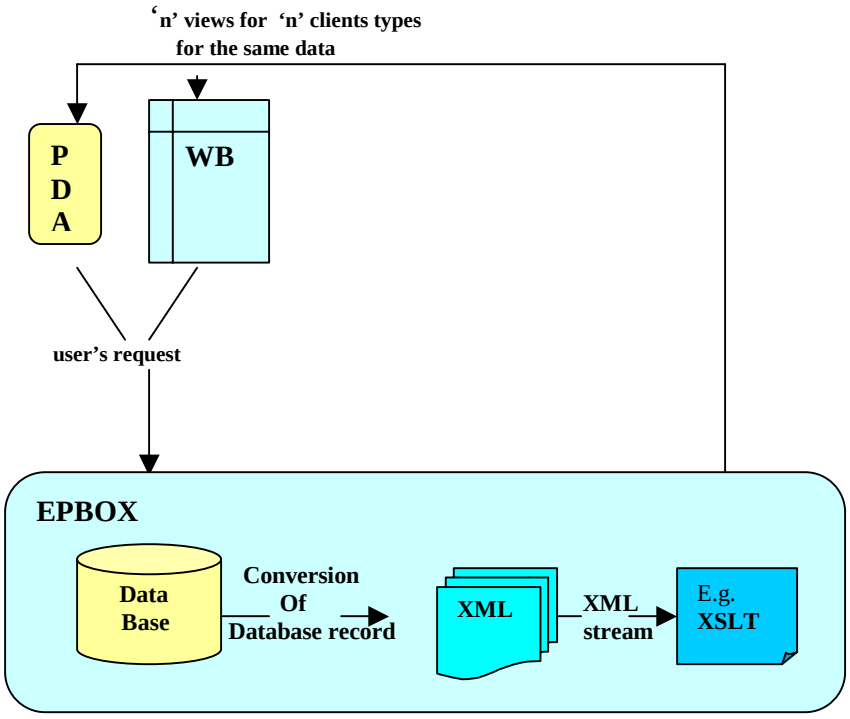


Fig 11: From the Database to the client

Now that the way of product's publications on the Internet is known, we can move to the next step which is the publication of the Treaty Section products in paper format.

5.5 Presentation in Paper Format

The Treaty Section publishes the text of treaties and international agreements in printed form too. As I mentioned before in *Section 2.1.3*, the publication of the

different products follows *three distinct publication processes*. During these publication processes, all required attributes *and/or* their corresponding content are extracted from the DMS and then manipulated by different applications before their release.

The most important publication processes, however, are those of e.g. *the Treaty Series Books* and *the "Bible"*. That is why I will focus only on these two products in this Section to explain my approach.

5.5.1 Treaty Series books

The publication of the Treaty Series books will remain almost the same.
First of all, the Treaty Series books are generated from the DMS. Then, the authoring system has to compile and format the DMS's output, using the predefined DTD's like they are used in the current publication system. Except now the authoring system has to be *extra-customized* in order to notify and report any modification made during the editing process to the DMS. This solution is necessary in order to avoid the inconsistency problem mentioned in *Section 5.2 Fig 5*.

This solution can be generalized to the other products produced in the same way as the Treaty Series books (e.g. Cumulative index)

5.5.2 Publication of the "Bible"

In *Section 2.1.3 (screen 3),* it was said that during the editing process of the "Bible", some information has to be added manually after its extraction from the DMS. This situation leads to inconsistency between what is published and what resides in the DMS. Add this to the fact that the author of the "Bible" has to add this information *manually*, which is inconvenient and time consuming. This situation makes the publication process of the Bible different and more difficult than the publication of the other products of the Treaty Section. That is why a separate solution in this case has to be found.

5.5.3 Solution for the "Bible" in details

The ideal solution to the "Bible's" problem is to allow the creators to add all the desired information during the creation stage at the DMS level and that is why the way of import of the data to the DMS has to be modified. To make this situation possible, the actual user interface of the DMS has to be modified to allow the input of the additional information; we cannot forget that additional functionalities have to be added at the DMS in order to extract this new information in the right way at the right place.
This solution will not be very difficult to realize it is a conclusion that I come to after deep study of the structure of the "Bible" and the source-codes provided by the DMS.

When the modifications mentioned above are done, the Bible can then be published in the way described in *Section 5.5.1*.

5.6 Data Consistency

One important point that we should not forget is that data consistency between what resides in the DMS and what is published has to be achieved. The priority of the new model is to control the content from creation until deletion. For this reason reports files have to be generated in order to notify each component involved in the publication process with any modification on documents during their lifecycle (*Fig 12*).

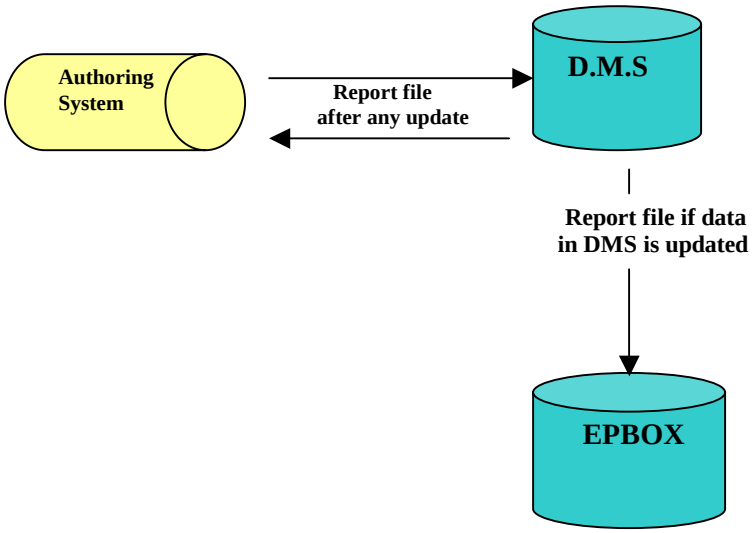


Fig 12: How events can be handed over to realize consistency.

When changes occur at the DMS site, the DMS then has to report this changes to EPBOX's database and the authoring system.
The changes of the documents during the editing process at the Authoring system level have to be transferred to the DMS, which in turn will transmit those changes to the EPBOX database.
However, based on my study of the system I remarked that the most significant modifications come from the DMS's side, where treaties and related information are modified almost every day.
Before closing this Section, I should note that reporting changes using log or reports files is not a hazardous choice. It is not possible to report the changes to the database and the file system of the DMS directly, because the only way to access it is through the DMS - which is very difficult to accomplish.

Chapter 6

Evaluation

It was chosen to use XML and XSLT for presenting data on the Internet. In this chapter, I will attempt to justify and evaluate each of these choices, before going into more implementation details in *chapter 7*.
I will begin with the choice of converting the content to PDF before its storage in EPBOX's database.

6.1 Why does the content have to be converted to PDF?

The actual publication system does not provide full portability of the extracted content when it is published.
Without the metadata, the extracted content does not mean anything. For example, extracted content does not contain information about its author, its creation date or even its ID.

To provide the content and/or the published documents (especially on the web) with the portability quality, it is necessary that the documents themselves contain meta-information. When each individual document contains its own text as well as its own fields, it becomes a fully ***self-explanatory and self-contained unit*** [5].

A format that operates on the web and supports mixed metadata, full text content, and images is *PDF*.

Converting content to PDF format provides many other advantages:

- PDF format provides tighter integration between text and images than HTML.
- HTML may not look the same in different browsers whereas Adobe Reader always gives the same precise look and feel.
- PDF is a lockable format, which make it more adaptable when file security issues are at stake
- PDF is a great format for Optical Character Recognition (OCR)
- PDF has the ability to support multiple searchable Meta fields, which make the document portable and re-usable.

To achieve the conversion process of content to PDF, third party software can be integrated in the EPBOX application. Note that actually there is a great deal of software able to assure this kind of conversion.

6.2 Data Presentation in XML format

XML will be used to present data. The question that the reader may ask is: why XML and not SGML or HTML?

6.2.1 XML vs. SGML

SGML is a standard document-formatting language that enables a publisher to create a single document source you can view, display, or print in a variety of ways. However, SGML is a large and complex formatting language when compared to XML, which itself is a simplified version of SGML, designed specifically to support Web development. Add to that the fact that XML provides 80 percent of the power and functionality of SGML, with only 20 percent of its complexity.

I also have to mention that customers, e.g. the General Assembly, want to access United Nation's systems via a back-end appliance PDA. The formatting of data returned to the PDA is most often done using XML because XML data containers can be positioned in different style sheets to format data for the screen size, memory capacity and processing power of the device requesting it. Therefore, moving to XML in data presentation will greatly profit the Treaty Section. The XML documents could be formatted using Extensible Stylesheet Language Transformation (XSLT).

6.3 What and why XSLT?

XML has been a very hot topic, and now it's XSLT's turn [10]. XML is used to structure the data in documents, and XSLT enables us to work with the contents of XML documents and to format that data in a detailed way. Of course, it is possible to work with the contents of XML documents by writing our own programs that interface to XML parser applications, but that involves writing our own code. With XSLT, on the other hand, it is possible to perform the same kinds of tasks, and there is no programming required. Rather than write our own Java, Visual Basic, or C++ code to handle the contents of XML documents, we can use XSLT style sheets to specify what we want to do, and an XSLT processor does the rest. For more details, refer to [11].

In *chapters 5 and 6*, I presented a solution model and I gave a justification of each choice involved in the model solution. Now, I will go into more details about the implementation itself.

Chapter 7

The Implementation process

The goal of this chapter is to present a way to implement the model solution presented in *Chapter 5*.

To implement EPBOX of the solution model, I opted to use *Java* and especially *Java 2 platform, Enterprise Edition*[5] (J2EE) that covers the Java platform for developing and deploying enterprise quality applications such as:

- Scalability: e.g. Java provides the developer with the ability to have several connections per thread, which greatly increases the scalability.
- Portability: Applications can run on every platform.
- Easy integration with existing application and data.
- Open standard: J2EE technology is a set of standards that many vendors can implement.

J2EE applications are made up of components. A *J2EE component* is a self-contained functional software unit that is assembled into a J2EE application with its related classes and files and that communicates with other components. The J2EE specification defines the following J2EE components:

- Application clients and applets are components that run on the client.
- Java Servlet[6] and JavaServer Pages[7] (JSP) technology components are Web components that run on the server.
- Enterprise JavaBeans (EJB) components (enterprise beans) are business components that run on the server.

With the EPBOX, I have chosen Java servlets and JSP to handle clients' requests on the server-side (Middle-tier or Web-tier). This choice allows us to use different web servers on multiple platforms.

---

[5] *J2EE technology and its component based model simplifies enterprise development and deployment. The J2EE platform manages the infrastructure and supports the Web services to enable development of secure, robust and interoperable business applications.*

[6] *Servlets are miniature programs that must be run inside another program called a container. The container is responsible for brokering communication between the servlet and the outside world. This servlet container is responsible for instantiating the servlets and calling the appropriate methods in the servlets.*

[7] *JSP* page *is a text document that contains two types of text: static template data, which can be expressed in any text-based format, such as HTML, WML, and XML, and JSP elements, which construct dynamic content.*

A Servlet can be thought of as a Java program similar to an applet that runs on the same machine as the web server. It does not have any user interface and is a way of extending a web server.

JavaServer Pages – an extension of servlet technology - simplify the delivery of dynamic Web content. They enable Web developers to create dynamic content by reusing predefined components and by interacting by the server-side scripting.
JavaServer Pages look like standard Web pages. In fact, JavaServer Pages include fixed template data and JSP elements. These JSP elements enable programmers to insert Java code that interact with components in JSP to perform request processing.
When a JSP-enabled server receives the first request for a JSP, the JSP container translates that JSP into a Java servlet that handles the current and future requests to the JSP.

Depending on the environment, other technologies like MS Active Server Pages, PHP or simple CGI scripts can be used to perform a similar task.

As I mentioned earlier, the EPBOX is a three-tier architecture that includes the client tier, the web-tier and the information tier. In the sections below, I will present how each tier may be implemented.

7.1 The Client Tier

The client tier is the application's user interface. The client interacts with the web-tier to make requests and retrieve data from the application. The client then displays data retrieved from the middle tier to the user.
This tier may consist of two client types such as a Web browser and a Wireless device. Each of these clients can render a different client type that receives a different page because each one (e.g. Web browser or a wireless device) supports different content type.

The first page that users will see when they go to the Treaty Section's site is a "login.jsp" page where every user needs to login with her/his member's username and password. After log in, a user can freely enter all parts of the Treaty Section's site except the administrator's part.
When the Webmaster (or administrator) logs into the site, he/she will encounter another interface with extra roles than those presented to ordinary site's clients.

With this application, almost all web pages will be generated dynamically – on the fly - based on the clients' requests and clients types. In the next section, we will see how clients' requests will be handled on the server side (*Web-Tier*).

7.2. The Web Tier

In this application, I have chosen Java servlets and JSP to handle clients' requests on the server-side.

In this application, we are going to use JSPs when most of the content sent to the client is fixed-template data and only a small portion of the content is generated dynamically with Java code. Such as the case of the login page (e.g. login.jsp) through which the users attempt to get access to the Treaty Section' site and the welcome page (e.g. welcome.jsp) that users get after they log onto the site.

Servlets are used when only a small portion of the content sent to the client is fixed-template data, for example the page containing a search result from the EPBOX database. Some servlets may not produce content. Rather, they perform task on behave of the client, and then invokes other servlets or JSPs to provide a response.

Clients can interact with a servlet/JSP by making ***HTTP*** requests. The server receives the request and hands it to the *servlet container*. The latter is a program that hosts servlets as run time environment.

The servlet container checks to see if any instances of this servlet are already in memory. If not, it loads an instance and runs the servlet's *init*() method.
The servlet container waits for the *init*() method to finish. It then calls the *service*() method in this servlet from a new thread. The service() method calls *doGet*() or *doPost*() method depending on what the request type is.
When the thread finishes, a response is sent to the web server, which forwards it to the client's browser.
Figure 13 shows the architecture of the application graphically.

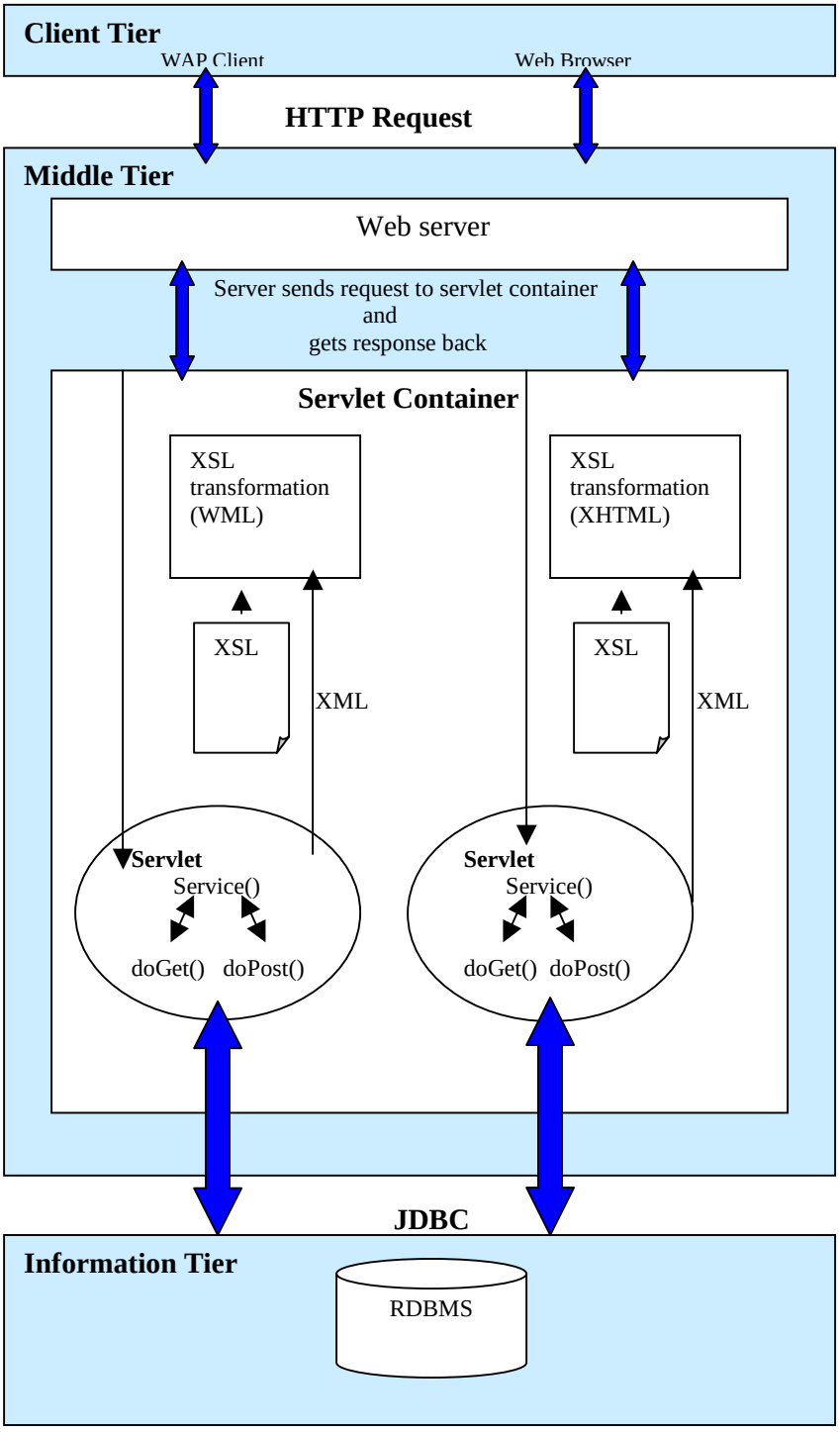


Figure 13: Detailed architecture of the EPBOX application

7.2.1 How will the Middle Tier behave for a regular user?

When an ordinary user logs onto the Treaty Section's site (untreaty.un.org), the Servlet first checks if a database connection is established. If not, it then connects to the database

and checks whether this client is registered or not and if he/she is a Webmaster or not. When the user gets access to the Treaty Section site he/she can interact with the application. The latter then runs the queries sent by the client and , when necessary, marks up the results as XML documents. The servlet, when necessary applies an XSL transformation to the XML document and sends the resulting content back to the client. Note that the clients will be able to register on the site if they do not already have an account. They also must have the possibility to edit or change their member details.

7.2.2 How will the Middle Tier behave for the webmaster?

The server has to perform additional tasks to those mentioned in *Section 7.2.1* in order to handle all possible Webmaster requests. The Webmaster must then log on as administrator in order to manage the website. He then gets a different welcome screen than the clients, with additional rules, such as the possibility to:

- Update the EPBOX' s database using log file produced by the DMS (refer to Section 7.4).
- Extract content and metadata from the DMS output to store them in the EPBOX database (*refer to Section 5.3 Chapter 5*). We have to be aware to avoid data duplication in the database.
- The webmaster also has to be able to see how many members are logged onto the web site and have the possibility to edit, add or delete members of the website.

Remark: EPBOX also has to convert the content to PDF before its storage. For this purpose, third party software can be used and integrated in the EPBOX application.

7.3 The Information Tier

The information tier, or data tier, maintains data for the application. The information tier stores data in a relational database management system (RDMS). For this application, we can use an Oracle database that will interact with the web tier through the Java Database Connectivity (JDBC) driver.

The database would contain:

- Treaty metadata such as title, registration date, place of conclusion, ID in the DMS, etc…
- Content information such as content itself (PDF file), treaty title, content id, treaty ID in the DMS…etc
- Member information such as member id, name, address…etc
- Membership information such as name, username and password.

Note that other tables can be added during the execution phase of this project. *Figure 14* illustrates an initial database diagram.

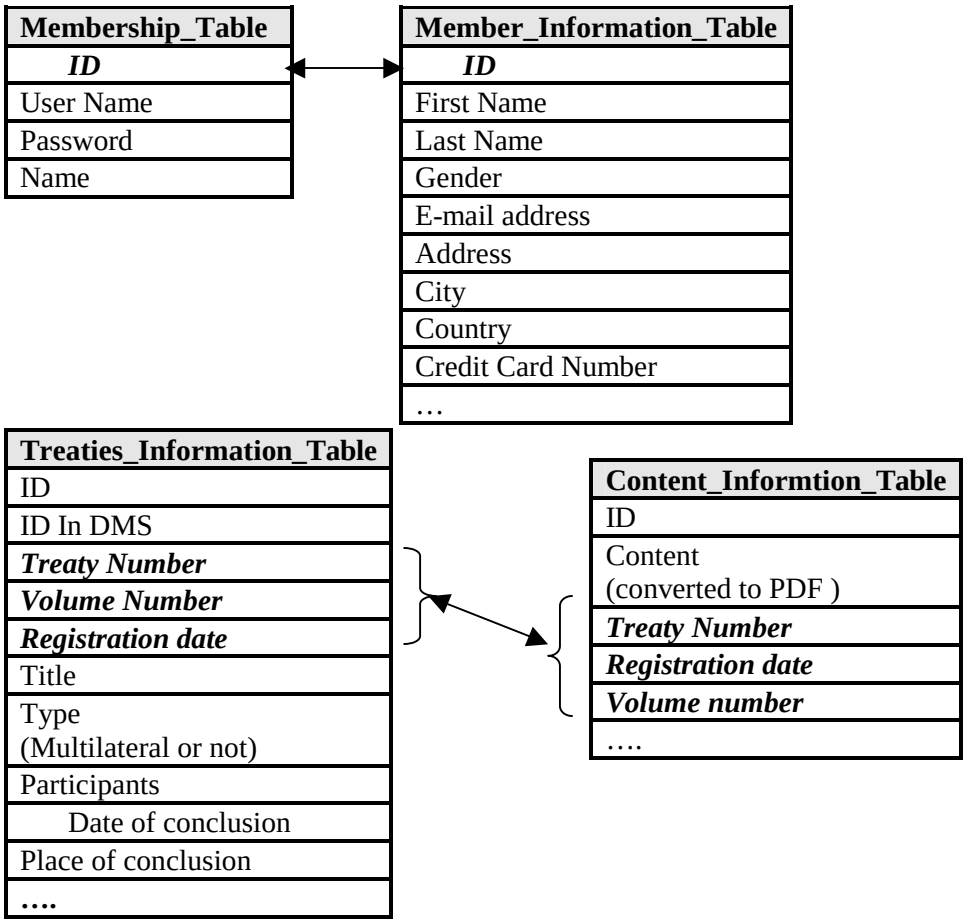


Figure 14 illustrates the database diagram.

In the section below, I will present a possible scenario in order to show the mechanism of a simple servlet request/response invocation via HTTP.

7.4 Possible scenario

This scenario shows how the EPBOX will behave when a client attempts to log into the Treaty Section's site from his web browser.

Step 1: The client goes to the Treaty Section's site and gets the "login" web page.

Step 2: To log onto the site the client enters his username and password. He/She submits his/her login information. The form sends the data to the web-tier.

Step 3: The server receives the client request to log on and forwards this request to the servlet that was invoked.

Step 4: The servlet then invokes the service method (for its configuration when necessary). The servlet then attempts to establish a connection with the database.

Step 5: The servlet sends then a query to the database, which in turn sends the query result back.

Step 6: The servlet uses a function to format, when necessary, XML data based on the query result.

Step 7: Using the XSLT Processor and a series of XSL stylesheets, we can transform XML data for presentation on a variety of clients, including digital assistants as PDAs, and web browsers.

Step 8: Finally, the servlet passes the completed document back to the client browser for presentation to the user.
The flow diagram of this scenario is presented in the figure below.

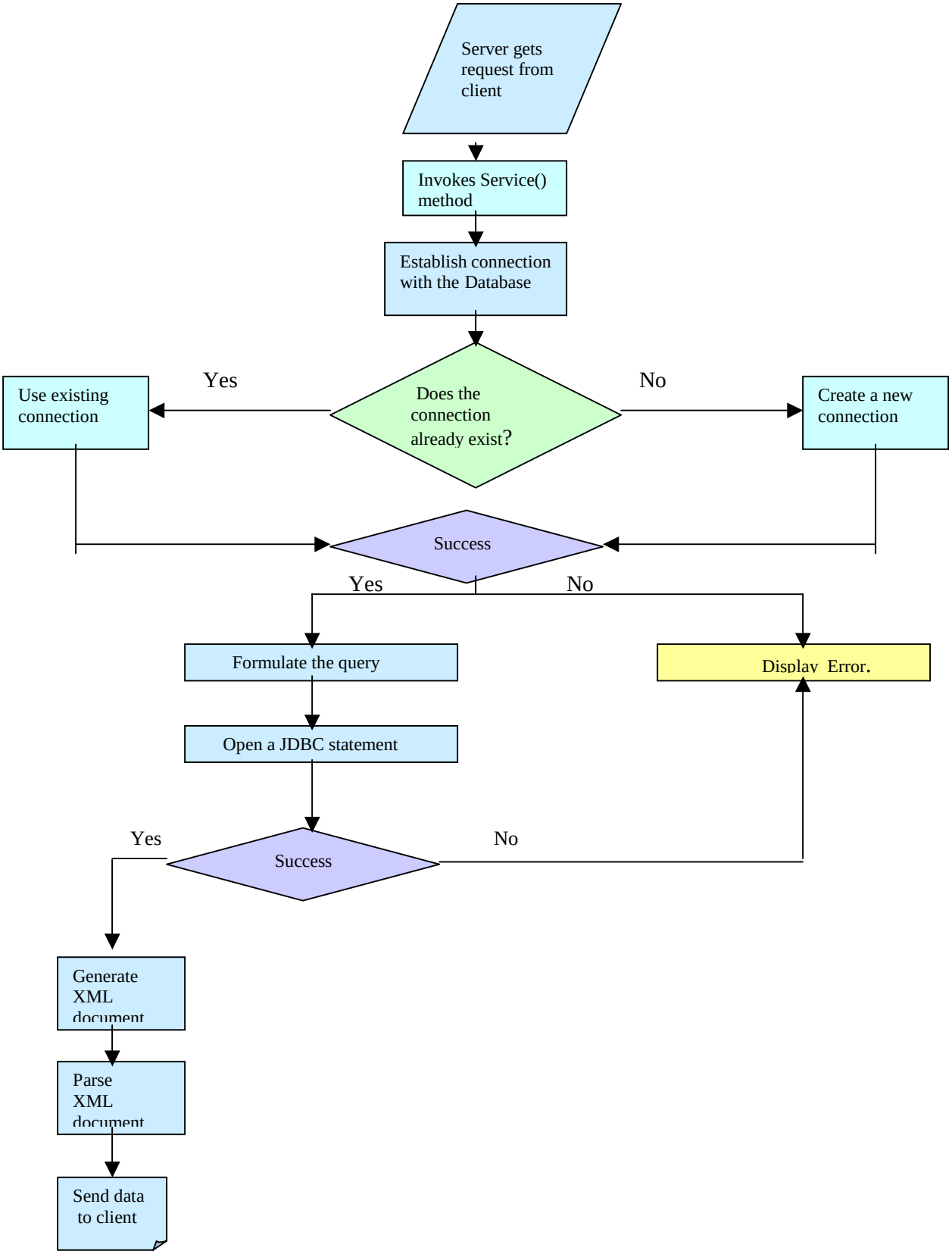


Figure 15: Program flow of the presented scenario

The implementation of the EPBOX is not the only thing that has to happen. The actual DMS and the authoring system also have to be modified.

7.5 Additional Modifications

The EPBOX is going to be effective only when additional modifications are performed at the DMS and the authoring system (*refer to Chapter 5*).

The modifications at the DMS are:

- *The User interface of the DMS* when entering data into the system has to be modified. This modification is necessary in order to enter all needed metadata at the DMS level so that we do not have to add any additional information manually to the produced products as the Bible (*refer to Chapter 2*).
- *The extraction program of the Bible from the DMS* has to be changed. This program has to produce SGML files with all needed metadata. Once this modification is realized the creator does not have to add any information manually after the extraction of products from the DMS *(refer to Chapter 2)*
- *The DMS has to produce log files* with detailed information about treaties that have been modified and what the modifications are.
- *The DMS has to use other log file(s) to update its data*. These log files can be produced by the authoring system when the users change and save their modification. They can contain information about modified treaties and what the modifications are.

From the last point, it is clear that the authoring system also has to be slightly modified. The modifications are related to the production of log file(s) that will contain detailed information concerning the documents that have been modified and what the modifications are when the user of the authoring system performs a change action. The program that has to perform this action can be integrated with the authoring system (FrameMaker+SGML) using the Save API in the *FrameMaker Development Kit*[8] *(FDK)*. The produced log file(s) can be then used by the DMS to update its database.

7.6 The needed tools for components' implementation

The realization of each of the model solution's components does not have the same degree of difficulty and does not make use of the same tools. The table (*figure 16)* displays the needed tools in order to realize each component of the solution model.

[8] *The Frame Developer Kit is an API (application program interface) that allows a C programmer to add custom features to FrameMaker and FrameMaker+SGML. These additions are terms clients or plug-ins and work seamlessly with the Frame product.*

Component	*Needed Tools for realization*
EPBOX application	*Java 2 platform, Enterprise Edition (J2EE) as version 1.4.*
The User interface of the DMS	*DOCBASIC*
The extraction program of the Bible from the DMS	*DOCBASIC*
The DMS has to produce log files after its update	*DOCBASIC*
The DMS has to update its data using log file(s)coming from the authoring system	*DOCBASIC*
The production of log file(s) by the authoring system when users save their changes	*FrameMaker development kit (FDK) and C or C++*

Figure 16 represents the tools needed to realize each component of the model solution.

Chapter 8

Validation

Once the EPBOX and all needed modifications are carried through, most problems of the Treaty Section will be solved.
In the table below, I show how the solution model can meet the Treaty section's needs. In fact, what I prove by this table, is that the solution model – EPBOX and the related modifications - can satisfy almost all the treaty section's needs without the adoption of an expensive solution like an ECMS.

What the Treaty Section needs for her publication system:	***Solution Model***	***ECMS***
The published content on the Internet must be able to be linked back in the DMS.	***Satisfied*** *(The published documents on the Internet can be continuously connected to the server because the choice was made to convert the content to PDF with several defined fields such as the title of the Treaty and the author. These fields and others will be useful to link the content back to the DMS.)*	***Satisfied*** *(Has the ability to manage content through their entire lifecycle)*
Data on the Internet has to be presented dynamically.	***Satisfied*** *(When information is requested, the EPBOX will execute a program that systematically will run queries on the database and generates responses based on queries' results.* *The presented web pages will be tailored to the clients based on their actions, requests or type).*	***Satisfied*** *(It has the ability to deliver dynamically personalized content to users)*
There must be data consistency between what resides in the DMS and what is published.	***Satisfied*** *(In the presented model the documents that are viewed are produced on the fly, which means that when relevant data in the database are updated, the web documents are updated too. On the other hand, the database of the EPBOX will be updated automatically when any changes are reported from the DMS)*	***Satisfied*** (It *has the ability to manage content during its lifecycle)*
Workflow to decentralize Content creation	***Unsatisfied*** *(This characteristic is missing)*	***Satisfied*** *(The Workflow is present in any modern ECM system.* *Workflow is a connected network of activities)*
The solution must be cost effective	***Satisfied*** *(Building, the EPBOX and reengineering the DMS and the authoring system can cost less than 30% of the total price of an ECMS.)*	***Unsatisfied*** *(The adoption of an ECM system costs millions of dollars an amount that the Treaty Section can not pay)*
The solution Once adopted must reduce the actual costs.	***Satisfied*** *(Once this solution is adopted many task will be performed automatically which means that Treaty Section can gain a lot of time and can reduce the number of people, about 17, that are actually working on the publication processes)*	***Satisfied*** *(With its total information management from end-to-end, the ECMS will be advantageous for the Treaty Section's future)*

I have to mention that the scope of the project and the limited time that was available to analyze the Treaty Section's system has meant that some aspects of the current publication system could not be studied deeply.

What I have not mentioned earlier is that there are two systems hosting Treaty information, the DMS and a web-publisher system called Liberty.
The Liberty system contains data from before May 1998, and Documentum (the DMS) hosts data from after May 1998. Therefore, the Treaty Section needs that the data of both systems be merged in order to facilitate its management.
During my internship, I was not able to devote much time to analyze the Liberty System's data to find out if it is possible to migrate it in the EPBOX database.
All what I can say about this point is once the Liberty's data is migrated to EPBOX system, it will be very easy for the Treaty Section to manage ***all*** its data because it will be concentrated in one system, EPBOX, instead of two.

Chapter 9

Learning points

Analyzing the Treaty Section's publication system has opened the door for me to other practices of the Computer Science.
Before my internship, my knowledge about Content Management Systems was limited. I did not know how useful this technology could be for an organization to manage its content.

Before reengineering the publication system, I had to understand it, but my task was made very difficult due to a lack of documentation. Therefore, when I was asked to fix some system bugs and to add extra functionality to the used DMS, I had first to analyze the source code and interact with the users in order to understand how this system behaves. This phase was the most difficult, but at the same time, it also was very educative. It offered me the possibility to improve on my client facing skills.
During the implementation phase of the extra functionalities that I was asked to add, I learned new programming languages such as *Omnimark* and *Visual Basic*. I also had the opportunity to discover new technologies to me, such as SGML and XML.

Chapter 10

Conclusion

The reengineering of the Treaty Section's present system was a difficult task that encompassed a series of activities, encompassing the analysis of the whole system and its programs before starting the rebuilding activity using modern technologies and reusing of some components that were judged useful from the analysis phase.

The presented model is not as perfect as an ECMS but it covers the most important present and future needs of the Treaty Section. Moreover, it is less expensive and will last for many years (as first estimation: about 10 years).

The presented solution model for the Treaty Section presents a good starting point for the Section.
The Treaty section's managers have mentioned their interest in this solution model. So, the EPBOX and the needed modifications presented earlier may become a reality.

The realization of the Model is not sufficient to solve ***all*** problems of the Treaty Section. What the Treaty Section also needs is a detailed documentation of the whole publication system. As previously noted, there is a lack of documentation in the present system, which makes the software's maintenance very difficult. In general, a program with weak documentation will be very difficult to adapt and maintain. The Treaty Section has to think seriously about this last point and when the documentation problem is solved, any software (re)engineering task in the future will be very easy.

Appendix A

A.1 Publication tools

There are many tools used in the different publication processes. Those tools are:

- Documentum 4i (A Document management software)
- FrameMaker+SGML.
- Omnimark Pro.
- WordPerfect.
- Accent Capture.
- Omnimark.
- Other programs written in C/C++, DocBasic, FDK and Visual Basic languages.

A.2 Additional tools

The production of minor publications requires specific utilities, the most important of which are:

- tstools.dll:, developed in the C language, contains functions that manipulate SGML templates[A3] to incorporate attributes extracted from the database into the SGML templates.
- Different batch files that are adopted for conversion matters, e.g. SGML2WP.BAT which is used to convert specific characters to ISO entities in SGML documents that are automatically generated.
- WordPerfect macros that are used to format SGML documents before there display in WordPerfect.

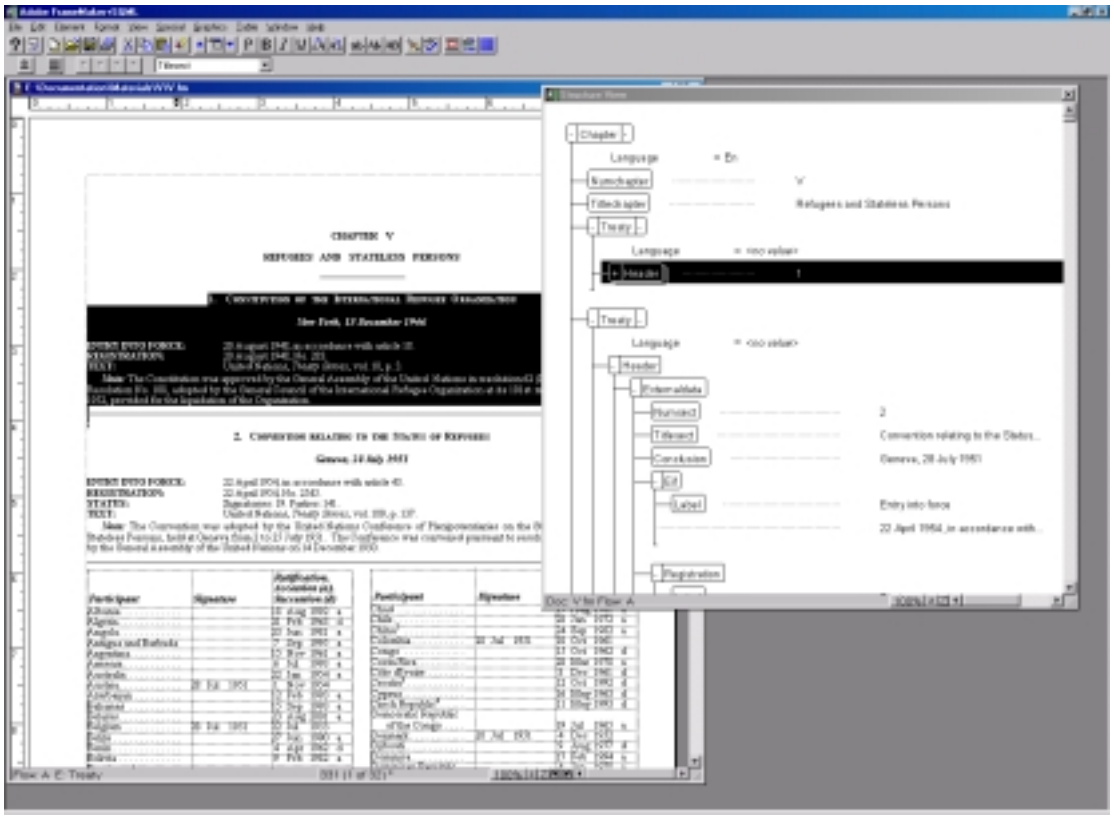
A.3 Template files

The template files, predefined text and unique codes of each template, allow the conversion tool how the attributes extracted from the database of the document management system should be incorporated in order to produce the SGML document in the desired format.

Appendix B

B1 FrameMaker+SGML structured documents

Once SGML files of a given product are extracted from the DMS, they are then compiled and edited by the *Authoring system* (in this case, the FrameMaker+SGML). After their compilation, the products can look like the example of the screen below.



A document, from *the bible*, after its compilation in FrameMaker+SGML, and its structure at its right side.

Appendix C

C1

Example of an extracted SGML file that contains metadata.

```
<TreatyNumber>34726</TreatyNumber>
<EngTreatyHeader>
<Participants>Austria<newline>and<newline>OPEC Fund for International Development</Participants>
<TreatyTitle destination = "TOC180032b98">Agreement between the Austrian Federal Government and the OPEC Fund for
International Development regarding the definition of the headquarters of the Fund</TreatyTitle>
<Attachments>with map</Attachments>
<Conclusion>Vienna, 13 April 1983</Conclusion>
<EIF><label>Entry into force</label>1 June 1983, in accordance with article II</EIF>
<Languages><label>Authentic texts</label>English and German</Languages>
<Registration><label>Registration with the Secretariat of the United Nations</label>Austria, 3 June 1998</Registration>
<LimitedPublication status = "Y"></LimitedPublication></EngTreatyHeader>
<FrTreatyHeader>
<Participants>Autriche<newline>et<newline>Fonds de l'OPEP pour le d&eacute;veloppement international</Participants>
<TreatyTitle destination = "TOC280032b98">Accord entre le Gouvernement f&eacute;d&eacute;ral d'Autriche et le Fonds de
l'OPEP pour le d&eacute;veloppement international relatif &agrave; la d&eacute;finition du si&egrave;ge du
Fonds</TreatyTitle>
<Attachments>avec carte</Attachments>
<Conclusion>Vienne, 13 avril 1983</Conclusion>
<EIF><label>Entr&eacute;e en vigueur</label>1er juin 1983, conform&eacute;ment &agrave; l'article II</EIF>
<Languages><label>Textes authentiques</label>anglais et allemand</Languages>
<Registration><label>Enregistrement aupr&egrave;s du Secr&eacute;tariat des Nations Unies</label>Autriche, 3 juin
1998</Registration>
<LimitedPublication status = "Y"></LimitedPublication></FrTreatyHeader>
<Footer>Volume 2017, I-34726</Footer>
<Registration.year>1998</Registration.year>
</TreatyHeader>
<ImageDocument>
<Footer>Volume 2017, I-34726</Footer>
<Registration.year>1998</Registration.year>
<DocumentType>[ English text &mdash; Texte anglais ]</DocumentType>
<Image.legal entity = "ID80065c08001">
<Image.legal entity = "ID80065c08002">
<Image.legal entity = "ID80065c08003">
</ImageDocument>
```

C2

The DMS produces log files when products are extracted. Those log files are used to link the content to its metadata.

In the example below, the log file was produced by the DMS when a Treaty Series book was extracted from the DMS. This file contains the ID’s of each treaty in the DMS and the corresponding treaty number.

```
Creating Part: 1
Treaty 08001f8780032b98 number 34726

Treaty 08001f878000b448 number 34727

Treaty 08001f8780015407 number 34728

Treaty 08001f8780015406 number 34729

Treaty 08001f8780020851 number 34730

Treaty 08001f878000a2b1 number 34731

Treaty 08001f8780017d43 number 34732
:
:
```

References & Bibliography:

[1]: *Roger S. Pressman, "Software Engineering, a practitioner's approach", Fourth edition. The McGraw-Hill companies, Inc.*

[2]: *"Technical Fundamentals for Documentum 4i", Documentum, Inc. 6801 KollCenter Parkway, Pleasanton, CA 94566*

[3]: *Gary L. Ashton, "Using a Document Management System to Manage a Web Site", Novell Inc.* http://www.w3.org/Authoring961001/ashton.html

[4]: *C. M. Sperberg-McQueen and Lou Bernard, "A gentle introduction to SGML", chapter 2 of the TEI Guidelines*

[5]: *PC AI Magazine, Nov/Dec 2000 V.14 #6*

[6]: *Thomas Lites, "Content Management: What's in it for me?", November 2002* http://knowledgemanagement.ittoolbox.com/

[7]: *Michael J. Halm and Michael Pelikan, White paper: "Enterprise Content Management System", May 2002* http://knowledgemanagement.ittoolbox.com/

[8]: *Lisa-Jean Boulet, "XML to the rescue! Improving the Efficiently of Web Content Management".*

[9]: http://www.adobe.com/

[10]: *Steven Holzner, Inside XSLT, ISBN: 0735711364, First edition, july 2001.* http://webreference.com/authoring/languages/xml/insidexslt/

[11]: http://www.w3c.org/

[12]: *Micheal Trafton, "What is Documentum?", Blue fish Development Group* http://www.dmdeveloper.com/articles/introduction/whatisdctm.html

[13]: *Arofan T. Gregory, "How XML Enables Internet Trading Communities and Marketplaces"* http://www.infoloom.com/gcaconfs/WEB/philadelphia99/agregory.HTM

[14]: *Ann Adams, "An XML Language for building Device-Independent user Interfaces".* http://www.infoloom.com/gcaconfs/WEB/philadelphia99/adams.HTM

[15]: *Prof. Stephan Morris (created by Apriliani Kartiko), "Telecommunication Application and Management".* http://www.usfca.edu

[16]:*James Robertson, "How to evaluate a CMS", Januari 2002.* http://www.steptwo.com.au/papaers/kmc_evaluate/:

[17]: *Michael Maziarka, "Content Management Solutions" (white paper),May 2002.* http://www.broadvision.com

[18]: *"What is Web Content Management?", CompuCraft Web site.* http://www.compucraft.com.au/ECM.htm

[19]: *"eXtensible Markup language", Report of 1999.* http://www.usfca.edu/fac-staff/morriss/651/tech_projects/XML/report.htm

[20]: *Dino Esposito, "Cutting Edge", August 1999, The MSDN Library 2000.*

[21]: *Paul Parry, "Delivering Information With XML", The MSDN library 2000.*

[22]: *H. M. Deitel, P. J. Deitel, S. E. Santry. Advanced Java 2 Platform "How To program", Prentice Hall.*