Database backed Websites
DRAFT – Revision: 1.16

Thorbjørn Ravn Andersen

March 20, 2000

Dette dokument er under udarbejdelse. Al tekst i dette skriftsnit er kommentarer til mig selv som jeg skal kigge på senere, og skal derfor vurderes derefter.

0.1 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*.

as well as list my path to interest for meta-data and content-descriptive languages.

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.

Chapter 1

Overview

This section is rudimentary – it must be written at the end. Currently there is just keywords.

In this thesis I refer to the author Annie, and the layouter Larry.

1.1 What is a "database backed webserver"?

A web-server is a program which hands out files to any web-browser that asks.

A database "is a spread-sheet that several people can update simultaneously" (greenspun).

A database backed webserver then is simply a web-server which knows how to talk to a database server while servicing bypassing web-browsers.

1.2 Why use a webserver in combination with a database?

In one word: *Synergy*!

The web browser provide a simple, efficient and well known user interface available to almost every computer user in the world, and the database provide efficient dynamical access to data.

The combination of the two allow for the personalized Internet where a given webserver can give any user customized information upon request. Examples are:

- Newspapers - the user may specify that she wants in-depth coverage of international affairs, and do not want sports, and her personal web-edition will then just contain that. Payment may be per-article instead of per-paper.
- Maps/Airlines/Hotels - the user may request the best way to travel from A to B at a given time, see the route, and order accompanying reservations on-line.
- Real stores - the user may browse and order from the inventory. The ordered goods are then delivered by mail or similar.

- Virtual stores - the user may browse and order from a virtual inventory. The order is then dissected and forwarded to the appropriate suppliers to the virtual store.
- Banks - there is a growing need from customers to be able to do home banking. If the bank offer a browser based solution, they can support customers regardless of their choice of computer. The transactions done through such a system will most likely be stored in a database to avoid dataloss.
- Program development - The Mozilla project[1] has shown the benefit of up-to-date information regarding just about anything related to the source code. write about it

Since its inception in 1993 the web has grown to be the "good enough for most things" graphical user interface, capable of showing text and graphics as well as provide navigation. Due to the simplicity of HTML it is not hard to build a plain basic client[2] meaning that just about any modern computer with Internet capabilites have browsers available for them.

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),]

1.3 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.

Documents must be written in a format describing *content* and not layout.

[1] ...Mozilla project...– http://www.mozilla.org

[2] Plan basic client implies no Java, no Javascript, no cookies, no plugins, no cascading style sheets, no HTTP-1.1 and almost everything else that characterize the modern browsers from Netscape, Microsoft, Suns and Opera??. However, if a webpage is well-written it can still be displayed on such a plain basic client.

Chapter 2

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.

The list will be sorted

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.

RTF *Rich Text Format* - a document description language designed by microsoft. Widely supported. May contain extensions to the original RTF-specification.

Jade A DSSSL-engine for SGML documents by James Clark.

JPEG An efficient method for representing photographs in computer files. Efficiency is achieved by discarding parts of the visual information which is hardest for the human eye to see.

GIF An image format well suited for computer generated images with few distinct colors, like icons. Is hampered by a patent on the compression scheme used and a maximum of 256 colors. Superceeded by PNG.

PNG The successor to GIF. Is supported in newer browsers only.

CGI The Common Gateway Interface. The original way to generate pages and other files dynamically. A CGI-script can be written in any language supported by the web server.

IIS Microsoft Internet Information Server. The webserver from Microsoft.

ftp File Transfer Protocol. This is both the name of the *protocol* (the method) as well as the *program* implementing the protocol. ftp can transfer files between a client computer and a server computer, and is a standard on the Internet

http Hyper Text Transfer Protocol. This is the protocol a web browser needs to speak with a web server, in order to retrieve documents. Http is intentionally very simple.

TCP/IP This is the procotol which a computer needs to speak to connect to the Internet. It allows two computer to establish a data stream, where one machine pushes bytes in one end of the stream, and the other machine retrieves the bytes from the other end of the stream, without either computer being concerned about the underlying network.

SP A what?

Perl A scripting language which has become popular with Unix system administrators, and which "was there" when a need for scripting languages for CGI arose. Has eminent regular expressions.

ASP Microsofts version of a scripting language intertwined with HTML. Works on Microsoft web servers only.

JSP Java Server Pages. Suns version of Java intertwined with HTML. Requires a servlet-capable web server.

PHP The Internet version of a scripting language intertwined with HTML. Can run as a CGI script in most browsers, but as a module (which is faster) in at least Apache.

Apache Open Source webserver which runs on a lot of different platforms. Most Linux distributions include it.

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.

Chapter 3

Dynamically generated documents

> *"404 Document not found"*
> A – B

!!! PUT SAMPLE OF FILENAME INSTEAD OF TT-FILENAME-/TT in SOMEWHERE.

3.1 The way it started: Handmade web-pages in a filesystem on a web-server

Back in the old days when the world wide web was designed, a web server could basically do only two things:

- Serve *static* files directly from an underlying file system. These files were either pages written in HTML, or images in GIF or JPEG formats.
- Call a program complying to the *Common Gateway Interface* with user-submitted parameters, and return its output as the document to send.

That was all!

> Such web servers still exist - for example the NCSA server[1] (which has been unmaintained since 1996. NCSA recommend that the Apache server should be used instead). Even so the February 2000 Netcraft survey[2] reported that 17172 NCSA servers were active, and could be reached from Netcraft.
>
> In August 1995 the number of webservers running the NSCA server was 10835[3], meaning that number has increased slightly in that period. For comparison the total number of public webservers on the Internet in the same period grew from 19 thousand to 11 million, with Apache and Microsoft Internet Information Server accounting for 8.8 million of these.

Even with such a simple model, it works well for even *very* large web-sites which only generates few documents dynamically.

[1] ...the NCSA server...– `http://hoohoo.ncsa.uiuc.edu/`

[2] ...the February 2000 Netcraft survey...– `http://www.netcraft.com/Survey/Reports/0002/`

[3] ...webservers running the NSCA server was 10835...– `http://www.netcraft.com/Survey/Reports/9508/ALL/`

A well-tuned server which serves documents directly from the filesystem can reach impressive numbers. Where was this - look for statistics. On a modern PC Apache has no problem saturating a 10Mbps Ethernet connection (what about a 100Mbps connection), meaning that even for very large and demanding sites the bottleneck will be determined by the hardware![4]

On a related note: The ftp-server `ftp.cdrom.com` serves a lot of files every day. On March 14 2000 the transfer statistics[5] said that the average output that day was 79.3 Mbit/s (with a peak of 95.8 Mbit/s). The average output on a yearly basis was 86.2 Mbit/s, which is 933 Gb/day in average). This single machine runs FreeBSD[6] (see www.freebsd.org[7] for details). Since an ftp-session have a notably larger overhead than an http-session, it is not unreasonable to expect similar performance from a suitably tuned web-server when serving static files.

CGI-scripts is another matter. For the occasional dynamically generated document CGI-scripts turned out to work well. All kinds of documents – images, webpages, progress reports – could be generated comparatively easy in the favorit language of the CGI-script author. Libraries to deal with the decoding of parameters, and encoding of the generated result were soon abundant, providing for many enhancements to the original "static web site" model.

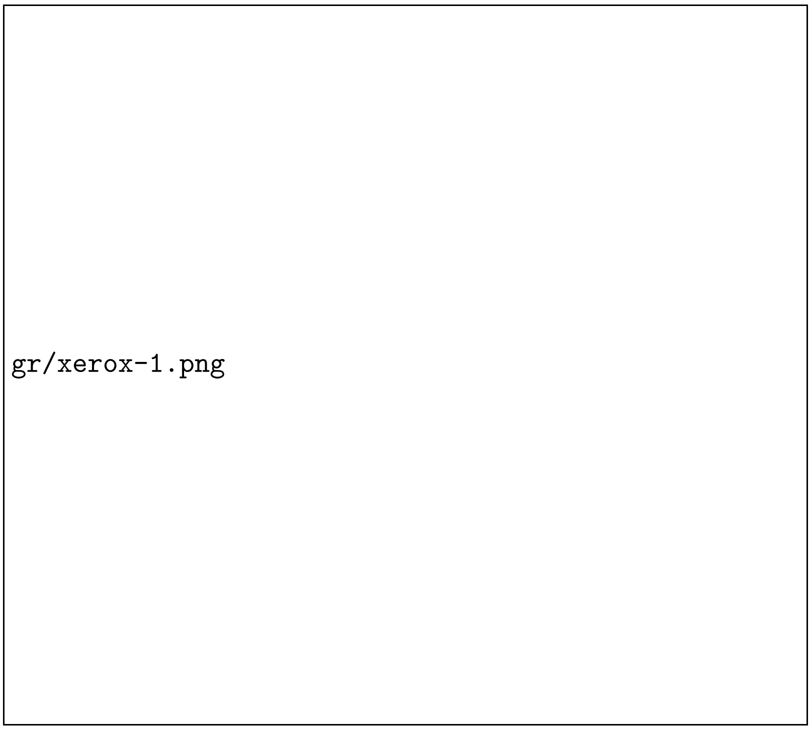

Figure 3.1: The original Xerox PARC Map Viewer

One of the first demonstrations of dynamically generated images was the Xerox PARC Map Viewer[8] (see figure 3.1) which allowed the user to navigate a virtual atlas, where each "window" to the map was generated on demand. This initial quick hack has since been superceeded by professional map

[4] Greenspun talks about SCHLOW webdatabaserservers - write a bit about that

[5] ...the transfer statistics...– `http://www.emsphone.com/stats/cdrom.html`

[6] ...This single machine runs FreeBSD...– `ftp://ftp.cdrom.com/.message`

[7] ...www.freebsd.org...– `http://www.freebsd.org`

[8] ...Xerox PARC Map Viewer...– `http://mapweb.parc.xerox.com/map`

companies which produce maps of an uneven quality on demand, like Yahoo Maps[9] (USA only) and MapQuest[10] (covers Europe too).

The problems with CGI-scripts is due to several reasons:

- **Script run as a subprocess** – the script is executed with the same permissions as the web-server itself, including access to `/etc/passwd` and other possibly sensitive files. Such scripts had to be *trusted* or - for ISP's - inspected before installation to ensure that the script would behave properly.

  This has been addressed with the development of "safe languages" where the execution environment is guaranteed that the CGI-script is confined to a "sandbox".

- **Incompatible platforms** – the CGI-programmer may not have access to a C-compiler which can generate binary code for the machine running the webserver.

  This has caused interpreted languages like Perl to be very popular. Perl programs can be run immediately on a given platform without needing to be recompiled, and are for most purposes as fast as the equivalent programs written in C or C++. Recently Java Servlets (see section B.5 on page 95) have appeared as a popular and well-supported alternative to CGI-scripts, which is being supported by more and more web servers.

- **Slow** – the overhead just for invoking the CGI-program in a subprocess is substantial even for moderate load on the web server, since the web-server must "fork" a new process for each request.

  This has been addressed by putting the execution environment inside the web server, using threads instead of processes, and caching compiled versions of scripts.

Several solutions to the above problems have emerged. Of these, Java Servlets are discussed in section B.5 on page 95, PHP3 scripts in section B.5 on page 95, and Perl (with the mod_perl acceleration module) in section B.5 on page 95).

Server side scripting never really took off before the PHP3 and ASP languages provided "persistent connections" to a database servers, which was unavailable to the standard CGI-scripts.

Persistent connections provide a real performance boost! Database connections are notoriously "expensive" to establish, so needing to open one for each and every CGI-script were a true performance bottleneck. Just by keeping the connection open (and for ASP-scripts - retain the particular database connection for the session) was extremely beneficial. That combined with a very simple way to switch between code and HTML plus that every user could use it without needing to ask the webmaster to install a potentially dangerous CGI-script, meant that the ability to generate webpages dynamically became available to everyone.

[9] ...Yahoo Maps...– `http://maps.yahoo.com/py/maps.py`

[10] ...MapQuest...– `http://www.mapquest.com`

3.2 When a site grows, it becomes hard to maintain

This section needs to be written.
Talk about the growth of the net, and broken links, both externally and internally
Jakob Nielsen. Start of next section should be rewritten

3.3 Navigational structure should not be maintained by the author

reference? Almost all dynamic web-sites which add and delete web pages run into the problem of maintaining site-integrity, regarding ensuring that all the "links" point to the correct document. For external document all you can do is to check occasionally that the page is still there, but for internal documents the webmaster has to do the updates manually. Updating HTML-documents manually is at best a tedious pain, because it is hard, repetitive, mindless labour, which should be left to a computer.

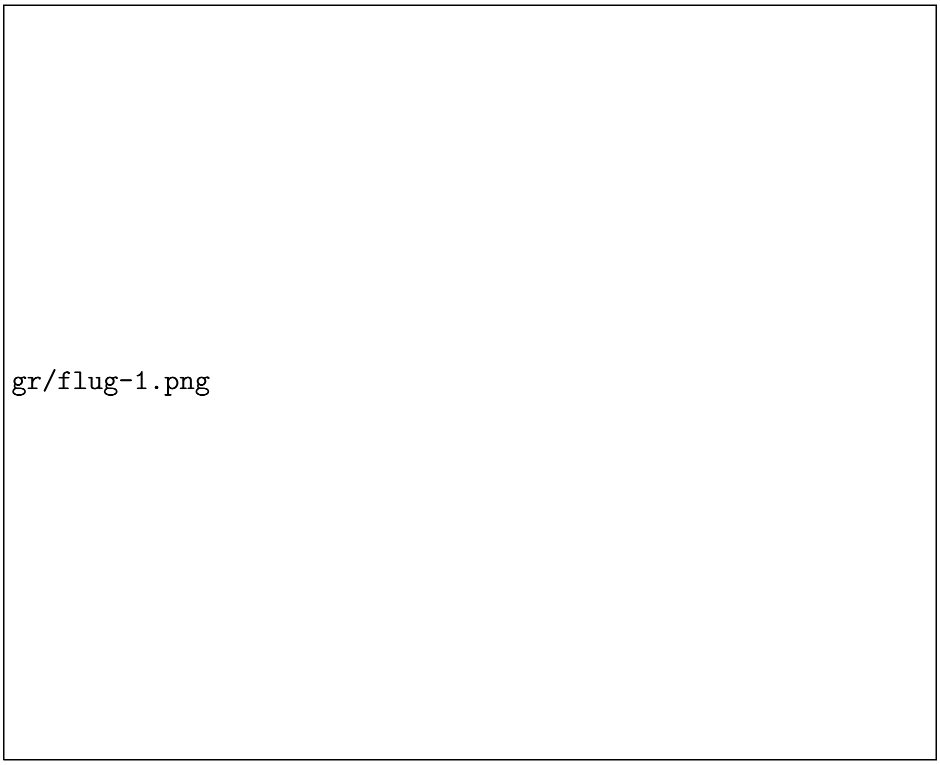

Figure 3.2: The Fyns GNU/Linux User Group web-site. The side bar and the top graphic was added automatically by a script by Tobias Bardino

Figure 3.2 shows a sample page from the Fyns GNU/Linux User Group web-site, showing the way FLUG chose to do it. The framework is added to the HTML-file with a small perl script which must be run to publish each page.

Does it hang together?. The most simple version of this, is to say that every web page on the site must have a static header usually containing at least a link to the entry page and a search engine, but where the header is the same for every page.

In the original webservers this required that every webpage was modified to include the snippet of HTML which produced this header, which is a relatively easy task with a modern scripting language like Perl. Note that care must be taken not to change the modification times of the files, since failing to do so invalidates local copies in browsers and webcaches, even though that the contents of the pages is unchanged. (Modern webservers allow a webpage to include other files with Server Side Includes, which takes care of all this - except doing it unconditionally on every page. The user must still remember to insert the appropriate command sequence in the server).

Figure 3.3: The web page for drivers for the Intel VS440FX motherboard

Many modern websites put a given page in a *context* where the navigational framework reflect this context. See figure 3.3 for a sample from Intel regarding the VS440FX motherboard[11]. This requires more discipline for the web author writing each page, but can still be implemented by including the appropriate HTML-snippet, since these are the same for all pages regarding this motherboard.

Reference to INTEL.COM with navigational fraework for motherboard vs440fx.

In order to ensure integrity in the navigational framework for a site this site, it should be created automatically. Doing so requires meta-data about the individual pages in order to place them correct in the framework.

[11] . . . Intel regarding the VS440FX motherboard. . . – `http://support.intel.com/support/motherboards/desktop/VS440fx/softw`

3.4 A good website needs meta-information about its documents

What to a webmaster is a nice, and well maintained web-site, is to a computer just a bunch of directories of files with bytes in them. In order to get any use of a computer in maintaining these, it is important to have easily accessible information about the desired functionality and the files to work with. Even though complicated rules can be constructed to extract information from webpages, the basic rule is still that

A human must enter the basic information for categorizing a given webdocument.

check spelling of altavista Incidentially this is also the reason why Internet search engines like needs elaborate information extraction techniques in order to remain useful. AltaVista (which is discussed in section 4.8 on page 36) was the first Internet Search engine for *all* webpages. Check precise way to do it and expand the text a bit. Web pages were considered in isolation or compared with the rest of the site?

It didn't take long for web authors (especially those with adult material) to realize that the best way to get their web pages frequently returned in a top position at AltaVista was by putting as many potential search terms in META-tags in each and every of their web pages, showing that just blindly extracting keywords uncrititically is too simple a method. The Google search engine (see 4.8 on page 36) uses heuristics to assert the quality of the results, and is - in my opinion - the only usable search engine for the net today.

The search engines failed because they had no control over authors and the authors had a desire for "breaking the system". For further reading, Douglas Hofstaedter talks a lot about the impossibility of building an unbreakable system for automatic detection of bad input in [Hofstaedter(1979)].

Screen shot of webpage here

An example of a document generated from meta-data which has been automatically extracted, is the MIP "Recently Changed Pages" (see figure B.5 on page 95) the original version of which I wrote while working for MIP as a student programmer. The script traverses three disjunkt set of web pages on the server - System pages, Sysop pages and user pages (one set per user) - and generates a list for each set sorted by title. Each file listed has its age in days next to it.

The meta-information extracted was:

- **Title** - used for the link, and sorting the entries in a set. Extracted from the content of each document, with a default of the filename if no title was present.
- **Age** - extracted from the underlying filesystem which registers the last change of the document
- **User** (for the user pages) - also extracted from the filesystem (or was it from the URL?). It is also used to look up the picture of the user.
- **Size** - Did I use it? For anything? Check code

To me this is just about all the useful information a flat Unix-filesystem can provide. Additionally nothing is guaranteed about the contents of a HTML-file - even the title is not even always there, making it unreliable to rely on authors providing the information expected by any automatic process.

If more meta-data than the above is needed, the authors must be involved and as a part of their web-authoring, deliberately and carefully ensure that the meta-information needed by the automatic processes is correct and up-to-date.

The *Yggdrasil system* (see section A.1.3 on page 72) was my first attempt to generate a navigational framework from information extracted from webpages. Yggdrasil was to function as the automatic webmaster on the intranet, in order to avoid having to assign staff to do so. Then the individual employee could publish information in form of a webpage, add a category either in the title or as a META-tag, and trigger the next update of the framework. This update would scan all web-pages, and extract tuples of (author, category code, publishing date, title, size) of those web-pages which had a category code, and generate a tree structure of web-pages to navigate the categories. Additionally lists of “Documents sorted by author” and “Documents sorted by date” were generated to help users locate documents.

This was on a closed Intranet, where the users were interested in using this as a tool. The incentive for “breaking the system” was very low.

Yggdrasil worked very well initially especially when its very low amount of meta-information is taken in consideration.

Screen shot from Yahoo vreffigAn example of a good site built with meta-information is Yahoo[12] which started as a directory over web pages where the maintainers manually categorized *many* web sites, and used this information to regularily generate static navigational pages . This works really well - I often use Yahoo as an electronic librarian, allowing me to find a website on a given topic I have never seen before (and Google to find it again at a later date) -

The question is then *where* should the author put the meta-data?

3.5 Avoid chaos - keep information in its place

In the programming world, a very visible example of meta-data is the documentation for programs. Experience has shown that it is notoriously hard for programmers to keep the documentation up-to-date, since it is usually considered a get good quote from somewhere :-), which is not felt a part of the “creative, fun” process of writing programs. If the writing of documentation was well established as an integrated part of program development, and the programming language itself helped as much as it could, it would be much easier for the programmers to keep the documentation updated.

Experience has shown (references man month) that documentation should be as close to the thing it documents as possible. For programs, that means that the documentation should be *in the same file* as the code. This has traditionally been done by using comments embedded in the source code.

grow terse - does not stand on its own - hard to maintain even for the author - abstraction layer: the idea was to *hide* the code and just show the meta-data

Typical example is Tanenbaum Minix with source listing with line numbers and a cross reference - bladre from og tilbage hele tiden. 1991 technology? Designed for teaching. Well written books, but horrible programmer [Tanenbaum(1987)].

[12] ...Yahoo...– http://www.yahoo.com

This observation is not new. New is that industry recognizes the usefulness of documentation on the web, and the need for programmers themselves to create such documentation. In order to gain wide acceptance such meta-data management systems must be *standards*, either as *de-facto standards* or as a standard body.

3.5.1 Javadoc - embedding web-information in programs

Get a screen shot

This is perhaps the most visible meta-data management tool available to programmers today. JavaDoc is a tool that creates documentation in form of HTML pages from Java source code with embedded comments, where all definitions are parsed to give a full overview of the Java source code.

Javadoc is a specialized tool which is only suitable for generating web-pages from Java source code, but as Sun both use it for their reference documentation (url to that) as well as ship it with every copy of Java Development Kit, it is a tool which is widely available. Programmers with a need to document their code, will most likely use javadoc to do so. link to the java code at ACME.com

Javadoc has also raised the expectations of the programmers, since they have become used to hyper-linked documentation in a browser to accompany any code they are to use. This tendency is a good step in the right direction.

3.5.2 The Plain Old Documentation format for Perl

list an example and say that this works well. Just look at CPAN

3.5.3 Literate Programming - Knuths approach ??

THIS MIGHT HAVE TO BE MOVED

> *"I believe that the time is ripe for significantly better documentation of programs, and that we can best achieve this by considering programs to be works of literature. Hence, my title: "Literate Programming." Let us change our traditional attitude to the construction of programs: Instead of imagining that our main task is to instruct a *computer* what to do, let us concentrate rather on explaining to *human beings* what we want a computer to do."*
>
> Donald Knuth – [Knuth(1992)] quoted from The Literate Programming web site[13]

Donald E. Knuth has written the TEX-system used to typeset this report, and documented it by publishing the source code to the entire system in four books. The TEX-system has impressed since it is of an extremely high quality, both in code and documentation. Knuth offers money to those who find bugs in his code - on an exponential scheme.

references to the TeX book, MetaFont book, Literate Programming Book, nuweb (my observations)

[13] . . . The Literate Programming web site. . . – http://www.literateprogramming.com/

In order to write both documentation and code of such high quality, Knuth developed his own method of coding which he named "Literate Programming", in which the author works with a single file containing the documentation *as it is to be presented to the reader* written in a TeX-dialect, with the actual code (with a few characters escaped) listed in named chunks along with their documentation, as it fits the author to present them. This file format is the `web`-format!

- The `weave` tool converts the web-file to a `tex`-file to be processed by TeX, and printed.
- The `tangle` tool generates the actual code to be compiled, by joining chunks with the same name, and inserting them in other chunks that reference them (a simple macro expansion), eventually producing one or more flat files which can be compiled.

We definitively needs a sample of this!

Reference to literateprograming.org

The problem with Knuth is that things that work well for him, does not work quite as well for the rest of us. No editors - not even Emacs - can help where a given file is separated in a lot of regions, being either TeX-code or source code, meaning that the editor cannot provide the supporting functions which a programmer might have become accustomed to.

You cannot code verbatim - some characters are reserved for other purposes and must be written differently, which may be very annoying to learn. Additionally the concept of chunks all over the document may be counter-productive if these are hard to navigate.

What else

Many tried this- cloned the functionality -etc - I tried it, and wrote a few programs with it.

Many people have experimented with the possibilities of a weave/tangle pair resulting in several software packages, notably:

FunnelWeb[14],noweb[15]

http://w3.pppl.gov/ krommes/fweb.html#SEC3

There is a webring for Literate Programming[16], which is an excellent place to look for further information. The Collection of Computer Science Bibliographies provides a search engine in literate programming publications[17].

Oasis has a page on Literate Programing with XML and SGML[18] what about it? - note the references for working with the TEI DTD

My previous experience with Literate Programs can be summarized as:

- The documentation gets bigger and better, simply because the printed documents look better that way. What would pass as a single line comment in a source file, looks almost pathetic when typeset. The full power of TeX also encourages usages of illustrations and graphs.

[14] . . . FunnelWeb. . . – `http://www.ross.net/funnelweb/`

[15] . . . noweb. . . – `http://www.eecs.harvard.edu/ nr/noweb/`

[16] . . . a webring for Literate Programming. . . – `http://www.webring.org/cgi-bin/webring?ring=litprog;list`

[17] . . . a search engine in literate programming publications. . . – `http://liinwww.ira.uka.de/searchbib/SE/litprog`

[18] . . . a page on Literate Programing with XML and SGML. . . – `http://www.oasis-open.org/cover/xmlLitProg.html`

- The program development gets cumbersome. You may have trouble using your favorite tools for editing, compiling and debugging. Users of most integrated development environments cannot use this model since the IDE does not provide hooks to provide this processing.
- A critical point is whether the intermediate files are visible to the author! In order to be usable, *all* derived files *must* refer to the original document in a transparent fashion, meaning that the user should not have to worry about intermediate files. (In the same way that the C-processor works under Unix).

My conclusion was that the Literate Programming paradigm does not pay off as a individual programmer, but may work very well for a larger programming team where good, current documentation is critical.

3.5.4 Rational Rose - the other way around

Talk to BNJ. Rational Rose - a UML development tool - uses another approach, work on meta-data with hidden code in the nodes, and generate code to actually run. Allows reverse-engineering to make code development easier. Experiences from users.

3.5.5 Compiling source code into an executable

Most software projects does not consist of a single huge source file which would take ages to compile, but of several smaller files, which can be compiled individually and linked to an executable file. If a file with source code is changed, it is not necessary to recompile each and every file but only those which depend on this particular source file, and then relink the executable to incorporate the changes.

This is possible because the authors has provided meta-data about the system, regarding which source files the system contains, which commands to call to compile and link the source files, which libraries should be included, etc. Normally an Integrated Development Environment (like Borland C++Builder[19], Microsoft Visual C++[20], or the Visual Age products from IBM[21]) knows about these things, or a Makefile[22] is constructed to utilize the known relations built into "`make`" as well as allow the author to add new ones.

In effect, adding meta-data allows the program development process to go faster.

3.5.6 anything else with meta-data?

3.6 Requiring multiple views of a document

Some web-sites have so many advertisements and auxiliary links in their layout, that they need an additional version which is well-suited for printing (i.e. only has a single ad). See figure 3.4 on the

[19] ...Borland C++Builder...– `http://www.borland.com/bcppbuilder/`

[20] ...Microsoft Visual C++...– `http://msdn.microsoft.com/visualc/`

[21] ...the Visual Age products from IBM...– `http://www-4.ibm.com/software/ad/`

[22] ...Makefile...– `http://www.eng.hawaii.edu/Tutor/Make/`

gr/zdnet-1.png

Figure 3.4: The ZDNet presentation of a story

gr/zdnet-2.png

Figure 3.5: The “Printer-friendly version” of figure 3.4

facing page for an example from ZDNet[23], and figure 3.5 on the next page for the "Printer-friendly version".

websites realizing that they cannot just offer an advertisement ladden service, without allowing for a reasonable paper version. Since their HTML cannot "remove" the ads for printing, they have to offer two different version of each article.

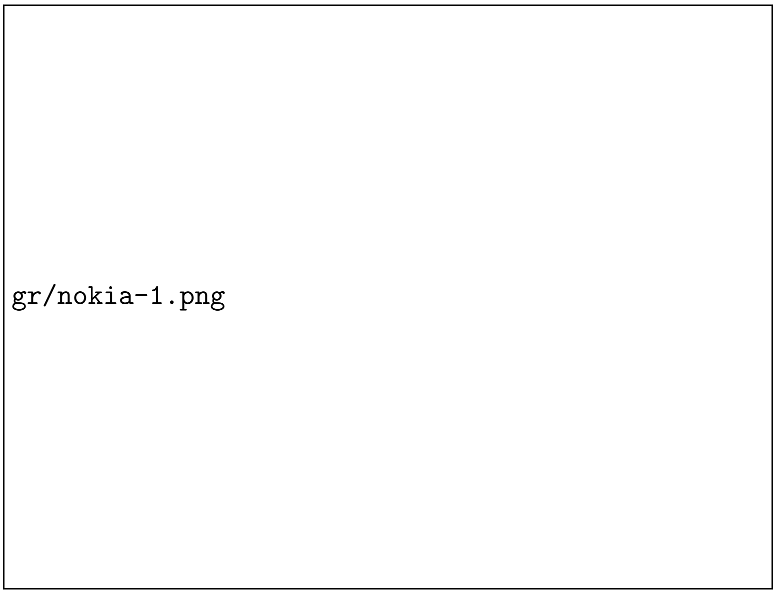

Figure 3.6: Browsing the Internet on the large [sic] display of a Nokia 9110

These are two different *views* of the same basic document. ZDNet are expecting their visitors to either see their documents on a CRT rendered by a reasonably capable browser or print them out on paper. That will probably change within a few years when it becomes common for other devices than just traditional computers to be connected to the Internet. Mobile phones using WAP do not have as much screen real estate (the Nokia 9110 is shown in figure 3.6) so such users may with good reason prefer short and crisp versions of the documents when they use a mobile phone. Nokia is working with 3com to produce a hybrid between a Palm Pilot and a mobile phone. Users have been known to request documents in a format suitable for reading on a Palm Pilot.

This tendency might as a side-effect be beneficial for the blind computer users (see the Blind Webring[24] for more information) . They must normally use text-to-braille software or a "screen reader" in order to use computers, neither of which work well with web pages loaded with graphics, but when web authors must write for many kinds of media instead of just a particular version of a given browser, this will also be beneficial to these users. Of course, it is possible to write ordinary HTML in a way that is usable by these users statens guide for webpublikationer, but that is rare these days.

[23] . . . for an example from ZDNet. . . – `http://www.zdnet.com/zdnn/stories/news/0,4586,2468874,00.html?chkpt=zdhpnews01`

[24] . . . the Blind Webring. . . – `http://www.webring.org/cgi-bin/webring?ring=blind&list`

3.7 SGML/XML: Generic formats for storing data and meta-data

The problem of representing data in a generic way is not new at all, and was solved for the printing industry in the 1970'ies by the design of the Standard Generalized Markup Language (SGML) which is widely used. SGML was deemed too complicated for web browsers, and a trimmed version was named XML and standardized in 1998. Chapter 6 on page 45 talks a lot more about this.

For now it is sufficient to say that it is possible to store both data and meta-data in a convenient form in a SGML or XML file, but that these files must be rendered into the formats expected by the users, like HTML, PDF or what else tomorrow may bring of new, exiting formats. Since SGML describes the *content* and not the layout, it is "just" a matter of creating a renderer for any new format and add it to the collection.

This has been recognized by almost all of the "Documentation Projects" (The Linux Documentation Project[25], The FreeBSD Documentation Project[26] and others) accompanying the various free operating systems available on the Internet.

Chapter B.5 on page 95 talks about rendering documents to a given format on demand (for example in a web browser).

Others? It seems to me that stuff is missing here.

[25] . . . The Linux Documentation Project. . . – `http://www.linuxdoc.org/`

[26] . . . The FreeBSD Documentation Project. . . – `http://www.freebsd.org/docproj/docproj.html`

3.8 The user should use familiar tools to publish documents

SGML-editing is hard and tedious - this is one thing that the folks in the `comp.text.sgml`[a] and `comp.text.xml`[b] newsgroups agree upon. Tools are essential for authors.

Very few tools exist, and the good ones are quite expensive. The general consensus is that the best OpenSource tool is the Emacs editor with the PSGML package (see B.5 on page 95), and that this tool is primarily suited for programmers and other people who are well acquainted with SGML and XML.

PRINT DIRECTLY TO A PDF FILE - WEB PUBLISHING

Computers should *help* you do your work, and users are generally most productive with the tools they know. Why should an author use SGML with an editor she loathes, if precisely the same result can be achieved by using a word processor she is familiar with?

In order to automate the conversion from e.g. Word to the equivalent SGML file, it is important that the author uses only those constructions that the conversion program understands. This is most likely given as a set of macros that must be used, as well as a verification program which the author can use at any time to check whether her document is conformant.

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.

[a] `...comp.text.sgml...` – `news:comp.text.sgml`

[b] `...comp.text.xml...` – `news:comp.text.xml`

3.9 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.

3.10 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).

Chapter 4

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.

4.1 slashdot.org - high volume information site for nerds

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

SlashDot[1] was started in September 1997 with the slogan "News for nerds - Stuff that matters", and do so by providing dynamically generated web pages with a lot of stories categorized with an icon, where a short summary is shown along with links to the full story, several references to the orignal sources, the authors email address, plus an overview of the number of comments in the discussion forum. See figure B.5 on page 95 for the top of the start page at March 14 2000.

```
Slashdot Stats

date: 2:37am
uptime: 68 days, 5:56, 2 users,
load average: 0.18, 0.19, 0.18
processes: 65
yesterday: 209994
today: 1
ever: 278137148
```

[1] ...SlashDot...– http://slashdot.org