Database backed Websites
D R A F T – Revision: 1.5

Thorbjørn Ravn Andersen

March 5, 2000

Abstract

This thesis discusses database backed websites; their current use as presentation engines, construction, pros and cons, list the current state of the art software, and how web sites in general function as *information presenters*. Hereafter it is discussed how end-users best can utilize such a system, without giving up their current software, allowing such websites to be used to ease the distribution of information between users, and let the web site provide *information sharing*.

Technologies are discussed with an emphasis on Open Source software, and two sample programs - "Cactus" and "Consensus" are presented, and discussed. Cactus allows any user to do automated SSP on an Intranet without a human webmaster to convert documents and maintain links. Consensus allows an open group of people to maintain a set of documents on a webserver.

Stuff to place: Java has unicode support, multithreads.
Note: Framed text is unexpanded keywords which eventually will be replaced with prose.

1 What is a database backed webserver?

A *databased backed webserver* is a webserver which has access to a database at the same time as it serves documents to clients. Such access opens new possibilities, some of which are:

- The database can protect against race conditions when data is updated on the server.
- Provide a searching facility more complex than grep.
- Allow fast access to data without worrying about limitations in file systems
- "users can extract data on their own if they want to"
- Databases are optimized for disk I/O, webservers for network I/O. Those go well together.
- Very large amount of data can be online
- Data can be managed remotely by others than the webmaster
- what else?

1.1 Protection against race conditions

A database can provide atomic operations on its data, as opposed to a stock Unix file system, where data unwittingly can be corrupted if two processes try to read and update information at the same time. The concept is well known from multi-programming the eskimo book?

Companies like "valueclick" and "doubleclick?" sell banner ad's (the 480x120? pixel advertisements allowing you to click through to the advertisors website if you are interested). These are either sold by the number of *impressions* (shown to a user) or *clicks* (where the user actually clicks on the banner), and these

must be counted in order to document that the customer gets what he pays for, and the banner ad provider does not expose more than what the customer paid for.

Here is a database system essential in order to protect the data from being accidentially overwritten. ValueClick states that they have 7 million impressions daily, which is around 80 impressions pr second.

And what else is new?

1.2 Provide searching facilities

A flat file basically only allow you to search the content and the filename with modifications based on the timestamps of the file. Searchings normally happen linearly through files and directories.

Since a database may have a lot more attributes, queries can be much more sophisticated, and may even be done against indexed data.

1.3 Allow fast access to data without using file systems

1.4 ????

1.5 Databases are optimized for disk I/O, webservers for network

I/O. Those go well together.

1.6 Very large amount of data can be online

1.7 Data can be managed remotely by others than the webmaster

1.8 Others?

Provide background information: Quick overview of the technology [webserver, database, http, sql, dynamically generated content vs] with a graph. Give simple example. Explain *when* things happen (on-the-fly vs statically generated pages). Explain the advantages and disadvantages databases have over flat filesystems [transactions(protection, atomicity), extra attributes, speed, indexed columns, complex queries, scalability(linear search in filesystems, multihost db's),

2 The new role of the webserver

Compare original functionality with static HTML-files with content and a few CGI-scripts, to the current dynamically generated sites with many hits pr day. Since HTML is generated and provides little abstraction it is hard to use on a higher abstraction level, and with unreadable code in ASP and PHP3 it is harder. CSS was not the answer since it did not provide any abstraction from the presentation (seperate content from layout).
Different needs of the user - wap/pdf/html/braille. CPU, storage, webmaster time prevents all formats being pregenerated.

3 Terms and concepts

"HTML is a SGML DTD"

This report uses a lot of abbreviations, many of which may not be widely known. This section lists mosts of them.

entity A named "unit" in XML/SGML which expands to a string. This is very similar to a macro without arguments in other languages.

FO Formatting objects. A generic description of the physical layout of a XML-document on paper. FOP converts this to PDF. Several users have expressed doubt about this XSLT transformation

SGML A family of languages. The *DTD* specifies which language it is. SGML can be formatted with FOSI or DSSSL style-sheets. "`jade`" can format to HTML, plain text and RTF.

XML A family of languages. The *DTD* specifies which language it is. XML-documents can be transformed with *XSL*-style sheets to another XML-document (this process is called *XSLT*). XML is a subset of *SGML*

***ML** Common description of both XML and SGML, where the two have the same applications.

DTD The Document T? Description is the specification which describes the exact syntax of the *XML

XSLT Either the process of *transforming* a XML-document into another XML-document using a XSL-stylesheet, or the process of *formatting* a document into FO, which then can be further formatted in PDF.

DOM W3C's initial model for representing an XML tree internally. Superceeded by SAX (for java?)

SAX Simple Application interface for? XML. An event based approach to XML-parsing and representation. Has an advantage in that the design allows processing to begin before the whole XML-tree has been read in. (URL?). Parsers and processors? which implement SAX can be selected freely, currently allowing for 20 different combinations of parsers and whashallicallem.

HTML Hyper Text Markup Language. The "language of the web" which was originally designed by a physicist to present articles to other physicists. Was later hacked upon by Netscape and Microsoft to do things it was never originally meant to do.

DSSSL SGML style sheets something. Is used to render an SGML document to another format. "`jade`" can render to FOT, RTF, TEX (must be post-processed with "`jadetex`"), SGML and XML. Is this a standard?

FOSI Another style sheet for SGML, which I do not know anything about. Apparently the US Navy uses it a lot. Check in DocBook.

SQL The Structured Query Language is the standard language for communicating with a database.

XSP `XML Servlet Pages` - A technique for combining code with XML in a single page, which is parsed and executed by the webserver when it is requested.

Explain XML, XSL, XSLT, HTML (yes!), SQL, SAX, DOM, XSP and whatever fancy terms come to mind. Draw graph of what happens with an XML document.

4 The consequence - multiple views of a document

Please read the "Terms and Concepts" before continuing.

Introduce publishing where HTML is just a backend amongst many (PDF, Word, ASCII).
Describe why it is important to be able to render finished versions of documents fully automatically from a single *annotated* source (www/print/cdrom/Palm Pilot/braille/handicapped persons,whatever). The better the annotation, the better the output (ref: Stibo). Describe the need for SGML (history/usage) and XML (why/browser support/on-the-fly publishing/bleeding edge - being standardized).
Donald Knuth - TEX/Web/Literate Programming - advantages (high quality, excellent math, superb algorithms, can be tailored to needs (basic interpreter written in TeX) and disadvantages (programming language, not abstract) (designed 20 years ago). Ask Steffen Enni about his thoughts. Javadoc. Perl POD. DocBook projects (FreeBSD, LDP).

5 The user should use current tools to publish documents

Publishing a document to the web should be as easy as printing a document. The basic principles are the same - you can do with a "Publish"-button and a dialogue box where the essential information is provided. It is not so today - discuss reasons . Use "print to fax" software as horrible example of this idea gone wrong.
When the webserver is dumb, you cannot do much. If there is a database underneath, much more is possible. Discuss the idea of having several ways of entering documents in the database (upload via form, send as email (fax software can do this too), print to virtual printer, scan, fax, voice) and letting the software do the conversions.
Use example with print PostScript to file and upload via ftp to printer (LexMark).
Writing XML directly is much harder to do than writing HTML (stricter syntax, more options, plainly just more to type), and should be aided by a good tool. Alternatively, the user should use well-known tools (Word) and mark up according to very strict rules, which is then automatically converted to the XML document. This does not give as rich documents, but allows users to publish existing documents with very little trouble.

6 The need for well-defined standards

The need for open, well-defined standards. HTML, PDF, PNG (versus GIF) RFC-things (SMTP, POP3, IMAP, MIME) good examples. Word counterexample (look for Bill Gates saying people can buy an upgrade to read Word97 files in Word95). Adobe has good documentation on their PostScript and PDF document formats (doing theirs to have it well supp orted). HTML is widely documented in RFC.

7 Why is Open Source essential?

Independence from hardware and software vendors, allowing free choice of hardware platform and operating system, as well as the possibility to fix bugs and use other products. (Examples: XT versus Xerces, Java Interpreter for Linux, servlet technologies, cheap - use money for hardware instead of licenses).

8 Sample websites

slashdot.org (examine code - ask developer), Politiken, DynaWeb (SGI - oooold check on fyn, when was it first developed?), valueclick (banners - ask Ask for full story. What are they running), php3 documentation online. photo.net (reread). Sun's docs.sun.com (DocBook SGML)
Talk about standard search engines on web pages.

8.1 slashdot.org - high volume information site for nerds

Describe setup. Describe flow, and the slashdot effect. What hardware/software.

8.2 Valueclick.com

Banner provider. Interview Ask for details, about how the NT-cluster was replaced with a small Linuxbox with MySQL and a squid in http-accelleration mode. Explain the problems with having fat processes serving slow modems.

8.3 Politiken

Talk with Lars. Politiken genrates their pages on the fly.

8.4 Amazon - an Internet bookstore

Figure 1: The splash screen for Amazon

Figure 2: The splash screen for Amazon

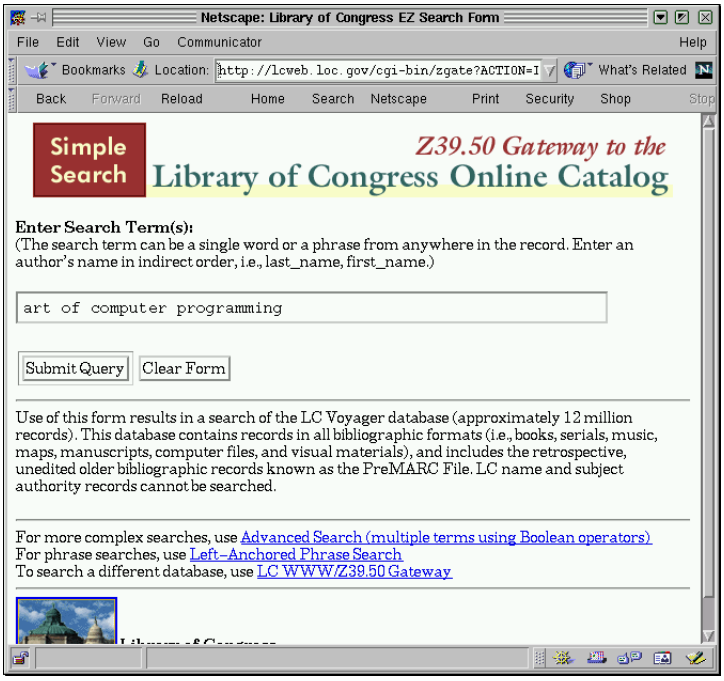

Figure 3: The splash screen for Amazon

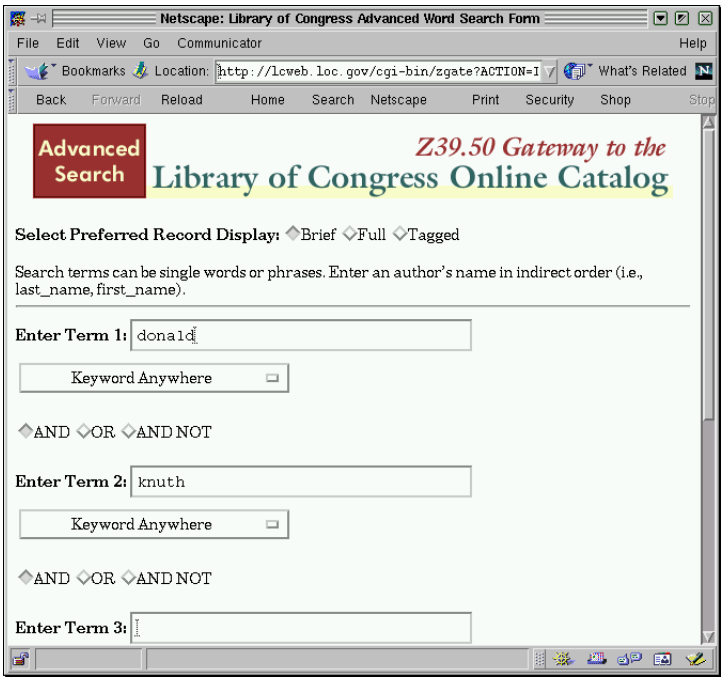

Figure 4: The splash screen for Amazon

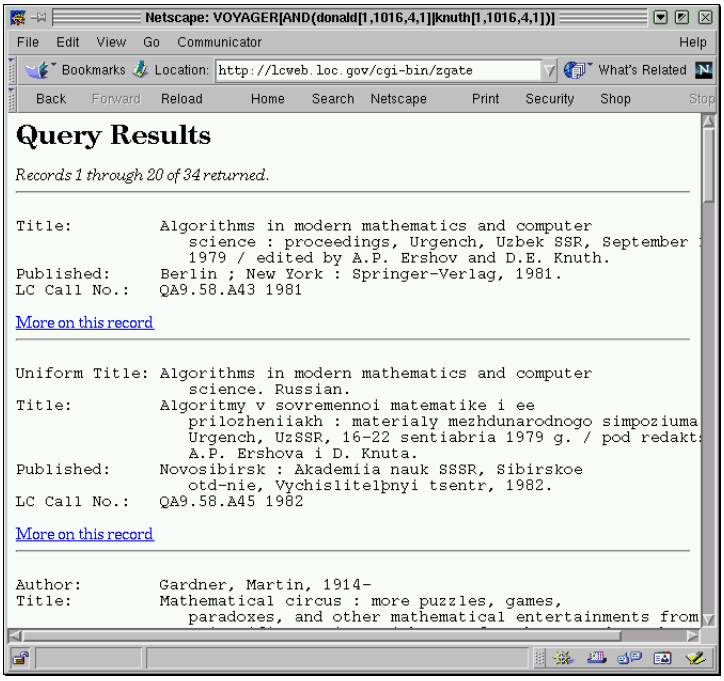

Figure 5: The splash screen for Amazon

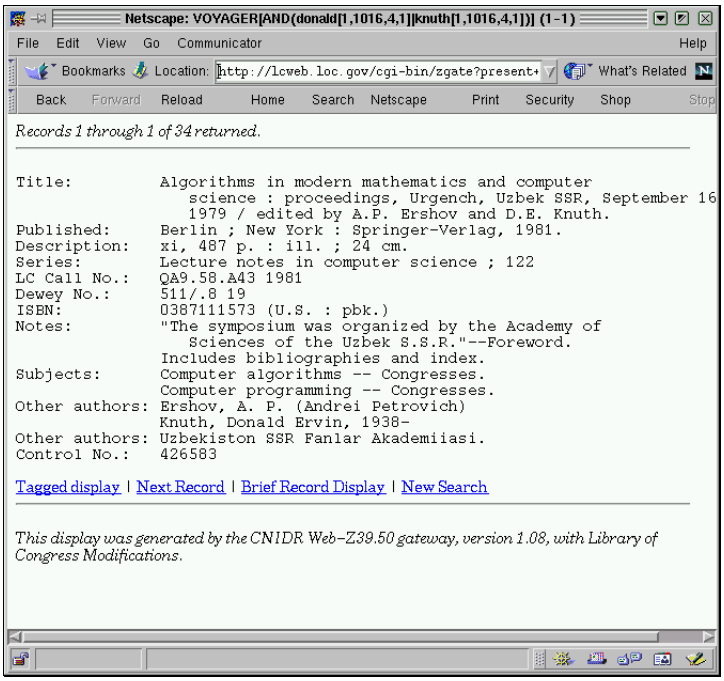

Figure 6: The splash screen for Amazon

8.5 Congress

8.6 www.krak.dk

The Danish map manufacturer http://www.krak.dkKrak A/S has a reputation of being the definite guide on maps, especially on driving maps in Copenhagen. Their website have been up for about two years now, and have been steadily improved with new facilities and cross-references. The start page is shown in figure 7, and is seperated in two parts; namely white pages (phone number based), and pink pages (company based).

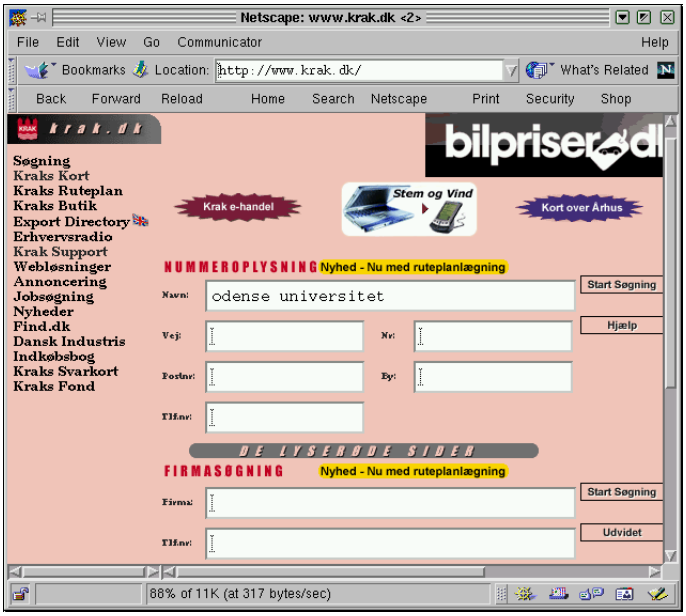

Figure 7: Welcome page (and search form) for www.krak.dk

The fields in the white page section are Name, Road, House number, Zip code, City and Phonenumber. The fields in the pink page section are Company and Phonenumber. The form has been filled out with a search request for “Odense Universitet”, and the results are shown in figure 8 on the next page.

Each line containing an answer may have icons referencing to facilities regarding the location of the answer. These answers have pointers to Rejseplanen, the Route Planner, and a map reference. Additionally pointers to a web page, and an email address could have been provided by the users. Clicking the map icon of the first line actually referring to Odense University brings us to figure 9 on the following page.

The blue dot usually indicates the location of the address in question, but in this case it is a bit off (the University is located 500 meters further to the south). When a map is display, the navigation area below the image allows for zooming and moving the contents of the shown area. By selecting “Zoom out” and “x8” a new map is shown (see figure 10 on page 11), where it is evident that the database show a great deal of detail. Roads, residential areas, streams, train stations, the motor way and green areas are shown. These maps provide a level of information corresponding to Krak’s printed maps.

Now go back to the list of results from the search for “Odense University”. The Car-icon gives a route to the listed location, where you must fill in the source yourself. Figure 11 on page 11 show the form filled in with my own address and the address of the university, and “Route on map” (Rute på kort) gives a visual route between the two locations.

This map is shown in figure 12 on page 12, and is perfectly adequate for a person driving a car. For cyclists it is often a good idea to look for short-cuts, if you are well known in the area.

If you need step-by-step directions, Krak can provide that too. Select the “???” button and print out the directions.

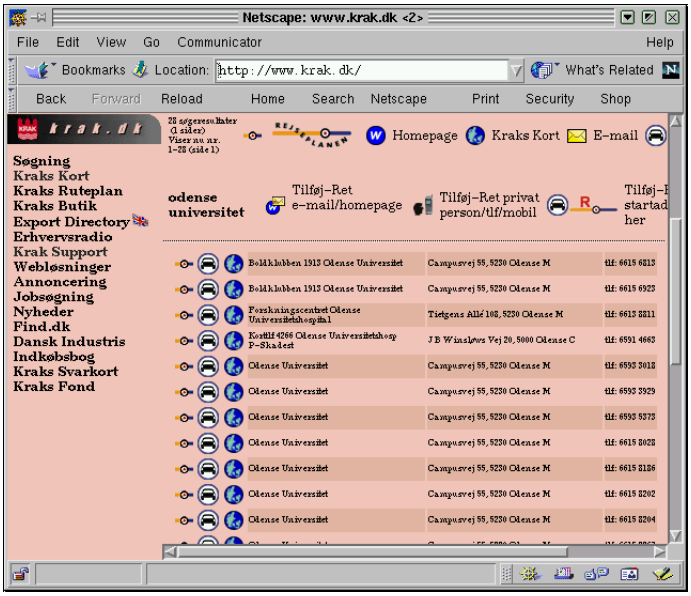

Figure 8: Result of searching for “Odense University”

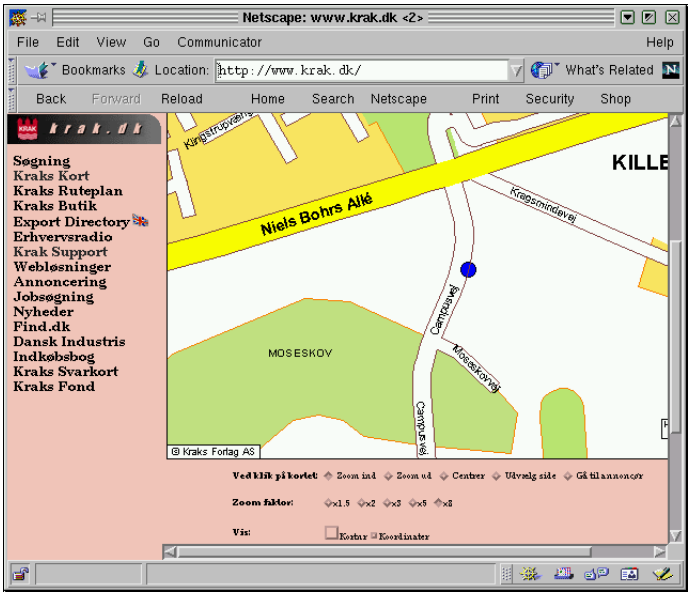

Figure 9: Following the map icon for the first “Odense University” reference

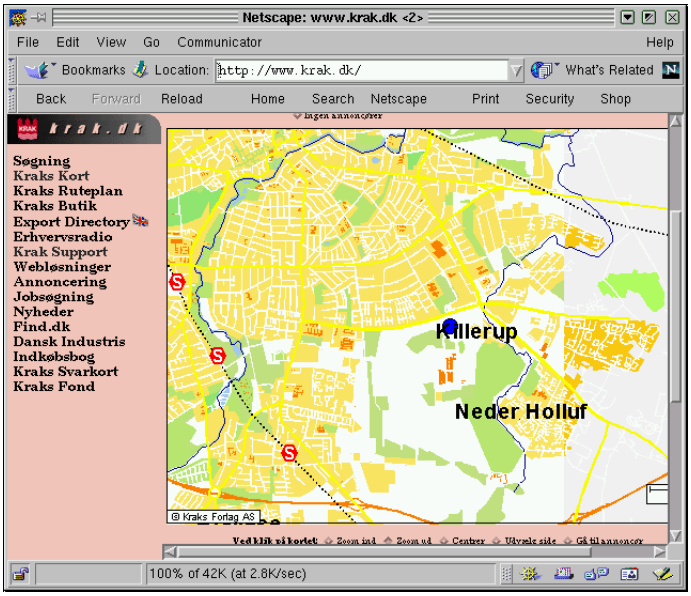

Figure 10: The map in figure 9 on the preceding page zoomed out with a factor 8

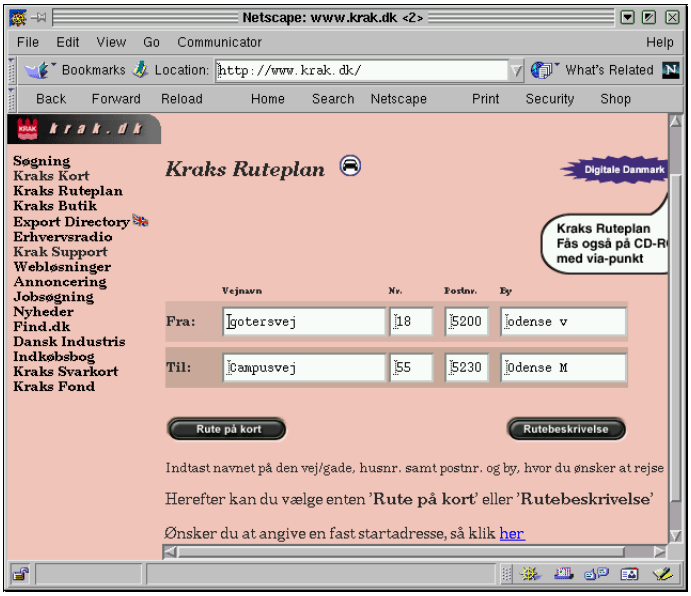

Figure 11: ????????????????????? University" reference

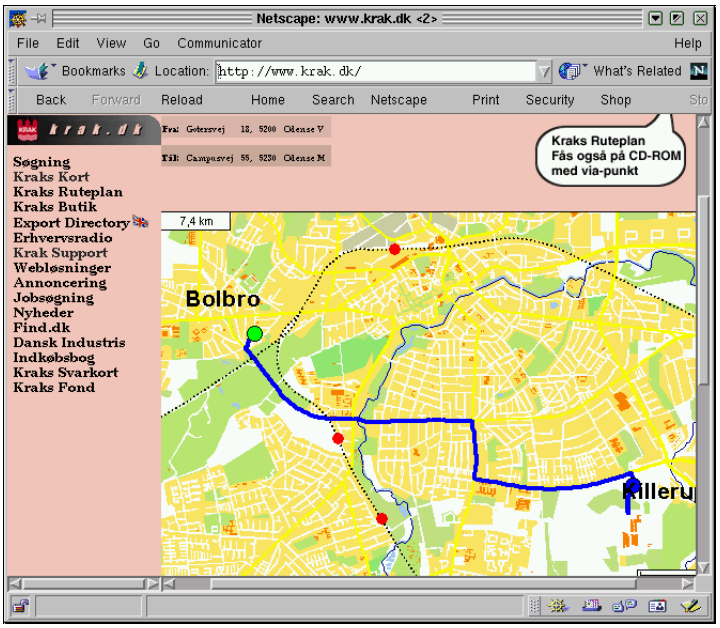

Figure 12: ????????????????????? University" reference

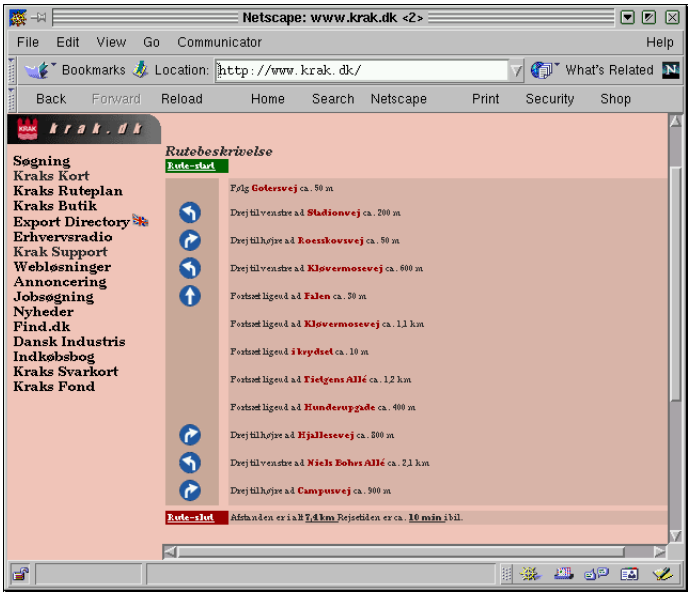

Figure 13: ????????????????????? University" reference

Necessary to do rendering according to requests. Cannot be predicted reasonably. What have been done? How does it work? What about

8.7 rejseplanen.dk

8.8 www.rejseplanen.dk

8.9 Google/Altavista

Search engines. How do they do it? www.909.dk, www.krak.dk, terraserver.microsoft.com. Encyclopedia Brittanica.
ASP+Access, tinderbox & bugzilla, javasoft (developers area), LXR+Bonsai. "Alle danskere paa nettet" - Dansk Journalistforbund, Pressemeddelelse. Greenspun selv?

8.10 The Collection of Computer Science Bibliographies

http://liinwww.ira.uka.de/bibliography/index.htmlhttp://liinwww.ira.uka.de/bibliography/index.html

8.11 Cactus – document capture and conversion

Two out of four methods are implemented (email, printer). (Increase blob limit in DBI::MySQL).
Sample PDF viewer, perhaps simple XLS viewer.
Demonstrate that Cocoon can provide the navigational framework.

9 Overview of technology available for Linux February 2000

Describe what products there are currently available for these categories. Evaluate as much as possible. Conclude that currently this is an area in great development.

9.1 Webservers

Apache, misc Servlet Javaservers (jetty, thttpd, java webserver, etc). Look for comparisons. Roxen (graphs). Discuss whether several webservers should be run on the same machine to provide better services.

9.2 Database engines

Full SQL servers: MySQL/mSql/DB2/Oracle (spank their FLUG policy)/Informix/PostgresSQL (discuss features: transactions, unique numbers, in-kernel code, speed, availablity, multi-host capabilities?). Any flat file RDBS?
Tricks: Menu system in LDAP (lr.dk).

9.3 Browsers

MSIE5 (XML support, Solaris), Mozilla (alpha), Netscape (too old), HotJava 3.0 (no XML, can be expanded to do XML conversion internally?), Opera (?), Amaya (other W3C browsers?), AWTviewer in fop (can it do xml directly?),

9.4 XML utilities

XT, Xerces, Cocoon, SQL2xml, validators (error recovering?), converters to XML (pod2docbook, tex4h, Perl with appropriate modules?, wordview modified?
Show the XML-¿SQL-¿XML-¿HTML on the fly conversion possible with Cocoon .... Without auxillary code. All done in W3C style.

9.5 PDF utilities

PDF generated by pdflatex is good. PDF generated by fop isn't. createpdf.adobe.com
pdfzone. acroread (ok, heavy), xpdf (font problems with bitmaps, ugly but fast, gives best result with usepackagetimes, no anti-alias), gv (pretty good). Distiller (check what it can do). Use pdfinfo to get the number of pages in the pdffile.
Conclude that it is possible to have a 100% Java solution for on-the-fly publishing XML to HTML and PDF through a webserver. Discuss why developers do it in Java, as opposed to any other language.
Very few utilites support creating PDF. Discuss how to create PDF.

10 SGML, XML and DocBook

"HTML is a SGML DTD". Tim Berners-Lee 1991. ??URLWhere was this said?

Looking from the DocBook DTD's this section describes XML, how it was developed from SGML inspired by the short comings of HTML, and how it is used,' including viewing, authoring, and conversions (transforms, formatting).

10.1 The need for separation between content, layout and ?? rewrite?

WYSIWYG tools often cause the author to concentrate on the visual layout instead of the document content. Seperating the writing process from the layout process, results in documents which can be visualized in many different ways depending on the actual needs. If the content describe the layout too, it is hard to change the layout without revising the text.

The better annotation a document has, the more possible ways it can be visualised. Blind people may require braille and aural versions of a document (possibly in parallel ARTICLE ABOUT MULTIPLE VOICES SPACIALLY SEPERATED), and mobile phone users may require short, crisp headlines with most text omitted. If a document is properly annotated it can be visualised effectively for these and other users.

!! The needs of blind people are very different from the needs of mobile phone users, yet

10.2 The design and evolution of SGML

Look up SGML history when/where/whom/WHY. What does SGML do well? Why is it so? Initially hard to write in. Development of DTD's to use in applications, with Stylesheets (FOSI, DSSSL (when)). Development of tools for making writing easier (complex, expensive, slow). Use liking to printer drivers for printers, which was sweeped away by PostScript for LaserPrinters (bitmaps are too hard to handle - talk about Stibo rendering in 2400 dpi to guarantee idential prints).

10.3 The creation of HTML

Look up creation of HTML on the net. Look for reasons why HTML was chosen to be a SGML DTD instead of e.g. Windows Help, or something else.

HTML: 1.0 (original, simple, <hr> tag was added due to popular request), 2.0, 3.0 never made it, 3.2 tables, 4.0 w3c straightens up things. Netscape added new tags *ad hoc* often. In the mean time HTML has been made to do things it was never meant to do originally, for presentation purposes (large imagemaps in tables without any textual information), and there is still a need for new features that designers want. *W3C saw that blindly adding new features to HTML would result in an even more bloated standard* with a lot of backward compatability to maintain - a modern browser must still be compatible with the bugs in Netscape versions 2 and 3. These were renowned for being very lenient towards users providing erroneous HTML, and have started a tradition of browsers doing their best to interpret what the users *might* have meant. This makes it hard to use HTML as a generic information data format, since parsers are complex due to the many exceptions.

The W3C took a bold step in saying: We need a new format - the eXtensible Markup Language. XML!

10.4 The creation of XML

XML was as HTML built on SGML, but with a lot stricter syntax than HTML and a lot less facilities than SGML. XML is a new standard - it was made a W3C recommandation on ??????1999, and the accompanying XSLT and XPath recommandations?...

What was the design goals

The resulting language for designing languages has these features:

- A very strict syntax which every XML-document must comply to (which is what a *well-formed* document does). All open tags must be closed explicitely. Tag attribute values must be in quotes.
- Easy to parse and generate. The strict syntax makes the parser simple.
- The DTD is not mandatory. DTD-less documents does not have a DTD to conform to, and must only be well-formed. Unicode?

TRANSFORMATON
FORMATTING

10.5 XHTML - XML compliant HTML

http://www.w3.org/TR/xhtml1/The XHTML W3C recommandation. http://www.w3.org

A big problem with XML is that it cannot be viewed by anything the majority of users have today. It is possible, however, to specify a form of XML called *XHTML* which bridges the gap between the two worlds of XML and HTML. Doocuments corresponding to a XHTML DTD are parsable by HTML-4.0 compliant browsers if a few, simple guidelines are followed:

- Elements (the HTML tags) must be closed correctly.
- Attribute values must be quoted
- Element names and attributes must be given in lower case
- Empty elements must either have an end tag, or the start tag must end with "/>". I.e. a horizontal line is written as "<hr />", where the space is important to make this acceptable to HTML-4.0 parsers.
- More stuff... Netvwork lag

The well-formedness of an XHTML document is achieved by using tricks like:

```
HTML: <hr noshade>
XML:  <hr noshade="" />
```

Why is this smart???

XHTML is an XML-variant designed to be viewable in HTML-4.0 compatible browsers (without XML support). XHTML-documents are well-formed XML documents, while at the same time being valid HTML-4.0, so that XHTML can be the target of a standard XML-transformation as well as a source for initial XML-processing.

This is very different from the usual situation, namely that HTML is the result from *formatting* a document, where the result is not XML any more.

Note: XHTML style sheet generated files does not trigger correct change to UTF-7 characterset.

WHICH CONVERSION UTILITIES ARE THERE? TIDY? READ XHTML? WRITE XHTML? WHAT DO W3C SUGGEST? WHY DO THEY THINK THIS IS A GOOD IDEA

10.6 What DTD should be used?

The Document Type Definition (*DTD*) defines the way a SGML/XML document should look in order to mechanism which defines , which specifies exactly

- which tags are valid
- which tags can appear at any given part of a document
- which attributes are valid for a given tag
- which *entities* (macros) can appear in a document, both for expansion strings but also for Unicode characters unavailable to the document author

- Others?

There is a lot of different DTD's available to many different purposes, since basically everywhere a datafile is needed XML can be used with a suitable DTD. ChemML, SVG?. If a need arises it is just a matter of creating a suitable DTD to work with it.

Unfortunately, for each and every pair of DTD and output format an XSL-style sheet must be written, tested and maintained. Therefore it is a very good idea to use a commonly used, documented, and well-thoughtout DTD for the documents, and several other groups have already done such work.

The Cactus system uses the DocBook XML V3.1.7 DTD with the DocBook XSL ————————-

10.7 How can a *ML document be viewed?

In order to be viewed, a *style sheet* must be applied to the document. Style sheets convert the tags in the document to a given output format, and depends on (DTD, output format pairs).

GRAPH DEPICTING STYLE SHEETS BEING APPLIED FOR DIFFERENT OUTPUT FORMATS

Commecial products usually have a viewer which can apply the style sheet directly `examples?` `Panorama?` `Framemaker+SGML`, but are due to licensing costs and supported user base rarely an option for general for documents targeting end-users. Table 1 discusses the current *de facto* document formats on the Internet.

available: Discuss this in a earlier chapter? "As discussed in chapter foo..."

Format	
HTML	Browsers are available for any modern platform. HTML-4.0 compliant browsers are Netscape Navigator 4, Internet Explorer 4... More?
PDF	Adobe Acrobat Reader is available for most mainstream platforms URL. The Adobe Acrobat Viewer is available for any platform with Java URL. The Open Source project Ghostscript URL is available for these and other platforms.
Microsoft Word	The `doc` and `rtf` file formats are widely used, but needs a full word processor to format properly. Anything else but Microsoft Word gives inferior results, and Word only runs on Windows.

Table 1: Document distribution format

The Microsoft Internet Explorer 5.5 (not yet available at the time of writing) will be able to transform XML documents with XSL-style sheets internally complying with the W3C recommendation. The previous version of IE was available on Windows, Solaris and HPUX.

The Mozilla browser in its pending release as Netscape Navigator 5.0 will not be able to apply XSL-style sheets, but only CSS cascading style sheets. This decision is rather old and may have changed.

For the immediate future web publishing in XML implies a need for server side conversion to other formats. The available Open Source software for this is surprisingly small, but that is rapidly changing. Please note that XML style sheets still implies SGML style sheets, which is interesting because these are more mature due to longer use.

10.7.1 SGML style sheets

CONFIRM THE BEHAVIOUR WRITTEN BELOW IS EXCACT

SGML style sheets are written in DSSSL (others?), which is a Lisp-dialect, and the usual Lisp-conventions apply. Several

several? what can this do?

The DocBook reference recommends using `jade` written by James Clark, to do DSSSL conversions. long processing times later implemented in XT with threads to allow faster output.

Show a sample stylesheet

10.7.2 XML style sheets

XSL - the XML style sheets - are written in XML too. The tags in the name space "xsl:" describe what is to be done with the source XML tree, in terms of tag-remapping (<para> to <p>), sorting of subtrees, and several programming constructions like "if", "while" and "foreach". New tags can be constructed, existing tags can be altered, and yes?

This style sheet creates an outline of a DocBook document by looking for headline tags, and make a tree of them Eh? Line breaks have been introduced for readability. Line numbers in this listing would be nice. What about colour encoding? Can it be done?

```
<!--
| Identity Transform Stylesheet
| ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
| $Author: ravn $
| $Date: 2000/03/02 21:55:31 $
| $Source: /home/ravn/CVS/SPECIALE/SPECIALE/x-outline.xsl,v $
| $Revision: 1.1.1.1 $
+-->
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                version='1.0'>

<xsl:output method="html"/>

<xsl:template match="/">
  <html>
  <body bgcolor="white">
  <xsl:apply-templates/>
  </body>
  </html>
</xsl:template>

<xsl:template
match="set|book|part|reference|article|sect1|sect2|sect3|sect4|sect5|section|
       simplesect|preface|chapter|appendix|example|figure|table|informaltable">
  <xsl:param name="treeicon">n</xsl:param>
  <xsl:variable name="level" select="count(ancestor-or-self::*)"/>
  <xsl:variable name="title" select="title"/>
  <xsl:variable name="this"  select="local-name(.)"/>
  <img src="s.gif" border="0" height="1" width="{22*$level}"/>
  <img border="0" src="{concat($treeicon,'.gif')}"/>
  <img border="0" src="{concat($this,'.gif')}"/>
  <xsl:variable name="color">
    <xsl:choose>
      <xsl:when test="$level = 1">navy</xsl:when>
      <xsl:when test="$level = 2">red</xsl:when>
      <xsl:when test="$level = 3">blue</xsl:when>
      <xsl:when test="$level = 4">black</xsl:when>
    </xsl:choose>
  </xsl:variable>
  <span style="color:{$color}"><xsl:value-of select="$title"/></span>
  <xsl:if test="local-name(.)='example' and programlisting/@role">
   <small style="font-family:courier">(<xsl:value-of
       select="substring-after(programlisting/@role,'-')"/>)</small>
  </xsl:if>
  <br />
  <xsl:apply-templates/>
</xsl:template>
```

```
<xsl:template match="*|@*|comment()|processing-instruction()|text()"/>

</xsl:stylesheet>
```

Since the DocBook reference is very vague on this matter (the book went to press before the XSLT recommendation was finalized) the Internet has been a great help. Table 2 lists the tested XSLT processors which conforms to the W3C XSLT-19991102 recommendation. emphasis on java due to platform indenpendance

Where should I talk about SAX?

XT	XSLT processor based on the XP parser. Both written in Java by James Clark.
Xalan	XSLT processor written in Java originally based on the LotusXSL processor by alphaworks?. Currenly under heavy development by the Cocoon team (`xml.apache.org`).

Table 2: Leading XSLT processors implementing XSLT-19991102 in February 2000

During testing it showed very quickly that the Java tools are quick enough to use (do a few benchmarks), even though they are not yet tuned for optimal performance. Additionally

Even so, implementations in C should only be considered by designers if Java clearly cannot do the job. The advantages of Java over C (reference big list, probably at JavaSoft?) are

Xalan is at the time of this writing in a development state, and contains bugs which causes it to crash on some of my sample DocBook XML document. After that I turned to XT which is extremely stable and reliable, and that I have used for the rest of the project - my only complaint is that it stops after reporting the first error instead of parsing the whole document.

When the Cocoon project stabilizes they will have a high end XSLT-engine embedded as a servlet in Apache, which makes this a project to watch. The technology is at the time of this writing still immature, but very promising.

The platform independance of Java was indirectly prompting the implementation of remote jobs? in Cactus. Very early in the evaluation Xalan threw the above mentioned exception when processing a medium sized DocBook document. In order to rule out errors in the underlying Java environment on Asserballe (see section B.1 on page 31 for technical details) an identical run was made with JDK 1.2 on Aalborg. The same error occured, but in less than one second instead of ten. Experiments showed that Java programs consistently runs 15 times faster on Aalborg than on Asserballe. and so what?

`incoming.xml`

```
<?xml version="1.0"?>
<?xml-stylesheet href="incoming.xsl" type="text/xsl"?>

<?cocoon-process type="sql"?>
<?cocoon-process type="xslt"?>

<page>

 <connectiondefs>
  <connection name="foo_connection">
   <driver>org.gjt.mm.mysql.Driver</driver>
   <dburl>jdbc:mysql://localhost/Cactus</dburl>
   <username>XXXXXXXXX</username>
   <password>XXXXXXXXX</password>
  </connection>
 </connectiondefs>

 <query connection="foo_connection" tag-case="lower">
```

```
  select when, mime, user, how, filename, comment,length(item) as l  from incoming
 </query>

Testing
</page>
```

incoming.xsl

```
<?xml version="1.0" ?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<?cocoon-process type="xslt"?>

<xsl:template match="page">
<html>
<head>
<title><xsl:value-of select="title"/></title>
</head>
<body bgcolor="#ffffff">
<img src="gr/organ_pipe_cactus-single.jpg" align="left" />
<xsl:apply-templates/>
</body>
</html>
</xsl:template>

<xsl:template match="ROWSET">
<table border="0" bgcolor="#F0F0F0">
<tr><th>when</th><th>mime</th><th>user</th><th>how</th><th>filename</th><th>comment</th><th>iteml</t
<xsl:apply-templates/>
</table>
</xsl:template>


<xsl:template match="ROW">
<tr>
<td><xsl:value-of select="when"/></td>
<td><xsl:value-of select="mime"/></td>
<td><xsl:value-of select="user"/></td>
<td><xsl:value-of select="how"/></td>
<td><xsl:value-of select="filename"/></td>
<td><xsl:value-of select="comment"/></td>
<td><xsl:value-of select="l"/></td>
</tr>
</xsl:template>
</xsl:stylesheet>
```

10.8 Converting documents to XML

XML is currently having the same problems that HTML had in its initial years, namely lack of software support. The difference is that HTML is so lenient that it is easy to write HTML by hand in a text editor. Not so with XML.

The difference this time is that Microsoft has been an active participant in the definition of XML, and has made ??? about Windows 2000 supporting XML directly in what precisely?

Even so there is still a severe lack of conversion tools for the many files out there in non-SGML based formats like Word, Excel, LaTeX, TeX, Lotus Notes, etc., which must be available before the documents can be converted. Table

Majix	Java based RTF to XML, which I have adapted to DocBook
pod2docbook	Converts Perl documentation to XML
wordviewothers sp?	Converts Word 6-95-97 to HTML. I have adapted them to produce DocBook output better do it too
tex4h	package for LaTeX which can what was it now?
others	??

Table 3: Conversion tools to XML [February 2000]

Due to the rather inflexible DTD-requirements in XML, it is most convenient that such a tool generates stand-alone XML (SGML ok if it can be converted? With what?), which can be processed freely in the system. This is hopefully something which gets resolved soon, since the validation of the tree structure in an XML document is as important as it being well-formed. rewrite.

Move filter descriptions to Cactus chapter discussing filters?

10.8.1 Majix - RTF/(DOC) to XML

Majix URL, company description is an RTF (for Word 97 documents) to XML converter. I have created a configuration and a small style-sheet which can render a RTF file to DocBook XML.

If run with the Microsoft Java Machine ("`jview`") while logged into a Windows machine with access to the GUI, it can launch Word to convert a given DOC file to RTF and process it, with very reasonable results. I have experimented with making this accessible to Linux by installing Microsoft Services for Unix on a NT machine with Word installed, and using telnet to invoke Majix. Unfortunately this doesn't work. Another approaches (which will be tested if time allows) includes having a pseudo-user watching a directory on a NT-server and launching Majix whenever DOC or RTF documents arive. These can at best be described as kludges.

Even though it shows great promise, it has been a year without new releases (including the promised Pro version), and the source is not available.

Conclusion: The Majix software is currently best for personal end-user conversions.

10.8.2 pod2docbook - POD to XML

The POD format is created for writing documentation for the Perl programming language, and was created to be "...an idiot- proof common source for nroff, TeX, and other markup languages, as used for online documentation" by Larry Wall (the perlpod manual page).

The original release produced SGML DocBook. I have submitted patches to the author which allows the user to choose between XML and SGML DocBook.

Conclusion: This do a good job for documents written in POD, and will probably with time take over as the default formatter for printed versions of Perl documentation.

10.8.3 tex4h - convert TeX to XML

What daelen does it do

Conclusion:

10.8.4 wordview with friends

Parses binary stream and produces HTML, WML plus more.

10.9 Formatting HTML on the fly

look at docs.sun.com - what daelen do they do? Render SGML to HTML on the fly?

When the decision has been made to provide information in *ML on the web server, another decision must be made. Should the various renderings to PDF and HTML be created before they are requested, or when the user asks for a certain rendering?

Look for an article on static versus dynamic

As always it depends on the nature of the data.

Static renderings is probably suitable for documents which rarely change their rendering or take long time to render, and which just must be served as fast as possible. Often there is a desire for an inexpensive solution[1], and in that case a standard Apache with a rich set of prerendered documents in a flat file system may very well be optimal. Personal experience has shown this to be a very robust solution which very rarely requires human intervention. esr Eric S. Raymond reports that a modern PC with Apache is easily capable of saturating a 10Mbps ethernet connection.

Dynamic rendering is suitable whenever the data changes very often, or there is a wish for the user to be able to personalize their view of the presented documents (should frames be used or not? do the user want large or small versions of images?), or if the conversion process is light and

The history and usage of SGML. Creation of XML. Describe document validation, and conversion (XSLT) to XML and other formats. Freedom from restrictions of HTML. Problems with tons and tons of DTD's. The need for well-documented standard, robust, supported DTD's (current ones: TEI, ebook [buh], DocBook). HTML conversion utilities can be tailored to generated DocBook XML for SSP (currently pod2docbook, Excel xls2xml). Found that SGML (jade) is powerful but too slow for on-the-fly stuff, as opposed to XML renderer. Several to choose from if they conform to the w3c standards (DOM and SAX). Who uses DocBook at the moment? Man pages in DocBook on Solaris (look on machine). IE50 cannot show "simple" DocBook XML yet but it is the goal of that project. the xsl/docbook/contrib/outline/outline.xsl is an excellent sample of a small, powerful style sheet.

[1] "Inexpensive" here is defined in terms of man hours needed to learn and maintain the web server, as well as the hardware needed

11 DocBook considerations

DocBook is a DTD designed for documenting software. Used by Sun, FreeBSD, O'Reilly and others.

The SGML is converted to HTML by applying a HTML style sheet for DocBook, and sending it through jade with request for a sgml conversion. The stylesheet generates a HTML file pr section in a chapter which is rather too much. The HTML is _very_ ugly - all induced line breaks are _within_ the tags to avoid introducing any artificial whitespace. The HTML is Lynx-compatible.

The generated navigation tags are a little troublesome in Lynx, but nice in Netscape. A "up", "next" and "previous" are available combined with a "Top title"->"Chapter title" -> "Section title" navigation bar at the top.

The localization code is manually maintained.

12 Can users transparently harness the power of XML for webpublishing?

Discuss how users can use their current tools to publish information to the web (Cactus principles: data capture and automatic conversion) by automatic conversion to XML and on-the-fly conversion on the way out of the webserver (while still having access to the original file provided by the user). This also allows people to view files made with software products not available to them.

13 Active and passive data collection for a webserver

[just thoughts] User can "push" data to the system, or the system can actively watch a resource (directory, webpage, usenet server, "channels") and update whenever new data is available. www.mind-it.com. RDF-format from netscape to push headlines.

14 Search engines and datamining

htdig (process website) swish (process files for website) are ok. With database you can ask the database directly (currently in blobs) but with more analysis of the files you can do better. "Which author wrote this back in 1948".

15 Conclusion

Databases are great, yea! Webservers are great, yea! Things must be powerful and easy for best results.

A Sample implementation – Cactus

Abstract

The Cactus system fills a gap in the current integration of the web with normal office procedures, as it provides easy web publishing for occasional authors, by letting them submit documents in several ways with their usual software.

The system automatically produces other versions of the documents on demand of the users reading the documents, as well as provide the navigational framework, relieving the local webmaster from these tedious and error prone tasks.

A.1 System description

Cactus is a sample implementation of the following issues:

- **easy publishing** - users can publish via email/fax/print/watch usenet/www which is processed into the SQL database, and confirmed.
- **automatic conversion** - system will automatically convert a given document to what the user can see (or want). Browser sends a capability string with each request - use that to provide stuff directly, or generate a "this is available" summary.
- **automated navigational framework** - each document has an annotation which tells Cactus where to place it in the navigational hierachy. The corresponding navigational pages are automatically generated and updated when new documents arrive. This also ensures system integrity without "broken links".

Implementation languages - possibly Java or Perl. Perl chosen due to better library support (with source). Rex thingie in Java. How can MIME, TAR, etc be done in Perl (remember jubilations!). Using Apache with Cocoon (perhaps) and MySQL. Graph algoritms in [Sed90].
Good summary from 19991114. Speed of PNG gzipping? PNG uncompressed intiially and then later compressed by pngcrunch.

Utilities: mswordview, xls2xml

A.2 Background

Explain the history. Yggdrasil to address the need of an automatic webmaster → analysis (seperate file written earlier). Cactus to address the difficulties of publishing information to Yggdrasil.

Explain Cactus as a successor to Yggdrasil.

The Yggdrasil system was designed to provide a simple, consistent navigational framework around a dynamic set of web pages provided by users, as well as providing a "recently changed pages"-list plus an overview of pages written by each person. This was a very successful idea in the start, but gradually lost momentum due to these factors:

The development platform was changed from Unix to NT, which made it much more difficult for the users to access their web-directories, as these were no more a part of their home directory[2]

The internal document format became Microsoft Word, which at that time could not be converted to HTML. Such documents were therefore unable to be published.

The users could trigger an update by sending an empty email to the system, as well as rely on an automatic nightly update. The trigger mechanism was not brought along when the email system was converted to NT.

New users was not informed about the system.

[2]The Apache webserver uses the /public_html directory for the users personal web pages. The transition to NT meant that the users - in addition to their normal file manipulations - should telnet to a Unix server, and change the file attributes every time the file was updated.

A.3 Installation

Perlmodules. MySQL server. SAMBA.

[Rephrase following paragraph]

I have concluded the following goals for CACTUS, in order to avoid repeating history:

1. ”Ease of publishing” is crucial.
2. Document preparations and transformations must be fully automatic.
3. The organisation using the system must be fully doing so.

A.3.1 ”Ease of publishing” is crucial

CACTUS uses two approaches in order to make publishing as easy as possible for the users, namely

Documents are accepted in their native format, and in numerous ways.

Discussions with potential users showed [...] to be realistic ways in which CACTUS would be used:

As a document storage for various versions of a document, being developed and emailed back and forth between authors and peers. By allowing CACTUS to accept documents as email attachments, it would be very easy to enter each draft in CACTUS by including it on the list of authors or peers. In this way the archival of the document in CACTUS is completely transparent. As a fax machine. When replacing a fax machine with a computer, it is very easy to send a copy of the temporary image of the fax to CACTUS, before printing it. The fax system is then enhanced with the possibilities of the web, relieving the need for the physical copy. As a printer. Cactus provides a ”printer”, which makes a web-version out of any document the users can print. Users can then share final versions of documents without requiring the recipients to have the software in question. Either the Adobe Acrobat Reader can be used, or the primitive multiple image viewer in Cactus.

The full list of the publishing methods in Cactus is listed [....]

The users were primarily expecting to use these file formats:

- Word DOC and RTF files. These are the storage formats of the Microsoft word processor Word.
- HTML files. The common format for the web.
- LaTeX files. This typesetting program is very popular with mathematically oriented academics.
- GIF, JPEG, TIFF and PNG images. These and many more are in common use. Cactus use PNG internally [except possibly for JPEG]. The full list of supported formats is listed in the implementation section [see somewhere].
- PostScript and PDF files.
- Fax images.

A.3.2 Document preparation and conversion must be fully automatic

There was a strong consensus amongst the users, that it would be very nice not to have to convert the documents manually every time they were to publish a document. Therefore Cactus accepts several document formats as described above, and abandons the requirement that the users should convert their documents to HTML before publishing.

The system have some very different views on the documents depending on which representation the user needs - not all make sense for all documents:

1. The unaltered original. This file is always available, allowing users with the correct software to continue working with it.

2. A normalised version of a document. This could be a PNG version of a TIFF image or BMP image which can be viewed by all modern browsers, and an XML version of a document. If a suitable encoding can be found, even images and sound can be represented in XML so that this does not overlap the textual representation.

3. A visual representation of the original. This is the ”look of the file”, i.e. as it would look when printed to paper, and allows users without the corresponding software to view and print the contents.

4. A textual representation of the original. This is the ”meaning of the file”, which could say that the line ”foo bar” is a level 2 heading, providing search capability.

5. An audio representation of a ”document”. This is e.g. an message left on an answering machine.

Conversions between all these document representation must be automated as much as possible. [write about the Cactus framework for providing existing and future filter types]. The normalised document is the only one created when a document enters Cactus - the rest are created on demand to avoid overfilling the underlying database, but cached for a reasonable time to improve performance.

[....]

The organisation using the system must be doing fully so.[rephrase]

In order for such a system to be successfully used within an organisation, it is vital that the organisation is using it whole-heartedly. [and more of the same].

Note on derivers:

A deriver is a pipeline which derives another version of an item. E.g. PNG to GIF,

There are the following tasks:

Derivers - derive another version of a given item. In theory, this is a reversible operation. These should generally be designed to be as fast as possible, since these will be called from CGI-programs with users waiting. Avoid compression (gzip should maximally be level 1).

Converters - create another item from one or more originals. This could be a multi-part MIME file which should be assembled into a fresh original. It could be a DVI file from several source files (tex files and images). These should generally be reasonably fast, since their speed specifies how fast a new item can be made available to the system.

Optimisers - A deriver is usually designed for being usable in a real-time interactive setting (also called being fast), which normally means that the result is sub-optimal. An optimiser complements this by doing a suitable optimisation step whenever the system is sufficiently idle. This could be running ”pngcrunch” on PNG files (which yields 30% on Ghostscript output), or ”tifftotiff” on TIFF files. The idea is that the file is still the same basic type, and can be used without further modification. It is an ”in-place optimisation”. Some file formats are using the gzip compression scheme internally. These would not benefit further from archiving.

Archivers - creates a long-term storage version of an item. This would typically be running ”gzip -9” on the content, creating a new entry (with a new mimetype), and marking the item as decachable. The dearchiving process must be fast, since it might be needed by a deriver without notice.

Validators [spelling] - validates whether the content of the item is consistent with the MIME-type, and is it conforming to the standard. (Can a gzip-stream be decompressed? is a PostScript file parsing correctly? etc.). If a type is unknown, guess MIME-type from filename and/or contents, and validate it. If all fails, assign a type of application/octet-stream.

Examiner - looks into an item to look for references and encoded data. A reference may be an URL or an emailaddress in a signature. Encoded data could be an uuencoded image in a Usenet posting, or a pointer to a usenet article.

These are complemented by

Acquirers - gathers items from ”outside” Cactus and enters it in the ”incoming” table with an appropriate MIME-type. An acquirer could accept email, emulate a printer, retrieve Usenet articles, etc.

Janitors - cleans up whenever the system is otherwise idle, or when the cache is full. Derived items can be emptied of content, originals can be archived. Expired items can be purged completely from the database, along with all their derived items.

Presenters - extracts data, and present them. This could be a CGI-script presenting a given item as a http-stream. A PDF item could be presented as a window with two frames, the leftmost containing a thumbnail pr page, and the rightmost a high resolution version of a given page. The page should be cut in smaller pieces to allow easier processing by Unix Netscape.

In an ideal world we have infinite storage and infinite CPU-speed, meaning that everything would be done instantly when we need it. In order to decide the order in which to do tasks, a price system must be developed which would guarantee that the system processes every item, that the system is still responsive while converting,

What to do when the system runs so full that archiving cannot be done fast enough. Can data be moved to ”outside storage?”.

A.4 The implementation of Cactus.

Goals:

- Stable, well-known and remotely administrative platform- Linux/Solaris
- Platform independence - the resulting system must not be tied to Unix
- Modular - in order to minimise actual development, software libraries should be used as much as possible.
- Extensible - it should be easy for the system administrator to enhance functionality.

Choosing an implementation language:

Since platform independence was important for the system, it was quickly found that reasonable choices would include scripting languages plus Java. Scripting languages are fully interpreted allowing a script to run on numerous platforms without change. Java is the only compiled language with this property, due to the ”Write once, Run everywhere” philosophy from Sun, plus the tight integration with Internet technologies in Java.

Initially I wanted to write the core of Cactus in Java, and spent a couple of weeks evaluating the language but found that

The mentality of the Java-community on the Internet, is very influenced by the shareware philosophy typical for the PC-user. Everything useful cost money, source code is not revealed, and the general tendency is for large, stand-alone applications.

Nobody had written a publicly available, stand-alone MIME-parser in Java. Several RFC’s should be implemented and tested, in order to get email-parsing in Cactus.

The general level of abstraction is - in my opinion - too low in Java, while the ”core language” is enormous.

[?]

Information extraction from items.

A given document contains one or more items, which again may contain further information. Cactus uses the MIME-type of a given item, to select the information scanning method with a fall back to the generic application/octet-stream examiner.

The application/octet-stream is parsed for:

The text is scanned for URL?s, either with the Tom Christensen urlify or the program posted or commented by Abigail. These include ftp, http, gopher, mailto and news references. These are stored as external references.

The text/plain is parsed for

uuencoded data in a text stream. These start with ?begin ### name?, contain lines matching ####, and ends with ?end?. Such a file is then assigned a validated MIME-type based on the name of the file, and entered in the system as a derived item.

An text/html item is parsed for:

normal references in <a>-anchors. While doing this, the text to be rendered is extraced and parsed as a text/plain stream, in order to get the sequence right (may be changed). The <meta> tags are examined in order to extract keywords for the label.

A.5 Converters

A.5.1 Microsoft Word

The Microsoft Word DOC-format is widely used, but apparently so hard to use that even Microsoft cannot do it reference to Word97 not being able to create Word95 files, and the reason why Cactus was started in the first place. I have spent a great deal of time looking for suitable software which could have been used with Cactus on the server side, and then testing it out.

Microsoft Word - The best solution would be able to actually run Word itself, but apparently this is integrated so well with Windows that it cannot run on other systems like WINE (URL). Since Microsoft previously have deliberately made the Windows 3.1 version of Internet Explorer 4 unable to run in the WinOS/2 subsystem for OS/2, I strongly suspect that this is also the case here.

Viewer for Microsoft Word - the Word Viewer is also available for 16 bit versions of Windows, which behave better in WINE. A full Windows installation was not available to me when I tested this - this might have improved. This can be used for producing the PostScript printouts.

Corel WordPerfect 8 for Linux - This is generally a very nice word processor, but the import filter for Word crashed WP when I tried it with a large document. Filters should be better in WordPerfect 9, which is due for Linux medio 2000.

StarOffice 5.1 - the German office suite for several platforms have excellent filters for importing Office files in general, but cannot be automated from the commandline. A StarBasic program must be written and invoked to do the job ask on newsgroups.. This is currently untested, but probably the way to go. Since I did the testing, Sun have bought StarOffice - most likely since they need an network based office suite for Java, which StarOffice provide with thin clients and a Solaris based server - and promise to make the source generally available. This has not happened yet, but it is currently the most likely alternative to the Microsoft Office solution.

AbiWord - An Open Source word processor which shows great promise. It was not able to parse my test documents in Word.

mswordview Others? an Open Source Word to HTML converter which currently do text well, but have trouble with graphs which are converted to the WMF format.

Do-it-myself - If you ask Microsoft very nicely, you can get a copy of the Microsoft Office Binary File Format Specification, and write a personal parser of Word. It took Microsoft 6 months to answer my email request, and then they just sent me the license twice with no specification. Second time they got it right. Unfortunately, the license was so restrictive that I decided not even to *look* at the specification.

Majix - This small RTF to XML converter can also parse DOC files, by invoking Word to save the DOC file as RTF. This shows great promise, except that this mechanism requires the GUI to be available for Word (it needs to pop up the window). The ingenious scenario of a telnet server on a NT-machine, which accepted remote logins from Cactus, started Majix which again started Word, did not work. Word hangs. Additionally the company has not released a new version in a year, and the license explicitely prohibits disassembly.

....

My intermediate solution has been to adapt the mswordview to output DocBook XML instead of HTML, which provides a text-only display.Yes?

A.5.2 TEX and LATEX

A.6 MySQL

On asserballe

```
create table incoming (when datetime, mime varchar(80), user
  varchar(80), how varchar(80), filename varchar(255), comment varchar(80),
  item longblob);
```

```
+----------+--------------+------+-----+---------+-------+
| Field    | Type         | Null | Key | Default | Extra |
+----------+--------------+------+-----+---------+-------+
| when     | datetime     | YES  |     | NULL    |       |
| mime     | varchar(80)  | YES  |     | NULL    |       |
| user     | varchar(80)  | YES  |     | NULL    |       |
| how      | varchar(80)  | YES  |     | NULL    |       |
| filename | varchar(255) | YES  |     | NULL    |       |
| comment  | varchar(80)  | YES  |     | NULL    |       |
| item     | longblob     | YES  |     | NULL    |       |
+----------+--------------+------+-----+---------+-------+
7 rows in set (0.01 sec)
```

B Available hardware and software

This section lists the hardware I have used during development of Cactus.

B.1 Asserballe

Urls in long banes for this.

- Pentium-S 133 MHz PC with 128 Mb RAM and 10 Gb IDE disk
- Linux Redhat 6.1
- MySQL 2.3.?? - security updates
- Apache 1.?? with mod_perl, ApacheJServ, Cocoon.
- IBM JDK 1.1.8.
- CGI, DBI::??? perl modules for writing CGI scripts and accessing MySQL databases.

The choice of JDK was made at a time when top bug at Javasoft was "please provide a port of Java to Linux", and where IBM beat Sun to it by porting their Java 1.1.6 implementation to Linux. The benchmarks showed it to be as fast as their Windows and OS/2 ports, which at that time was the fastest Just In Time compilers on the PC-platform. It performs very well on Linux, and the installed version is just a maintainance upgrade.

MySQL and the corresponding JDBC driver *must* be the latest stable versions due to security and bug fixes.

B.2 Aalborg

(4-CPU UltraSparc 440MHz with Sun JDK 1.2)

C Bibliography with Web references

Full reference with a lot of annotated URL's.

References

[Sed90] Robert Sedgewick. *Algorithms in C*. Addison-Wesley, 1990. ISBN 0-201-51425-7. 25

Contents

1 What is a database backed webserver? 1
- 1.1 Protection against race conditions . . . 1
- 1.2 Provide searching facilities . . . 2
- 1.3 Allow fast access to data without using file systems . . . 2
- 1.4 ???? . . . 2
- 1.5 Databases are optimized for disk I/O, webservers for network . . . 2
- 1.6 Very large amount of data can be online . . . 2
- 1.7 Data can be managed remotely by others than the webmaster . . . 2
- 1.8 Others? . . . 2

2 The new role of the webserver 2

3 Terms and concepts 3

4 The consequence - multiple views of a document 4

5 The user should use current tools to publish documents 4

6 The need for well-defined standards 4

7 Why is Open Source essential? 4

8 Sample websites 5
- 8.1 slashdot.org - high volume information site for nerds . . . 5
- 8.2 Valueclick.com . . . 5
- 8.3 Politiken . . . 5
- 8.4 Amazon - an Internet bookstore . . . 5
- 8.5 Congress . . . 5
- 8.6 www.krak.dk . . . 9
- 8.7 rejseplanen.dk . . . 13
- 8.8 www.rejseplanen.dk . . . 13
- 8.9 Google/Altavista . . . 13
- 8.10 The Collection of Computer Science Bibliographies . . . 13
- 8.11 Cactus – document capture and conversion . . . 13

9 Overview of technology available for Linux February 2000 14
- 9.1 Webservers . . . 14
- 9.2 Database engines . . . 14
- 9.3 Browsers . . . 14
- 9.4 XML utilities . . . 14
- 9.5 PDF utilities . . . 14

10 SGML, XML and DocBook **15**
10.1 The need for separation between content, layout and ?? rewrite? . . . 15
10.2 The design and evolution of SGML . . . 15
10.3 The creation of HTML . . . 15
10.4 The creation of XML . . . 15
10.5 XHTML - XML compliant HTML . . . 16
10.6 What DTD should be used? . . . 16
10.7 How can a *ML document be viewed? . . . 17
10.7.1 SGML style sheets . . . 17
10.7.2 XML style sheets . . . 18
10.8 Converting documents to XML . . . 20
10.8.1 Majix - RTF/(DOC) to XML . . . 21
10.8.2 pod2docbook - POD to XML . . . 21
10.8.3 tex4h - convert TEX to XML . . . 21
10.8.4 wordview with friends . . . 21
10.9 Formatting HTML on the fly . . . 21

11 DocBook considerations **23**

12 Can users transparently harness the power of XML for webpublishing? **24**

13 Active and passive data collection for a webserver **24**

14 Search engines and datamining **24**

15 Conclusion **24**

A Sample implementation – Cactus **25**
A.1 System description . . . 25
A.2 Background . . . 25
A.3 Installation . . . 26
A.3.1 ”Ease of publishing” is crucial . . . 26
A.3.2 Document preparation and conversion must be fully automatic . . . 26
A.4 The implementation of Cactus. . . . 28
A.5 Converters . . . 29
A.5.1 Microsoft Word . . . 29
A.5.2 TEX and LATEX . . . 30
A.6 MySQL . . . 30

B Available hardware and software **31**
B.1 Asserballe . . . 31
B.2 Aalborg . . . 31

C Bibliography with Web references **32**