International Spring School on the Digital Library and E-publishing for Science and Technology
CERN, Geneva, Switzerland, 3 – 8 March 2002

IT and Electronic Publishing
Issues, Trends and Developments

Teun Nijssen
Senior Project Manager, Tilburg University, Computer Centre, The Netherlands

1 INTRODUCTION

This text is the paper reflection of a presentation given on 4 March 2002 in Geneva as part of Ticer's International Spring School on the Digital Library. The presentation is of an introductory nature and aims at providing enough technological background to set the stage for more detailed subjects later in the Summer School.

2 DIGITAL LIBRARY PROJECTS AT TILBURG UNIVERSITY

In the past decade Tilburg University Library, often in cooperation with Tilburg University Computer Centre, has done numerous projects that were related to providing an end-user environment of which a modern Digital Library can be proud. Some of these projects integrated day-to-day office tools into a well-managed environment and others added Internet facilities.

There has also been a set of projects that applied emerging technology to construct what is now called a pre-print server or an internationally distributed full-text scientific document service. The relevant issues, trends and developments can be made clear by extrapolating what was done in the most important five technology projects of the past. So these are summarised first.

2.1 KWIK

In the early 1990s, Tilburg University started a project called Kwik (pronounce "Quick"). The Dutch word kwik is a literal translation of the word mercury, the liquid metal. At Carnegie Mellon University in Pittsburgh a project called Mercury had produced an integrated service that allowed users to do free text searches in scientific journal texts. Also, the full text of the original publications could be displayed on screen, and printed on a laser printer. The system ran a pilot service on the CMU campus; visitors were impressed.

The main disadvantage of Mercury was that it only ran on Unix workstations. In the early nineties PCs ran Windows 3.1. WWW browsers were available, but HTML Forms and web server CGI programming were still in the future, so user interfaces were hand crafted. CMU had learned a lot on the Unix/Motif graphical user interface.

Tilburg University library and computer centre set out to make a Windows GUI client in cooperation with CMU and Digital Equipment. Articles from Elsevier journals were used to start a service. Once again, visitors were quite impressed with the results, but the project proved to be only a model for future developments. It did not become something that was used elsewhere on the world.

Several reasons contributed to this:

- Not too many libraries yet had contracts for the supply of digital versions of copyrighted journal publications.
- It is amazingly hard to introduce a piece of software for specific narrow purposes on many PCs. The Kwik client simply didn't make it to 100,000 PCs.
- Not too much after Kwik, the web got Forms, CGIs and Get and Post methods.

But Tilburg University learned; about parsing metadata supplied by publishers, about Z39.50 and about storage of documents. The stage was set for follow-up projects.

2.2 ELISE AND ELISE II

Between 1993 and 1995, Tilburg University became a participant of the EU project Elise ("Electronic Library Image Services in Europe"). De Montfort University in Leicester led the project; other partners were IBM UK Scientific and the Victoria & Albert museum.

When the project was conceived, the Joint Photographer's Expert Group had specified the JPEG standard and software was available to produce .jpg files from scanner output. Also, Kodak started actively marketing its PhotoCD product. No experience was available in producing large image banks or in making the results searchable and retrievable across networks. The Internet of course was orders of magnitude slower than nowadays.

The project produced a beautiful collection of images from the treasures of the V&A and digitised the Topographical Historic Atlas of Noord Brabant. As all librarians who have cataloguing departments know, the production of metadata turned out to be expensive and time consuming in the case of the V&A were descriptions were not available electronically. Three user interfaces were produced, several Z39.50 libraries were explored and integrated and the results worked fine.

Once again, visitors were impressed and the EU evaluators were very positive. In Hollywood, a successful movie often implies a sequel, like Rocky-2 or HomeAlone-2. It is not know if there are relations between the EU and the movie industry, but anyway Elise inspired the Elise 2 project. It ran between 1996 and 1999 and aimed mainly at exploitable user services.

From the perspective of technology usage, the emergence of Java in the period before 1996 was relevant. Elise 2 tried to introduce Java at the client side of a browser connection and learned a lot about the subtle differences between Netscape Communicator and Internet Explorer and about version non-interoperability. These lessons were learned the hard way. Client side scripting without rigorous standardisation of implementations is something to really hate.

Another technological subject area that was explored in Elise 2 was multimedia documents. With the cooperation of a regional TV broadcasting company, Omroep Brabant TV, some 120 hours playing time of MPEG documents was produced from historic short amateur films. Real Players and Microsoft Media Players were considerably more primitive than they are now and Internet bandwidth was only just getting sufficient for streaming video.

2.3 DECOMATE AND DECOMATE 2

In 1995, while Elise was still unfolding Tilburg University started another EU project. It was motivated by the number of publishers that were willing to try it as test bed for electronic scientific journals and also by Kwik that started to be outdated in the web world. Together with London School of Economics and the Universitat Autònoma de Barcelona "DElivery of COpyrighted MATerial tot the Enduser" set out to write code.

The objective of the project was to establish highly configurable document servers in three countries with a free text search and full text retrieve interface that could also be adapted to local needs. Important components of the resulting servers were obtained in the form of freely available open source code. The design and the integration of the components were the work of the project partners. Most of the components implement functions that are also available in other products that are now available; they will be discussed in section 3 of this paper, while the glue between the components is discussed in section 4.

When Decomate finished, it was evaluated as excellent. The project partners decided to make the full source code and documentation freely available. However they also made clear in the texts and in presentation at the Decomate Conference that support was not included in the package. Libraries outside the project team refrained from installing the Decomate software.

Fortunately, the good old Hollywood trick worked again: Decomate 2 started in 1998 with the original participants plus the European University Institute and SilverPlatter. Implementing the personal Digital Library in Economics took until 2000. The result includes simultaneous searching in many databases across Europe, doing interesting things with XML. In early 2001 the Decomate owners contracted further development and commercial exploitation with Pica. The new name is iPort and Thomas Place presents it in another paper and another Ticer Summer School presentation.

2.4 CONSTANTS IN THESE PROJECTS

It is interesting to reflect on how Kwik, Elise and Decomate used technology to reach their goals.

It is clear that there are risks in applying quite new technology (in the examples Z39.50, JPEG, MPEG, Java, SGML and XML). It is also clear that staying with aging technology (e.g. Windows GUI programming) is more dangerous and much less fun. Clearly, libraries have to decide which of their concerns can best be handled by waiting for finished, proven products and which areas they want to tackle with innovative projects.

It is also clear that introducing new technology to end-users takes time and stamina. Fortunately you do not have to do everything yourself. Productivity is much greater, if you integrate building blocks supplied by others than when you take a 'not invented here' attitude.

Finally, it is strange to see how few dedicated competent people are required to implement new good ideas. The programming teams were always small; with five good people you can change the world.

3 COMPONENTS OF SERVERS

Pre print services come in different sizes. As long as the number of documents they are targeted for is small, say 25000 documents, anything goes. However when it gets interesting at ten times that size, scalability has to be included in the design from the outset. If any component of the server cannot be distributed across multiple servers, it is bound to be a future bottleneck. The only way to be able to distribute components is by having a modular design, with clean separations in functionality. This chapter describes these building blocks.

3.1 PLATFORM "WHAT SYSTEM SHOULD WE BUY?"

One of the least interesting questions about servers in general is actually asked frequently in Summer School evaluations: "on what type or brand of computer should you run a pre-print server?" The typical answer by an informatics specialist will be: "it depends". A more useful answer is that it depends on your environment and specifically on what 'platforms' are supported by in-house staff. The reason is that support staff is far more expensive than the server, and basically every organization has a less skilled support people than it would like to have.

Unfortunately, there is an exception to the rule. If the software you wish to run is not available for your locally supported computer platform, it can dictate the introduction of a type of machine and software that is new to the organisation. You should be really motivated to take such a decision however. At Tilburg University, ordering a server for streaming video content to end users was delayed for two years until it could be bought on a supported platform.

There is another problem associated, the challenge of supported but aging platforms. The question is how and when to start retiring old stuff. In the view of the author, organisations should review the list of supported platforms annually. If at that moment it is no longer clear that the platform will have a significant market share in say three years, further developments in that line should be frozen, so that staff can be retrained early enough, and change processes cause a minimum of pain.

Currently, there are only two main streams of servers, distinguished by the running operating system, a version of Unix or a version of Windows. There is no question that the most scalable and stable servers on the Internet run Unix, but small-scale Windows-based servers can be successful. Run only one service on a Windows server, to keep it stable. If you want to run an http server, run Apache and avoid IIS; it is junk. Apache is available for Unix as well as Windows.

Unix has more variants than Windows. Of course the support-decides-unless-software-dictates is true within the Unix world too. If a free choice can be made, the author's current order of preference is: Sun Solaris, Linux (Red Hat, SuSE), FreeBSD, IBM AIX, HP-UX and SGI Irix or SCO Unix.

3.2 STORAGE HARDWARE / SOFTWARE

A rather easy building block for a pre-print server is storage. Storage is used for storing documents, metadata, software and possibly search indices. The *size* of storage hardware is decided by the documents, but the disks that are involved in searching, not storing dominate the overall *speed*. The performance of library machines has always been decided by the number of fast disks and the even spread of disk accesses among them.

While large disk capacity is fairly cheap nowadays, backup facilities that can overnight make copies of all that disk capacity is rather expensive. At Tilburg University there is a trend to no longer want local copies of all documents. If they can be retrieved from a server at a publisher, why bother?

Software for storage is a bit more challenging than the hardware. Simply storing documents in a directory structure on some disk file system works surprisingly well if (and only if) the number of documents per directory is not too big and the documents are static. Big directories or high volatility destroy the directory caches of the file system. If documents are stored in a directory structure, you should generally not let that structure be reflected in URLs or other path descriptions: a mapping that dissociates physical storage structure from end-user addressing is important if you ever want to change the storage structure.

Once a document store gets truly big, simply storing in directory structures may become unmanageable. Above some dozens of Gigabytes of documents, the interface between the documents and the server should be a true database. Examples of this phenomenon are big mail servers. Big IMAP servers that store mail files in directories are orders of magnitude smaller than servers that store mail files in a database with appropriate access mechanisms.

3.3 SEARCH ENGINE

Another component that a pre-print server needs is a search engine. Free text searching traditionally is not something implemented by the big suppliers of relational databases, like Oracle. Even if the data structures are available, the tools for the management of large data volumes may be absent. One does not want to rebuild a complete database if a small problem occurs.

Tilburg University has during its document server projects encountered a number of free text database environments. It ran some packages itself, but implemented interfaces to other packages that were available at project partners as well. A clean programming interface (an API) to a free text database is of major importance. Of course simple speed is another criterion.

In the last years, Tilburg University Library itself ran production services with a product called 'Trip'. This product used to be very popular in Scandinavian libraries and met the speed/API/management-tool constraints that were imposed on it. Even when product support was discontinued after the supplier was taken over by another company, it ran without problems for years.

Recently however, Tilburg decided to move to a new environment. The Danish company Index Data offers Zebra, which is good and free for a university. The commercial version is called "Z'mbol", marketed by Fretwell-Downing Informatics. It has facilities not only for flat textual data but also for textual data structured as:

- Bibliographic MARC records;
- XML Data;
- Government Information Locator (GILS) records;
- World Wide Web documents;
- Encoded Archival Description (EAD) records;
- Dublin Core records;
- Mail archives and USENET news directories;

Index Data positions itself however mostly as a company that is an expert in Z39.50 This paper discussed that in section 4.1.

3.4 USER CLIENT

UNIVERSITY End-users need an environment to search and retrieve documents that reside on servers. Indeed, the Kwik project motioned in section 2.1 was done to make this environment by implementing client software for the Windows desktop platform that was at that time already perceived as "doomed to be successful".

When soon afterwards the WWW exploded, its browsers became the de-facto client tools for accessing documents. Many organisations including Tilburg Library made decisions that formalised the situation: services shall be web based unless there are very hard motives for other solutions. These decisions have turned out to be remarkably long-lived. There are no signs that anything can replace pre-loaded browser plus plugins; except newer pre-loaded browsers with newer plugin software. This has serious consequences for the digital document formats that libraries can reasonably use, or accept from publishers (see chapter 5).

As is widely known, the Microsoft Internet Explorer has taken over the leading role at the desktop from the Netscape Communicator. Fortunately, Microsoft is quite active in implementing XML in its browser, as is Mozilla that comes from the Netscape heritage. Opera has its dedicated user group who boast small size and speed but in the world at large few seem to care about those. Most people love 'features' more.

3.5 USER INTERFACE

Implementing a good user interface is relatively easy; designing it is very hard. One of the problems is that different people have different tastes, and every person is an expert on his own preferences.

Technical people tend to get rather tired of user interface discussions and pull back on statements like: "you tell me what you want and I'll make it" or "I have no taste". Graphical designers like to throw all restraints in the air. Loading the Tilburg University internal homepage involves opening some 50 connections and place heavy graphical elements at the left side of the screen. Apparently users of small screens are no target group (the author of this paper is an expert on his own preferences).

Search and retrieve specific user interfaces for non-specialist end-users might take into account what their users have always done. Even if the Dublin Core is a small subset of traditional catalogue title descriptions, users typically use words from a title, or they search the author name. In free text searches they go for a few keywords with an implied 'and' in between.

The far most important search and retrieve tool that reached public awareness in the last years is www.Google.com. Its user interface is very sparse, yet extremely intuitive, and it actually delivers. The interface for the Decomate iPort library environment has an equally simple initial search screen. It also feels natural. Perhaps the lesson is that user interfaces need to be as simple as possible for the targeted user group.

4 THE GLUE BETWEEN BUILDING BLOCKS

One of the reasons why building blocks exist is scalability. This was mentioned already in the introduction to chapter 3. Distribution of documents and search databases across multiple servers implies that these components need to communicate to each other. This means that network communication is required, and that clients and servers need to use data formats for sending and receiving messages.

Given the choice for web browsers with plugins as the user client, obviously http plays a role as a communications protocol. Http is a quite lightweight protocol, which imposes little structure on the transferred content. In principle, not only the transfer of retrieved documents with various Mime types to the end-user can be done over http. All communications that has to do with searching and inter-server communications could be done over http as well.

Yet, certainly in the library world, another specialised protocol is popular. It is Z39.50 and it played a fine role in many of the projects of the last years. It will be discussed in the next section. Z39.50 is a Search and Retrieve protocol. Yet, not all glue that connects the building blocks of servers is about document transfer or search and retrieve. Another function is that of traffic coordinator between servers. Certainly in situations where one user query leads to interaction with multiple search engines on multiple servers, such a traffic coordinator server is important. The iPort server knows such a server, that dispatches queries to other servers and that also implements the user interface.

Messaging among servers is not only in terms of Z39.50. The structure in message protocol can very well be modelled with the flexible, yet precise structures of XML. SGML and XML are discussed in chapter 5.

4.1 Z39.50

Z39.50 is not a lightweight protocol. In its full form it is complex and feature rich. An ISO set of standards for search and retrieve exists, The most important is ISO 10162, which was deemed really too complicated when people started writing code. That started the development of Z39.50 in the late 1980s. The first version was briefly successful, being implemented in WAIS. Like Gopher, WAIS became obsolete when the WWW started.

In 1992 version 2 of Z39.50 was defined. It is usually described as a 'clean superset' of ISO 10162 Search and Retrieve. In 1995 Z39.50 Version 3 replaced V2. It again was a superset.

The main functions that a Z39.50 client (or origin) can ask from a server (or target) reflect typical search patterns. After an *initialise*, a *search* is done. The result of a search is a so called result-set, but this set is typically not immediately sent back to the client. Instead the number of hits is returned, so the client can do new searches, possibly combining result sets with new narrowing or widening search terms. One or more calls to the *present* function retrieve subsets from the result-set, e.g. in groups of 20. The *delete-result-set* function does what its name implies. Result sets can live long. They exist to survive between subsequent interactions between client and server. Implementations can of course actually destroy every result-set immediately and recreate it with the following query, but the protocol was designed with a 'stateful', 'session oriented' approach in mind. This is unlike http connections to WWW servers, which are stateless one-trick ponies.

A quite useful function in Z39.50 is *explain*. It asks a server what data structures it contains. Unfortunately it is often not implemented (although it was available long ago in CMU's Mercury). One of the reasons why Z39.50 became popular with its users is that a client can speak only one search retrieve protocol, while the complexities of the search engines that live behind the Z39.50 target remain hidden.

This does not mean that there are no challenges with Z39.50 itself. One of the issues stems from the semantics of textual data. Library users like to search, say for authors. If the server has no concept of which part of its data is about the author, it cannot look specifically for a situation where 'van Tilburg' is an author instead of a reference to the city with that name. Library history has given us an extended list of catalogue formats, which indicate author fields. Many of these formats have been implemented in Z39.50 'record syntaxes' which are defined in so called profiles. Essentially all Marc formats exist; there is a complicated 'true' record syntax called GRS-1 for Generic Record Syntax. At the other end of the spectrum there is SUTRS, Simple Unstructured Text Record Syntax otherwise known as flat ASCII. Simply too many Z39.50 profiles exist. This variety makes interoperability difficult. An important development is that all implementations should support at least one profile called 'Batch' which is in the process of standardising it. The iPort implementation supports in principle all Marc formats, SUTRS, GRS-1 and XML. Batch will be there once it is stable.

The Z39.50 V3 version has done well for the past 6 years. In June 2001 a proposal called 'ZNG' for Z39.50 New Generation was formulated. For some the aim is to bring Z39.50 back to the desktop by simplifying the protocol and using http. Others concentrate on allowing 'soap (a kind of remote procedure call; heavily supported by Microsoft) as a transport. Early versions of the protocol called it "Z39.50 over XML" and although the name did not stick, it

is clear that Z is not dead. Good keywords to feed Google for finding out how ZNG is doing are: ZNG, ZIG and eZ3950.

Tilburg University uses Index Data YAZ; obviously it works nicely with Zebra/Z'mbol from Index Data.

5 FORMATS

Even without librarians inventing new ones, the world of documents knows many formats. This chapter explores where librarians should adopt existing document formats and where their contributions make a positive difference.

Anyway, document formats supplied to library end-users should keep our choice for the user client in mind, the WWW browser plus plugins. Publishers who supply documents to a library should be forced to use formats that fit in this scheme.

In the opinion of the author any document standard that is expressed in a non-textual binary form should be taken as it comes, from elsewhere in ICT. Textual information however (only slightly exaggerating) is either metadata, which is core business for librarians, or it can be peppered with metadata by tagging it with more information, making it metadata, or it is free text that librarians can throw search engines at.

The richness of textual data is nowadays the target of one of the most interesting developments of the last couple of years: XML. An overview is given in section 5.1. Binary documentation standards that librarians should simply use based on their position as a de-facto standard include GIF and JPEG for stills, PDF for printed pages packaged electronically, and mp3, MPEG and Real for song and dance.

One aspect of document standards is the size of typical documents. For a long time size mattered when disks were small and expensive. This phase has passed. Some people say that communications speeds still limit what documents can realistically be transported to the end-user. This is actually true in the third world were universities can be connected to the Internet with 64 kbps lines. Generally however it is no longer true. With cable-modems and xDSL already reaching homes and Fibre To The Curb (or Home) being planned in many countries, new Digital Library applications should no longer feel constrained by datacom speeds even if video documents fill DVD disks.

5.1 SGML AND XML

Historically, one of the first activities undertaken by Ticer has been a symposium on SGML, organised together with Elsevier Science. Elsevier was one of the early publishers who adopted SGML tagging to mark-up publications. As is common in an SGML environment they had developed their own set of tags, rigorously defined in a so-called Document Type Definition or DTD.

SGML is already quite old. Its roots can be traced back to a research project that started in the late sixties within IBM: GML was in 1969 called after Goldfarb, Mosher and Lorie. In 1986

standardisation led to “ISO 8879 Standard Generalized Markup Language (SGML)”. In the early years probably the biggest users were in aviation and the military. In the academic world TEI became well known, the Text Encoding Initiative which aimed at defining a DTD for natural languages.

In the mid-nineties, SGML was still something for specialists. Even if a growing number of libraries received metadata from a growing number of publishers, the average library user, or even the average library staff member could ignore the technology.

XML has changed the field rather dramatically. Actually XML is a form of SGML, so why is it hot? The reason is in the importance and the shortcomings of the World Wide Web.

The native document format of the web, HTML is a tagged language. In theory and by definition the tag structure of HTML is an instance of SGML, but in real life HTML is actually defined not by its DTD, but by the behaviour of web browsers like the Internet Explorer and the Netscape Communicator. Even the reaction of the various browser brands and versions on receiving well-formed HTML is different. The browsers unfortunately have always tolerated faulty HTML: they try to make the best of erroneous documents, motivating generators of HTML not to correct mistakes. The result is a mess, perhaps best characterized by infamous statements like: “This page is optimized for Internet Explorer 5.01”.

Not only the syntax of HTML has problems, the semantics is plagued too. Tags wild wildly different purposes are used next to each other. The standard example is to compare those tags that are about the structure of the HTML document (e.g. <p>)and those that are about formatting (e.g. <font>). When the problems with this aspect of HTML became already pretty big, while layout designers asked for more, more, more tags, the World Wide Web Consortium decided to step in. As a stopgap measure they developed a HTML specific stylesheet language called Cascading Style Sheets (CSS).

This development did not yet really solve the problem of validity checking of HTML. As a result, HTML remained very hard to maintain. Sets of linked HTML pages and external links to them had the problem of aging links.

All these problems had to be solved and the W3C, backed up by real programmers that actually produced working code set up XML and related other standards.

Now that XML was no longer something for specialists or academics, but a standardised environment with tools and implementations for everyone’s desktop, it also attracted other subject areas, not in the HTML replacement area. Currently, the author of this text sees new, real applications of XML at least monthly. They are as varied as log files (that used to be comma separated lists), configuration definitions of Cisco IP Phones, the visual layout of a PKI chipcard and a project that makes an inventory of administrative paper forms.

The evolution of XML goes very fast; new tools, new standards and new products outdate any book that was published before 2001 already in the autumn of 2001. A good current text is XML, The Complete Reference by Heather Williamson (2001, Osborne/McGraw-Hill). A better resource to track progress is the web, specifically http://www.w3.org/ Almost half of

the 45 working groups listed on this page are directly related to ongoing or finished XML topics.

Some important ones are:

- XML: a catchall group; note the link farms at “XML software” and “Bookmarks”.
- XSL and XSLT: the eXtensible Stylesheet Language and XSL Transform for transformations between XML and other formats.
- XPath: a language for addressing parts of an XML document.
- XLink: XML Linking Language, which allows elements to be inserted into XML documents in order to create and describe links between resources.
- MathML: foundation for the inclusion of mathematical expressions in Web pages.
- XHTML: suite of XML tag sets with a clean migration path from HTML 4.
- XML Schema: provides a means for defining the structure, content and semantics of XML documents as an upgrade from DTD.
- DOM: Document Object Model, an API interface that allows programs and scripts to dynamically access and update the content, structure and style of documents.
- SOAP: exchanging structured and typed XML messages between peers in a decentralized, distributed environment.

