document solutions

EXPLOit
An SGML/XML Content Management Data Base

Introduction

EXPLOit handles structured documents (SGML or XML) contents: Adobe FrameMaker is the visualizing and publishing system while Microsoft SQL Server is the storing system.

Documents are stored in fragments of diversified granularity, according to hierarchical levels of the document structure; the choice of these levels is guided by the DTD together with the initial project configuration based on user needs.

Storing fragments leads to the idea of reuse of common parts of different documents through references to them, without duplication. Fragments can be assembled to produce the complete document to be visualized, edited or delivered in several formats (print, XML, PDF, HTML, etc.).

This kind of database structure allows to deal with the whole life cycle of each fragment, which can be handled as a single individuality or as part of several documents, simultaneously. Many central activities involved in document creation and maintenance, like multilanguage translation or revision control, can be heavily facilitated.

Functionalities

Here below procedures to access data stored in the DB and main functionalities of the application are described.

Fragments: storing and access

Each fragment is uniquely identified by means of its name and the language in which is written. inside, it can include, as sub parts, references to smaller fragments.

Hence, the document is represented inside the archive by a higher level fragment, which contains references to all parts and sub parts. All typical information needed for its management are associated to the document: author, revision number, creation data, modification data, status and so on.

Several types of attachments can be associated to the document, like certificates, information sources, etc.: this files will be stored in the DB; additionally, the final publication can contain SGML/XML documents as well as files in different formats, chosen from that ones admitted by FrameMaker. Those files will be considered like static attachments to the final publication, rather than contents fragments.

The user interface allows navigating in the document structure, to investigate and administer fragments inclusion. The hierarchical tree represented in the interface looks like file system directories (see fig. 1).

It is possible to see a preview of each fragment (see fig. 2).

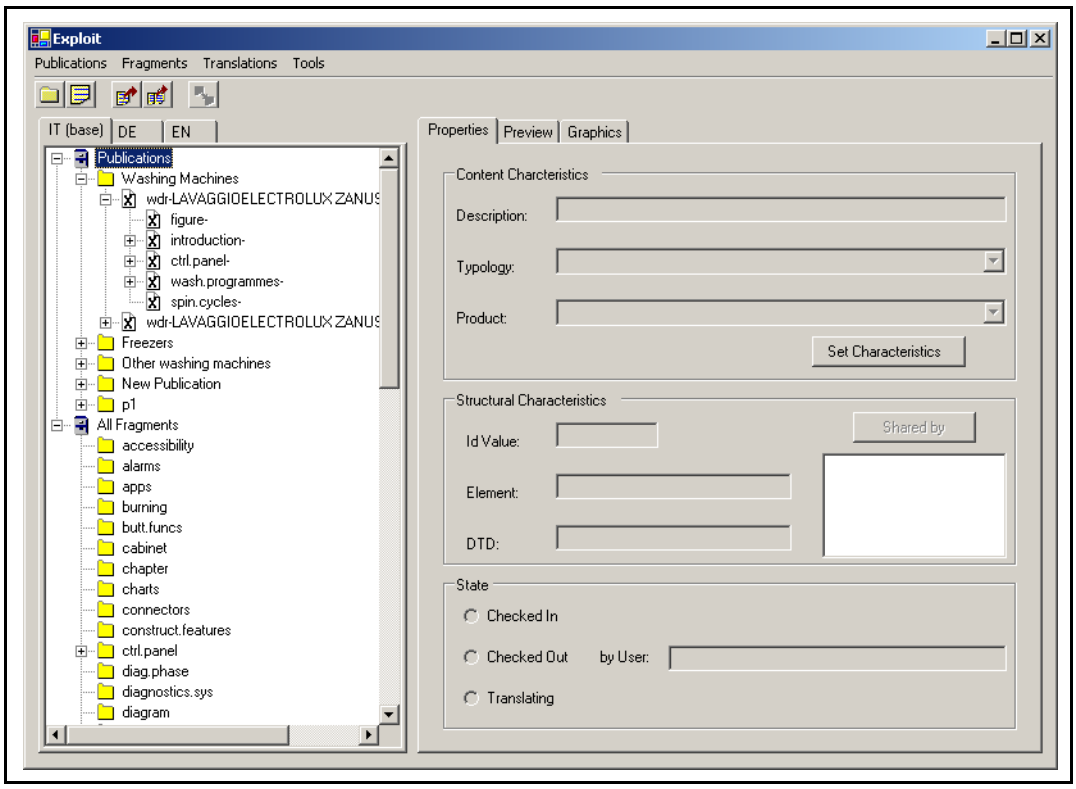

Figure 1: View of fragments included in documents

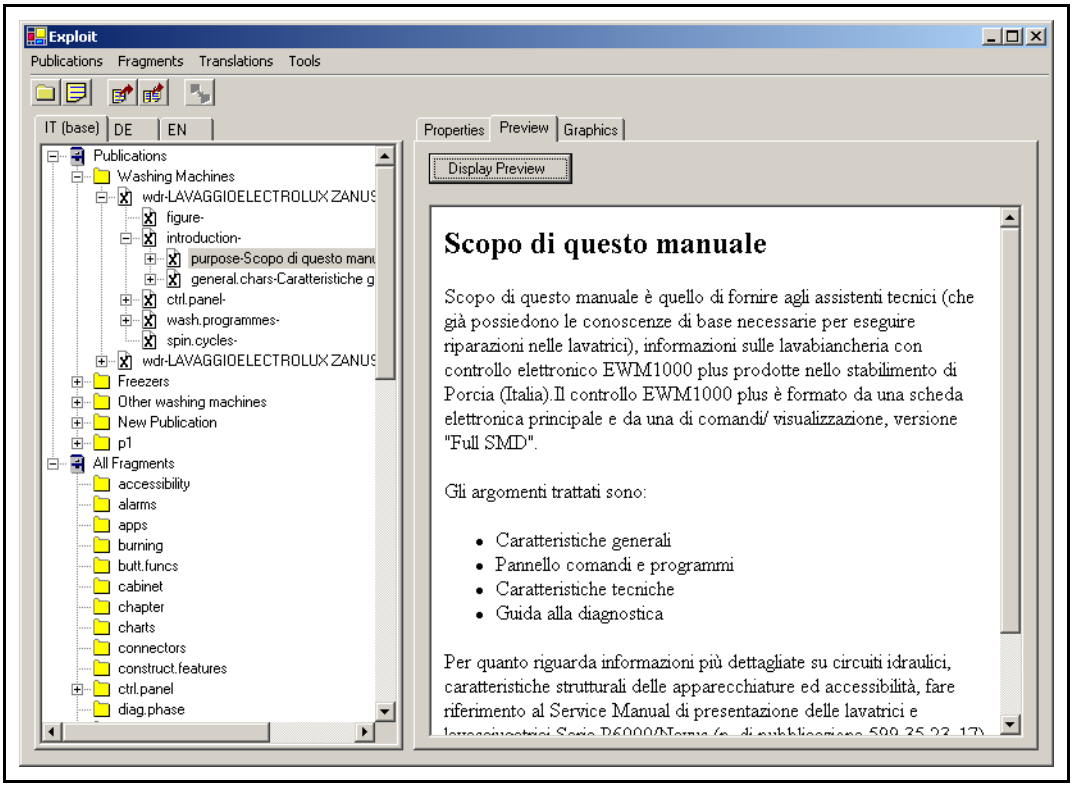

Figure 2: Preview of a fragment

As already said, each fragment can be part of one or more documents which refer to it, but it can also be not included in documents, because, for example, it is still under construction (translation, earliest composition, etc.); in any case, user needs to find in the archive any fragment. Therefore, fragments must be reachable also through their contents meaning, in other words through a category to which it belongs.

An example will better explain. A “Foreword” element will be visualized in two ways: as a fragment of a larger document or as one of several fragment of “Foreword” type included in the DB.

Translation and multi language versions

Fragments can also be handled for translation activities: it can be checked-out, delivered to the translator and later checked-in the DB.

Each fragment contains an attribute to identify the language; the attribute, together with an additional value, will form the unique name for the fragment.

This compound name allows an easier implementation of routines which deal with documents in more than one language: when a new language is required for an existent document, the application duplicates the document tree with references to each part and sub part, changing only the language attribute and obtaining the new unique name for the new language fragments.

The user can specify which languages he wants to handle with and immediately obtain the visualization of fragments tree in the preferred language: he can see all translated fragments, all fragments that need to be translated and those needing a new translation because they have been modified in the basic language (see fig. 3).

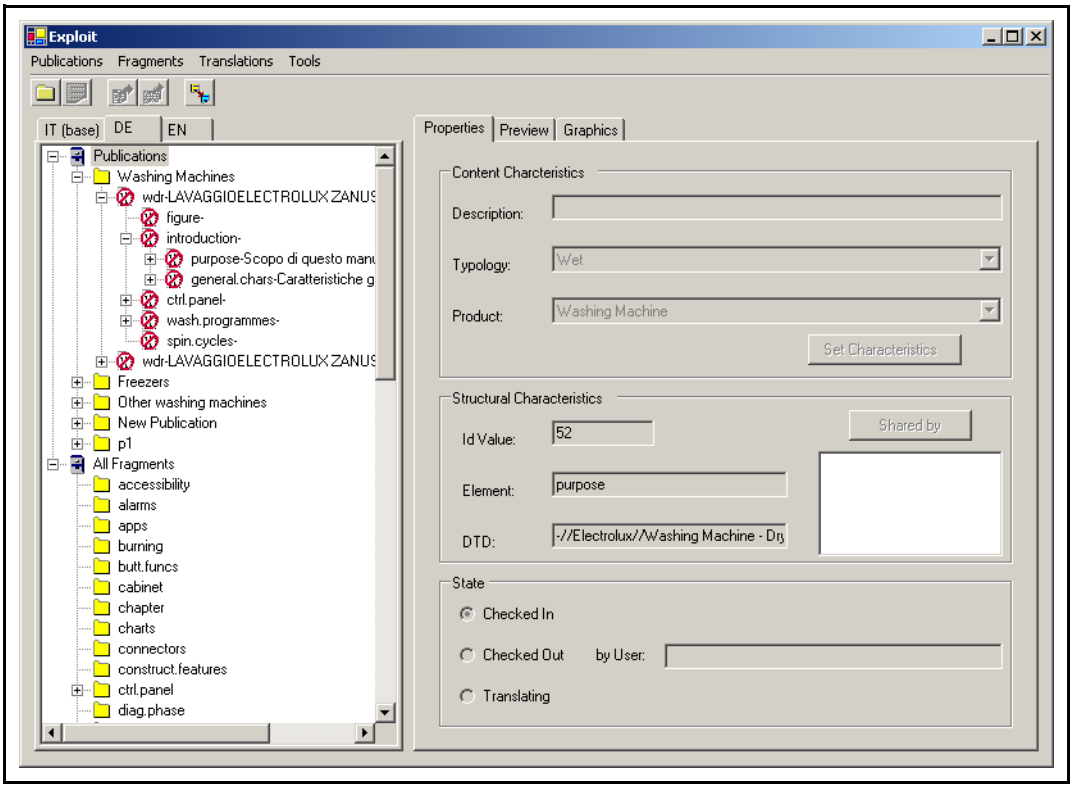

Figure 3: Browser on different language contents

User can choose the right fragment to be send to translator, navigating into the specific language view. The fragment is extracted together with all referred sub parts needing to be translated too, while a reference is inserted in place of those already translated.

EXPLOit directly communicates with **EleXML,** a Logos' product which pre-translates XML fragments, allowing users working in an integrated environment in which they can control:

- contents storing concept
- contents editing and publishing
- documents translation and handling activities in all languages

Graphics handling

Graphics included into documents are handled through standard functionalities of FrameMaker: all standard formats supported by FrameMaker can be used; further, a new functionality to analyse figures have been added.

EXPLOit allows users to manage graphics repository, asking for preview of figures and controlling their use in fragments (see fig. 4).

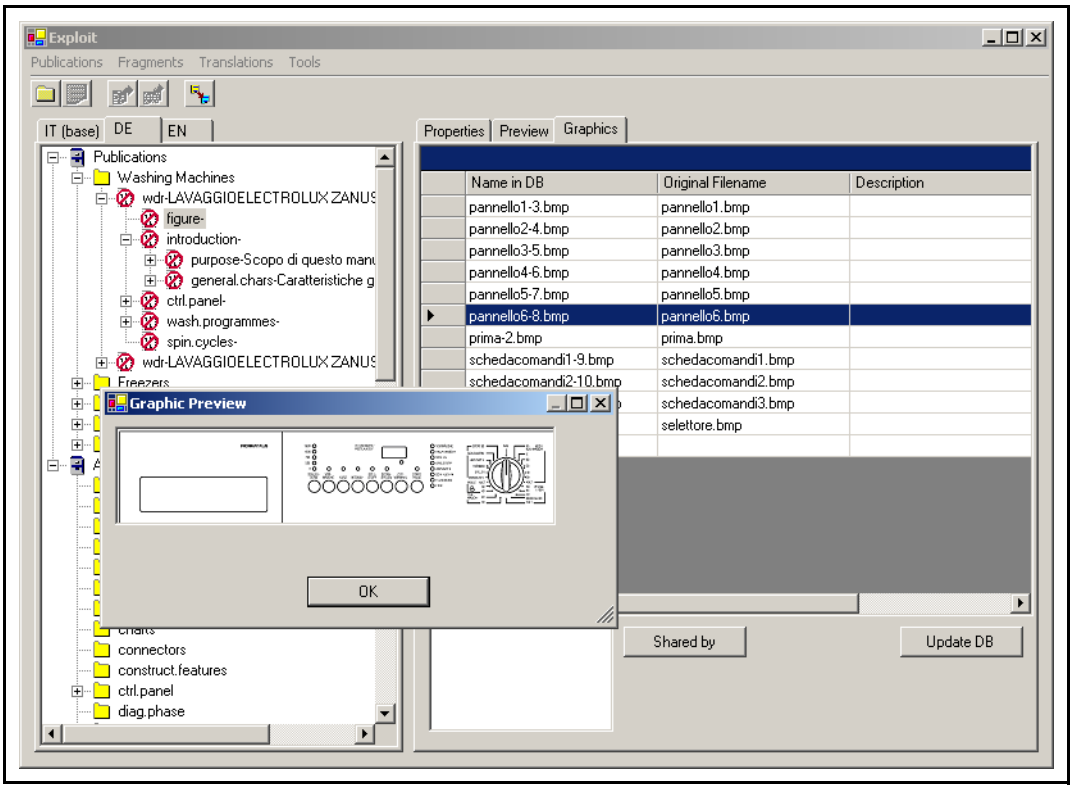

Figure 4: Graphics archive

EXPLOit controls images sharing between documents: if a new document inserted into the DB contains figures already stored, they are not duplicated, because are identified as already classified, and are shared with fragments including them.

Publishing

FrameMaker is the SGML o XML interpreter of choice. Users can define their own "SGML/XML Application" depending on specific publication purposes: in other words, users can choose to use specific *templates* and/or DTD/EDD (Element Definition Document) relying upon particular output being produced.

Output formats are those of FrameMaker: Postscript (print), PDF, HTML, XML, SGML, RTF, Microsoft Word, etc.

Workflow

EXPLOit allows to create several kind of users, one for each specific task - Administrator, Author, Translator, Graphics Manager, etc. - with different rights and tasks to be accomplished.

A project defines working progress for each document and users can access/modify documents depending on his role and document status.

Activities affecting document life cycle are monitored and produce time statistics on demand. At any time, users having rights to access such information, can ask for listing of completed documents, in progress documents and so on.