5 June 2003

Topologi Application Note

A Publishing Solution using Topologi's ***Universal Wire***

info@topologi.com

Description of a reference site of a representative Universal Wire installation

CUSTOMER

The customer is a small, established and respected XML/SGML/PDF publishing-systems integrator, with a capacity for project managing data conversion and markup jobs. They run Windows and Linux systems, and have existing databases, scripts, file servers, intranet Webservers and batch typesetters which were previously not integrated.

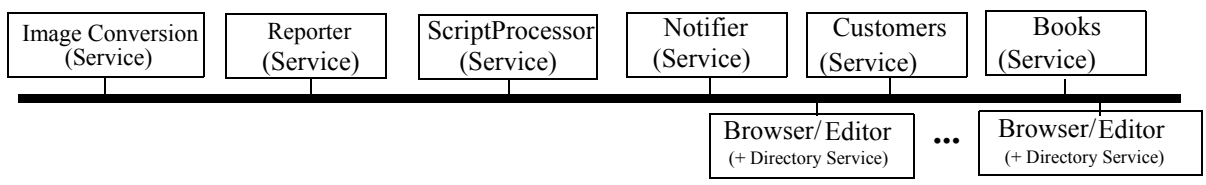
OVERVIEW

The system has two parts. The first retrofits a Universal Wire solution to integrate their existing general production systems. The second uses a Universal Wire solution to integrate the components of a particular large job, the complete documentation of a military equipment platform, which involved conversion of almost a quarter of a million scanned page and seventy thousand images; further more, various parts of the data conversions were outsourced, or spread around several geographic locations. Even more complex, the system has to fit into a strict data migration regime, because some of the data sources were under continual updates, unlike a data conversion job where the source data was static. An additional requirement was a very high accuracy and quality requirement, including the use of external auditors.

SOLUTION

All users run the TreeWorld browser/editor on their local machines. One server system is used to run all the Universal Wire services. The services include the off-the-shelf services such as *Topologi Reporter*, *Topologi ScriptProcessor*, and *Topologi Notifier*, as well as custom services to interact with the customer's databases.

FIGURE 1. Topology

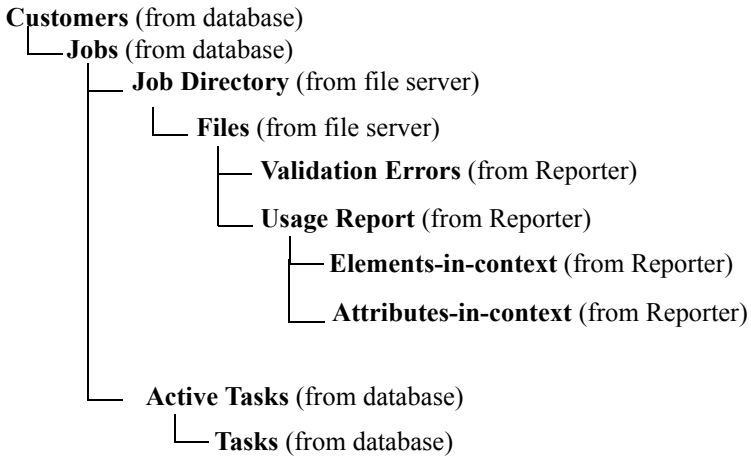


SOLUTION PART 1: GENERAL PRODUCTION SYSTEM

We looked at the client's databases, files and workflow to see which objects form a natural hierarchy, and to see how users' think about the data, in particular how the users can drill down: what are the natural starting points and what related information would or should be available at each next step. The solution addresses the three major problems which users reported:

- Inconvenience in remembering the job number for a job, and then locating that job on various file servers (jobs could be moved to different servers to balance load or to match disk capacity).
- Inconvenience in remembering the job number for a job, and then locating that task lists for that job from the existing job application.
- Difficulty in creation and use of command-line tools for all kinds of quality assurance purposes, for example validation, sampling usage, detection of markup anomalies. This is coupled to management's need to track and verify the status of projects, as well as making it possible for programmers and markup staff to switch to a new job and perform common tasks quickly.

A service was created which queries the job control databases and provides a list of customers to the user's TreeWorld interface. This allowed the following kind of tree of information:



Selecting a graphics file or an XML or SGML file causes that file to be displayed in a preview pane.

Utilizing the *Topologi Reporter* service, the user can select XML files and check their validity or well-formedness. If there is an error, it is displayed underneath the file. Opening one of the errors takes the user to the *Topologi Collaborative Markup Editor,* where the error can be examined and corrected more. The user can also select files and perform many kinds of formatting and search-and-replace operations.

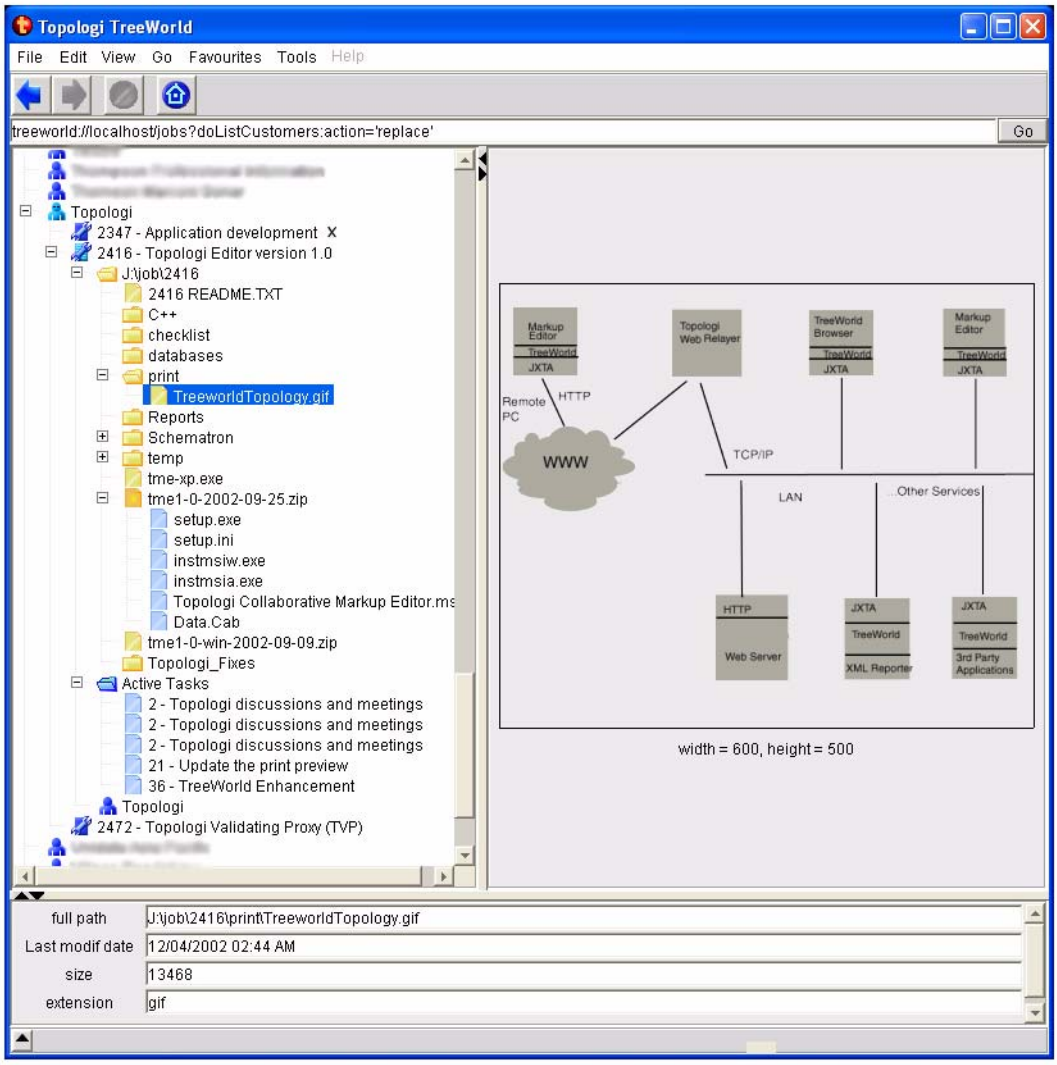
The user can select an XML file and generate a usage report: this displays which elements have been used and where. The user can select a group of XML files and generate a combined report: the report replaces the current data view, but the user can return to the previous view using the back-arrow button, in the fashion of a Web browser. The user can examine this report to see whether all files have some element, or whether an element has been used in some files only, for example. The user can select some XML files and generate a usage schema; this schema can then be used to check that other, newer files do not have elements used in different ways than those of the sampled documents: this is important for consistency and to reduce

the programming effort to only cope with elements or contexts that are actually present.

Another service was also created to start with a list of all active jobs. The database has a flag for whether a job has been archived or whether it is on the file servers: archiving is indicated by a black cross. Sometimes a job may be archived without the database being correctly updated: a red cross is shown against a job which is marked active but has no directory.

Following is a screenshot of the system in place; customer details have been obscured for business confidentiality reasons.

FIGURE 2. Screen shot of User Interface for Part 1 solution



The figure shows the TreeWorld browser in operation. At the top are menus and back, forwards, and home buttons, like a Web browser. At the left is the information tree, which is the main focus of navigating and selecting information. At the right is

a panel which is used for previewing files. At the bottom is the attribute pane, which is used to view attributes of the currently select element in the tree. When you right-click on an object in the tree (or multiple objects) the browser will display a menu with the various actions available from different services for those objects. The more services you add, the richer the actions that will be available to the user.

In this figure, you can see an example customer “Topologi” is open. This customer has three jobs: 2347 (which is archived), 2416 (which we are looking into) and 2472. Inside the 2416 job we can the home directory for that job, opened with various files and subdirectories open. The file TreeworldTopologi.gif is currently selected: it is being shown in scaled preview in the right-hand pane and its attributes are shown in the bottom pane.

Further down, the user has opened a zip file, and the TreeWorld browser is displaying its content files. Services can be created to inspect inside any kind of file.

After the directory, we have a list of the active tasks. This list was much longer when the information was originally retrieved from the service, however the TreeWorld browser allows you to sort and select information based on the attributes available.

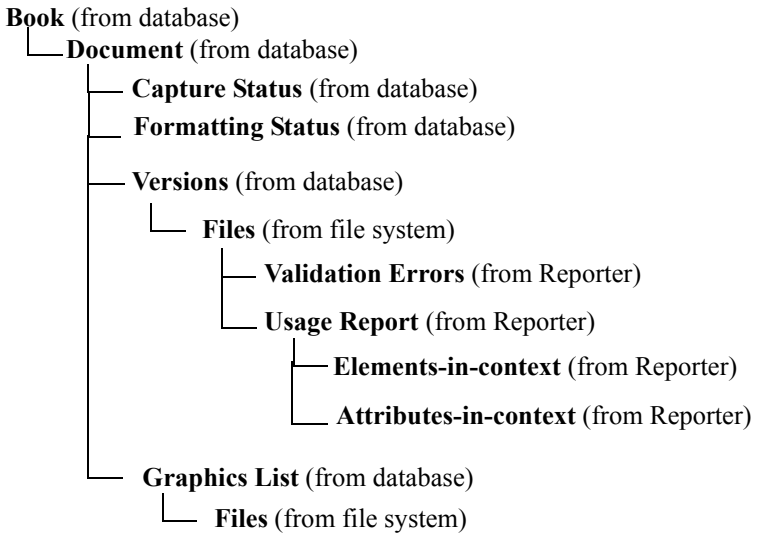
(For those viewing a colour version of this document, note that the Dark Blue icons comes from the database service, the yellow icons come from the built-in file service, and the light-blue icons are default icons used when no special icons are provided: here they are used for the contents of zip files and for tasks.)

SOLUTION PART 2: LARGE DOCUMENT CONVERSION SYSTEM

We looked at the client’s databases, files and workflow to see which objects form a natural hierarchy, and to see how users’ think about the data, in particular how the users can drill down: what are the natural starting points and what related information would or should be available at each next step. The solution addresses the three major issues which the clients reported:

- Need for project management to have flexible, convenient and direct inspection of the status of data. The extreme requirements for data migration and the extremely large size of the project made it essential to have
- Need to conveniently track the status of jobs as they progress through stages, and to allow operators to notify the next stage when a document or file was ready.
- Need to demonstrate to the client’s client that the system was under control, with a demonstrable quality assurance regime.

A service was created which queries the database and generates a list of Books for the user's TreeWorld interface. This allowed the following kind of tree of information:



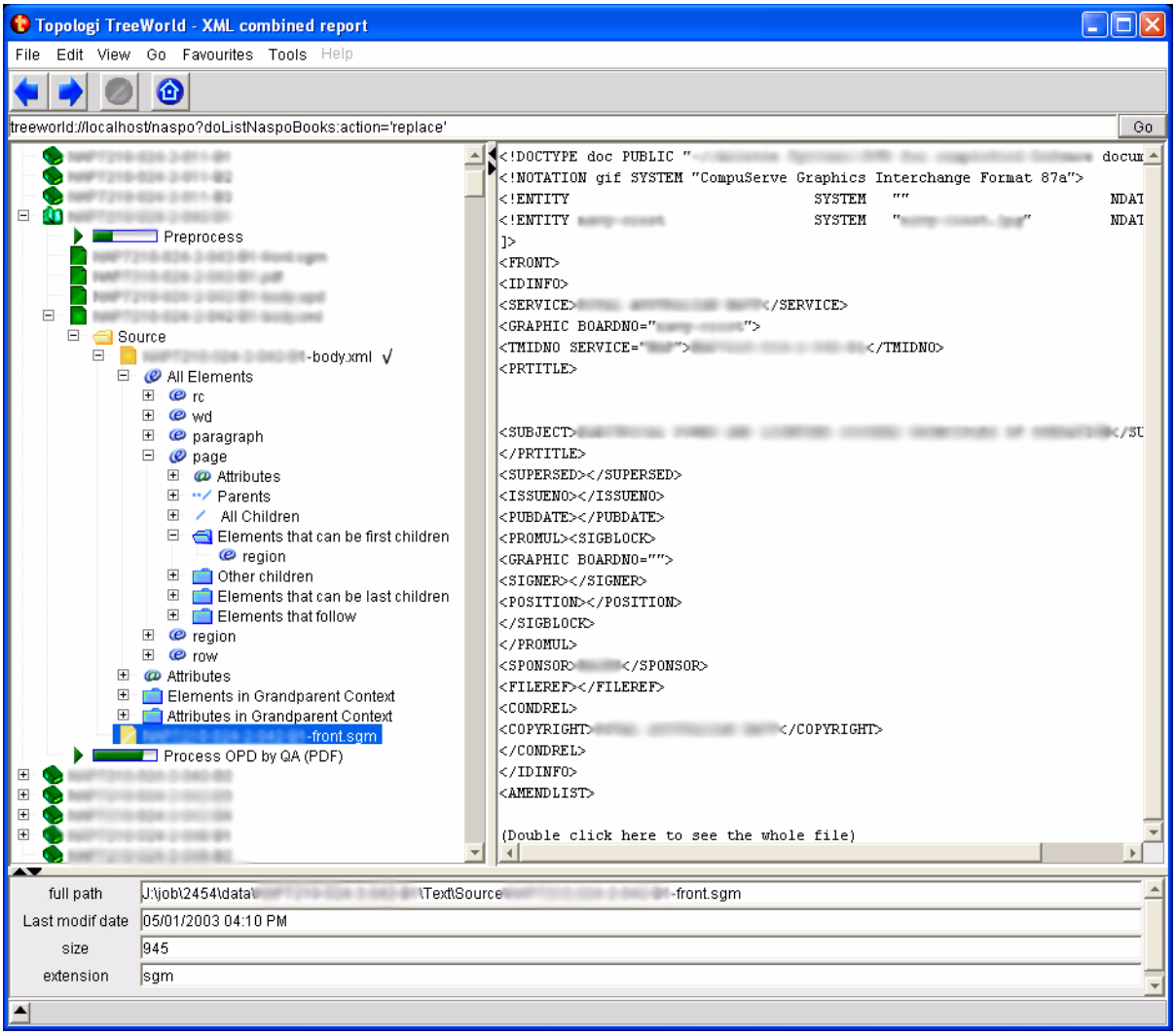
Below is a screenshot of the system in place; details have been obscured for business confidentiality reasons.

In this figure, the top-level of icons in the left-hand panel gives all the books in the project. One book has been opened, and you can see six elements are being displayed: first an indication of the capture stage ("Preprocess") and percentage complete (giving in an attribute in the attribute panel and displayed with a bar graph); next there are four versions of the data (obscured, but these are .opd. .sgm, .xml and .pdf, with the document being converted from one to the next); finally there is an indication of the markup and formatting state ("Process OPD by QA").

One of the (obscured) files, an XML file, has been selected and the *Topologi Reporters* "Generate Individual Usage Report" has been invoked. The results of the usage report are displayed under the file's icon. The report shows all the elements and attributes used in the file (with counts) and the grandparent context. In the figure, we can see that the usage of the element "page" is being inspected: for example, we can see that it can have a first child (i.e. the first subelement) of "region".

Further down in the figure, an SGML file has been selected. We can see its file attributes in the attribute panel at the bottom, and a preview in the right-hand panel.

FIGURE 3. Screen shot of User Interface for Part 2 solution



Other services created include

- a notifier service, which lets a user select a document and progress it to another stage, emailing the next user,
- preview/typesetting service, sending the document to an Advent 3B2 batch formatter to create the PDF files, and
- image conversion service, allowing the user to select image files and convert them to different formats.