An Intranet Solution for a Real-time GPMS in Newspaper Production

Björn Hedin, Fredrik Fällström and Vlad Ionesco
KTH/Royal Institute of Technology, Division of Graphic Arts Technology
SE-100 44 Stockholm, Sweden. E-mail: bjornh@gt.kth.se

Abstract
Newspaper production is a complex and often subopti-
mised process, characterised by heterogeneous computer
based production tools in a distributed environment. At-
tempts to optimize this process have generally been on a
local subsystem level, without a global perspective. With
the emerging IFRAtrack standard for exchanging local and
global tracking information, new possibilities for global
optimization are emerging.

This paper presents the Intranet client-server-server solu-
tion we have chosen as the platform for a Global Produc-
tion Management System (GPMS) aimed at solving these
problems. A production database contains the relevant in-
formation supplied by the local systems and acts as a glo-
bal collector of production information. A web server com-
municates with the database through scripts and other
mechanisms to dynamically retrieve and update its con-
tents. Users interact with the system using WWW browsers,
which provide the basis for the client user interface. Dis-
tributed code modules, applets, can provide functionality
not supported by HTML.

1. Introduction

Newspaper companies of today are highly computer
based with many different local computerised production
systems in use [1]. Advertisements are booked and pro-
duced in one group of systems, stories and editorial mate-
rial are produced in another group of systems and the final
pages are produced in a third group of systems. The same
structure holds for the entire process, including circulation,
printing, mailroom and distribution. The process is handled
by a collection of heterogeneous local systems.

Despite the high level of computerisation, very little
communication of process status information is taking place
between all these systems [2]. Local systems have no
knowledge about the status of other local systems. They
control and optimise their sub-task in the production chain
without taking global considerations into account, which
leads to an overall sub-optimization of the entire process. A
decision that generates profit in one system might cause
much greater losses on a global level.

In an attempt to solve this communication problem,
IFRA (Inca-FIEJ Research Association), the International
organisation for research in the newspaper industry, recom-
mend the use of the IFRAtrack standard [3] as a format for
information exchange between local newspaper systems.

Our research group has taken a leading role in the devel-
Opment of IFRAtrack, and is also implementing a prototype
global production management system (GPMS) called “Big
Brother” in an ongoing research project [4]. Tracking is the
first level of three needed for full functionality in a GPMS
[5]. We consider a global production management system
(GPMS) as including the three basic functions of monitor-
ing, or tracking, the production, scheduling the activities
and resources, and actively controlling the process. A
GPMS can be implemented in a modular, stepwise way,
starting with basic monitoring and evolving toward produc-
tion control.

The Big Brother prototype system is named after the
book “1984” by George Orwell, as a reminder of what the
system should not be like. It is a decision support system
for the entire newspaper production process, used to help
decision makers make right choices. It is implemented on a
middle sized Swedish newspaper, Östgöta Correspondent-
en, with a circulation of 67000 copies every day except
Sundays. The first prototype which consists mainly of the
tracking system, was installed in 1996. Development of the
scheduling and control parts will take place during 1996
and 1997.

Several criteria must be satisfied when developing sys-
tems like these. The systems must be modular so that new
parts can be added when new needs arise. It is also desira-
able to build the system so that it easily can be ported to dif-
ferent platforms and have clients on MS Windows, MacOS
and perhaps other operating system platforms. Good per-
formance and an appealing user interface is also vital.

We have examined possible environments for develop-
ing such a system. We evaluated Oracle Power Objects,
SmallTalk and pure C-programming before we finally de-
cided to base the system on Internet and WWW technolo-
gy, an Intranet.
2. Intranets

The Intranet FAQ [6] defines an Intranet as:

“An Intranet is the use of Internet technologies within an organisation (or company) to achieve better results than the conventional means of data access and transfer. Intranet helps in cutting costs, easy and fast accessibility of day to day information on up-to-date basis to employees spread over different countries, also assuring communication with the outside world as it is connected to the Internet.”

2.1 Advantages of using an Intranet

The choice of an Intranet solution offers many benefits over other types of GPMS solutions. The benefits are not so much of a technical nature, indeed, from a technical point of view, better functionality could be offered by writing customised, platform specific applications, taking full advantage of all the possibilities supported by the operating system. Instead the benefits are more in terms of ease of maintenance, lower costs and a low threshold of learning.

The use of an open, platform independent standard like the World Wide Web offers, allows the company to have a very flexible hardware and software strategy. If the company later wants to change client-side OS-platform, no extra work has to be done except to install a web browser on the clients.

The installation of an Intranet system is relatively simple. If the users already have browsers installed and configured on their client computers, only the server and the server software needs to be configured and installed. There is no need to install any extra software on the client computers. Upgrading to new versions of client software is also a simple task, when new versions of the browser are released.

One aspect that must not be overlooked is the short time of learning required for teaching the users how to use the system. Many users work with the World Wide Web today, and that number will increase over the next few years. That means many users are already familiar with the user interface, and know how to navigate between different web pages. Time required to learn the basics of the system in reduced. This can save thousands of dollars per user in terms of education that otherwise would have been required, and reduce the number of mistakes by users who are unfamiliar with the user interface.

Another advantage in using World Wide Web technology is the platform independence of HTML. HTML-code will be rendered more or less the same on whichever platform the client uses. The actual rendering is up to the client browser. This can be a great advantage, especially in a company such as a newspaper where both Macintoshes and PCs are used widely. There is no need to maintain separate versions of client software for different operating system platforms.

Use of standard WWW technology makes it possible to access the system from anywhere in the world as long as there is a browser connected to the Internet and the server is set up in such a way that access from the Internet is allowed. The decision on whether it should be possible to access the system from the Internet must be based on the corporation’s security policy, but the possibility of doing so can be an advantage.

It is also possible to connect to the system with a modem, using SLIP or PPP. This might be safer from a security point of view than allowing direct access from the Internet.

2.2 Disadvantages of using an Intranet

There are, however, aspects that speak against the use of Intranet technology. One of the main problems is the limited functionality of HTML. HTML is not designed to be a full power programming language, it is a markup language for documents with hyperlinks. This means many functions that are available in a customised client-server solution, written with the full range of features offered by the operating system, quite simply are not possible in HTML.

Solutions to this problem are now emerging, especially with loadable code modules, or “applets” which are downloaded and then executed on the client computer instead of on the server. This is discussed in greater detail in chapter 5.

Another problem with HTML is that it is not designed to provide real layout possibilities. In a company like a newspaper this might be a problem since employees are often quite sensitive about poor layout.

Languages such as Java can solve these layout problems, but that is a bit of an overkill. A better idea to provide improved layout possibilities in HTML documents is the “style sheets” proposed by the W3 Consortium [7]. Style sheets will separate form and appearance from the structural information which HTML is intended to provide. It will be possible for a company to design a specific look on their WWW pages, with particular fonts, margins and other layout features that make the pages easier to read. Many browser vendors have indicated their support for style sheets.

3. Hardware/software overview

The network at Östgöta Correspondenten is based on fiber optics and has a capacity of 100 Mbit/s. The Big Brother system runs on a Sun Ultra 1/170 with 64 Mb memory and 6 Gb disk space, using Solaris 2.5 as operating system.
system. We have also tested our system on an IBM RS/6000 computer running AIX as operating system without any problems. We chose to base our system on Sun/Solaris and IBM/AIX partly because they are the most common UNIX platforms in the newspaper industry, and partly because new software is often released earlier on Sun/Solaris than on other platforms.

The database software we have chosen is a standard relational database, Oracle 7.3, even though the IFRAtrack model would have been easier to translate to an object database. In systems such as these, the costs of a database like Oracle is relatively small compared to the total cost of installing the system, and the benefits in form of support, reliability and performance will make the extra cost worthwhile. Many newspapers also already have relational databases installed, and have qualified staff to administer them. This will cut the costs for maintaining the system.

As WWW server we use Apache, but it is very simple to change to another server if that should be desired later on.

The client computers use MacOS, Windows 3.1 and Windows 95 as operating systems. There are a few dedicated Big Brother client computers at Östgöta Correspondenten, but most client run on the computers used in the daily production. The WWW browser on the client is Netscape Navigator with Java support.

3.1 System overview

The Big Brother system is built around two databases, the IFRAtrack database and the internal Big Brother database (see figure 1). External systems such as monitoring systems for printing presses and XTensions to Quark XPress send information about the current status of the production process to the IFRAtrack database using the IFRA Message Format (IMF). This database is open to all local systems which are able to provide tracking data, and is not exclusively used by the Big Brother prototype system. Other systems should also have access to this information if they have a need for it.

The messages in the IFRAtrack database are parsed by a parser which then updates the second database in the system, the Big Brother database. A web server then updates its web pages based on the contents of the Big Brother database. Users request to see production data via the web interface. Some users also fill out WWW forms in order to send information to the IFRAtrack database, which cannot be provided by other systems. These forms generate normal IFRAtrack messages and can conceptually be treated like any other IFRAtrack message provider.

In our prototype, the two databases and the web server run on the same computer. This need not be the case however. The web server can reside on a separate computer, thereby reducing the load on the database server. The question of using more than one server must be based on performance, security and economical aspects.

3.2 The IFRAtrack database

The IFRAtrack database is used to store the IFRAtrack messages supplied by the external production systems. It is very simple and consists of only one table with three columns: one for unique identification, one for a timestamp and one for the actual messages. The external systems do SQL-inserts on a view of this table, which consists only of the message column. The unique identifier and the timestamp are generated by the Big Brother system.

To simplify the generation of these IFRAtrack messages we have written a C++ class library which is freely available for suppliers of subsystems. We have used this class library ourselves, and DENEX, a supplier of external systems, has incorporated IFRAtrack generation based on this class library into their CopyTrack system which is installed at Östgöta Correspondenten.

3.3 The production database

The production database reflects the current state of the production. Each class of objects in the IFRAtrack model (see figure 2) has a corresponding table in the database, and each object has its own entry. For one issue the number of objects is about 500 at the granularity level we have chosen in our model.
We have not yet come up with a satisfying solution of how to store a history of the events that take place. Until we find a better solution we have a number of help tables in which we log the time of certain events.

We are also currently working on how to model resources such as computers, printing presses and people. This information is needed in the second phase of the project, when we will introduce scheduling.

### 3.4 Poller/parser

The translation from the IFRAtrack database to the production database is done by a parser. A process looks at the IFRAtrack database at regular intervals, and when new messages have been added these are passed to another process, the parser. The message is parsed resulting in SQL statements which then update the production database.

The poller and the parser are implemented in C and Perl.

### 3.5 WWW server software

We create dynamical HTML-pages using the Common Gateway Interface (CGI). CGI-scripts can be written in a number of languages, but the predominant ones are Perl, C/C++ and, in the case of Macintosh computers, AppleScript. We have chosen to use Perl for most scripts, since writing Perl scripts can be done very quickly. This is a great advantage during the development where being able to rapidly try out new ideas is more important than higher performance. A few scripts which demand much computational power and/or are difficult to write in Perl are written in C/C++.

### 3.6 Connections between the database and the WWW server

We have mainly chosen to use Perl with the DBI database connectivity interface [8] as a link between the WWW server and the database. DBI is a very flexible, free of charge set of functions and variables that allows any valid SQL-statement to be passed to the database. DBI is also a database independent interface, as long as there are drivers to a particular SQL-database. This can be a major advantage if the company plans to install a system, but wants to keep an option open to change database vendor in the future without having to do any major alterations of the coding of the database connection scripts.

Another way to connect to Oracle databases is to write CGI-scripts in C using the PRO*C precompiler from Oracle. This gives good performance and high flexibility, but has several disadvantages. The binaries require a lot of disk space and the coding is quite tedious. It is also difficult to pass dynamically created SQL-statements to the database, which is a problem.

A third way to get data from a database to a web server is to use a web server with these capabilities built in, like the Oracle WebServer [9]. With the Oracle WebServer, hypertext links can point directly to an Oracle stored procedure. According to Oracle this is both secure and more efficient than using the CGI-interface. We have not yet had the opportunity to test this solution.

### 3.7 Client software

We have chosen to use Netscape Navigator for the client computers. When we installed the first prototype, Netscape Navigator was the only browser on the MacOS platform that supported Java. Netscape Navigator was also already installed at Östgöta Correspondenten.

We have also implemented two applications directly for MacOS, FilmWatch and PageWatch which are used to generate IFRAtrack messages for films and pages. We will rewrite these applications in Java later on.

We have also implemented an XTension to Quark XPress, which delivers IFRAtrack messages about page status automatically, and also includes a bar-code in the document. This bar-code is used for automated IFRAtrack generation from films and plates later in the production chain.
3.8 Newspaper production simulation

An important help during the development of the Big Brother prototype system has been the newspaper production simulator TidSim, developed at our division [5]. This simulator, which is a freestanding system that runs on the NEXTSTEP operating system, can simulate the entire production of a newspaper. We have added an IFRAtrack message module to this system, so that IFRAtrack messages can be generated and inserted into the IFRAtrack database. This has been very useful since we have been able to test all the parts of the system without the need to have external systems deliver the messages. We have also been able to test the performance of the Big Brother system by sending it messages for an entire production run in just a few minutes. In the future it will be useful, since we are able to test and validate our system without disturbing the daily production at Östgöta Correspondenten.

4. Security aspects

One of the first aspects that has to be considered before setting up a WWW server is security. Who needs to have access to the information? In an Intranet system such as the one described in this paper, it is probably not necessary for anyone outside the company to have access. This will greatly simplify the task of making a secure system, since the web server can be placed inside the firewall. So far we have chosen this solution for the Big Brother system.

If Internet access should be allowed, greater care must be taken to prevent unauthorised users to gain access to the computer systems. This chapter mainly describes some of the aspects that must be considered in this case.

4.1 Setting up a server

The server can be placed on the inside or the outside of the corporate firewall. If no external access should be allowed, the question is simple. The servers should then be placed inside the firewall. If, on the other hand, external access should be permitted the question becomes more tricky, since placing the server inside the firewall might compromise the entire security in the network if security holes are found. It is probably better to place the servers outside the firewall, a “sacrificial lamb” configuration. The risk of breaking into the servers exists, but at least doing so will not compromise the inner network [10].

There is also the possibility of placing the WWW server outside the firewall and the database server on the inside. Communication between the WWW server and the database server then goes through the firewall, and can be more tightly controlled.

The server should be launched as root to allow access to the standard http port 80, but should then fork off a child process that runs with privileges as user nobody. This user can read documents and execute programs usable by everyone, but not much more.

A good, but rather complicated, way to increase security is to run the server in a chroot environment. To do this you set up a special area on the server in such a way that no other parts of the file system are visible. The programs that are necessary to run the web server must be copied to this area, but otherwise the area should be as barren as possible.

4.2 CGI-scripts

CGI-scripts are potential security holes in any WWW-system. The problem is that any user with access to the WWW-system gets to execute programs on the server, programs that may contain serious bugs or that may give away information about the system if they are not written with care.

Generally, scripts written in compiled languages such as C are considered to be more secure than scripts written in interpreted shell-like languages such as Perl. Often CGI-scripts written in Perl make use of system commands, like system() and exec(). Careless use of these system commands can prove to be very serious security holes, if used with unchecked user input [10].

The problem of executing system calls with unchecked user input as arguments can be reduced with mechanisms such as “taint checks” in Perl. Taint checks will cause Perl to exit if user supplied variables are passed to the shell without being “untainted” first.

Another potential problem with interpreted languages is if an attacker somehow gets hold of the script source code instead of executing the script. One example how this can
happen is if a Perl script is edited in emacs. Then a ~-file is left in the same directory as the script. If the server is configured in an unfortunate way, an attacker will be able to load the file `script.pl~` which contains the old version of the source code. The attacker can then examine this code and can find bugs in it, which can be used to launch attacks against the system.

Other problems present themselves in C-scripts. If the length of inputs isn’t checked this may cause the program stack to be overwritten, which in turn can be used to invoke commands. This cannot happen in Perl since Perl dynamically extends its data structures to accommodate the data size.

Finally, scripts should be placed in a special cgi-bin directory, rather than being spread around the file system. This allows tighter control of the scripts.

### 4.3 Server side includes

Server side includes can be used on some servers, like NCSA httpd. This can provide convenient functionality in several cases, but is also a security problem. If for example users can write text that later will be displayed in HTML documents, such as in guest books, this can be exploited to execute programs. For instance, a user could write this in a guest book:

```
Björn Hedin
<!–#exec cmd="cat /etc/passwd"–>
Sweden
```

The safest solution to avoid these problems is to disable server side includes.

### 4.4 Java

When using Java applets on HTML pages, care should be taken to verify that they are not maliciously written. If such applets are used, current implementations of Java have security holes that might compromise the security of the system. Most of these holes have been corrected but it likely that new security holes will be found in the future. More work is required to make the execution of remotely-loaded code safe [11].

In an Intranet using Java or similar languages, this should not be a major security problem. The applets can be expected to come from a trusted source. One problem, however, can be that the Java-option must be enabled on the browsers. Then malicious scripts might be executed when the user loads pages from the Internet.

### 4.5 Encryption

Is encryption needed for systems such as these? That depends on whether the system should be accessible from the outside world or not, and on how secret the data really is. If it must be possible to access the system from the Internet, the communication should be encrypted. The mechanisms of authentication built in to most web servers where passwords and/or specific IP-addresses can be required to access specific pages are not safe enough to prevent a determined attack on the system. If, on the other hand, the system is only accessible from the company’s local network, encryption is probably not required.

### 4.6 Passwords to access pages/scripts

The use of passwords to restrict access to certain HTML pages and CGI-scripts, which is supported by most servers today, can be very useful, even on a system not connected to the Internet. For instance only the management should be able to see some of the reports, and only certain employees in certain positions should be able to fill out forms which will change data in the databases. From a security point of view, these password mechanisms should not be trusted too much, but they can be useful just in order to restrict access for the users who are supposed to use the system but who are not supposed to see everything.

### 5. Applet code modules

The last year has seen a tremendous development in the interactivity of web pages. Code modules, or “applets”, which are loaded to the browser from some server, are interpreted by the browser. The results can provide much of the functionality that is not provided in HTML, especially when it comes to interactivity. Another important aspect is that it moves computations from the server to the client.

The principle behind applet technology is that code written in some kind of platform independent language is loaded in a web page. The instructions to load the code is embedded in a special HTML tag like `<APPLET>`, `<OBJECT>`, `<EMBED>` or in an anchor element using the REL="embed" attribute. The browser needs to understand the tag in order to load the code module, and of course it has to be able to interpret the code itself.

When executing foreign code on a computer, care must be taken to prevent “attacks”. Attacks from a malicious applet can take several shapes. Interfering with other programs, consuming system resources, destroying files or permitting foreign access to your system are but a few possibilities. Therefore applets like this usually run in a protected environment, where they cannot access files and are not allowed to make connections to other hosts.

### 5.1 Different applet languages

The most well-known programming language used to
write applets is Java by Sun Microsystems. Java is an object-oriented, interpreted language with a syntax close to C/C++ and is designed to be platform independent. As long as a Java interpreter exists for the operating system, Java code can be executed, and the screen output will look more or less the same on every type of computer. Because of security reasons, Java applets do not have the full functionality offered by the Java programming language. For instance, file access and connections to other computers are strictly controlled, since that could violate the security of the system. Most commercial browsers support, or will support Java-applets.

Python is another interpreted, object oriented language that runs on most kinds of UNIX dialects, MacOS, MS-DOS, MS Windows, Windows NT and OS/2. Combined with the Tk User interface toolkit and interfaces to various databases, very powerful, interactive programs can be constructed in Python. The browser Graal which is implemented in Python supports Python applets, which work in the same way as Java applets.

Yet another browser that supports a language for applet technology is SurfIt!. It is written in Tk and Tcl, the free user interface toolkit and scripting language. SurfIt! supports applets written in Tcl.

5.2 Applets in the Big Brother system

When we decided to use an Intranet for the Big Brother system we realised that HTML was not powerful enough for the user interfaces. We soon decided to use applet technology, and to use Java for writing these applets instead of more exotic languages such as Python and Tcl. More effort is put into developing Java than Python/Tcl, and better support is available. Also, Python and Tcl require special browsers which only exist on UNIX, while Java can run on the market leading browsers.

The applets now included in the Big Brother system display the current state of the production database. For example they show how the page production and film production proceeds, like in figure [4]. We use a polling strategy to update the applets. At regular intervals, decided by the user, the applet asks the database for the production information the applet will display. We communicate with the database by opening a DataInputStream to a URL which points to a Perl script on the server. This is a slightly round-about way to get access to the database, and we will probably change to using JDBC, the Java database API, later during the project.

The polling strategy puts a lot of unnecessary work on the server. Most of the time, nothing has happened in the database. Currently only a few computers actively use the applets at any one time, but in the future more computers are expected to be connected simultaneously. Then an event driven solution would be better, and we plan to implement an event server to accomplish that (see below).

Applets will also be used to move computation intensive activities from the server to the clients. For instance, generating graphs of printing curves is a computational intensive activity currently done by the server. If instead, the points which are to be plotted are passed from the database to the client where an applet does the actual graph generation, the server load is reduced.

6. Reports

Since most of the employees at Östgöta Correspondenten have access to email, that medium is an excellent way to distribute reports.

At the moment all reports are generated at 05.00 AM, that is after the newspaper has been printed but before most of the work on tomorrow's paper has begun. Key employees can subscribe and unsubscribe to the various reports using a web-form.
These reports consist mainly of key numbers such as the time each page was sent to the printing plant, when the actual printing started, and the number of waste copies produced when the printing press started. They can also contain links to web-pages with graphical reports such as gif pictures of the printing curves for a printing job. These graphical reports can also be sent as attached documents in the email as an option, if the addressee does not have access to the Intranet WWW server.

7. Future work

Our future work will focus on adding scheduling and control to the system. We will also add components to the current system to increase performance, add functionality in form of more forms of report. We plan to implement as much as possible in Java.

7.1 Event server

For the next phase of this project we plan to implement a more general event server. Resources such as people, and programs like applets should be able to subscribe to various types of events.

There are several events taking place during the production of a newspaper. One type of event is the state-transition in the objects described in the IFRAtrack model, such as when a film changes state from in-progress to completed, or when a printing job changes state from created to in-progress [3].

Another type of event is when an object exceeds a deadline in its current workflow. An example is if page 17 is scheduled to be sent to the printing plant before 11.15 PM, but exceeds that deadline.

A third type of event is a combination of the events described above, and can be derived from these. If the production of two pages have exceeded their deadlines, this would generate a warning event. The event server can then distribute this signal to the subscribers of level 4 warning events. These events can be of different types, so if seven pages have exceeded their deadlines a level 1 panic event is generated instead.

7.2 SMS service

We are also experimenting with sending some reports via an SMS server. SMS stands for Short Message Service and is a way to send text messages to cellular phones using the European GSM network. These reports would only be urgent warnings, when it is vital that some person is notified of some event. Since an SMS message can only consist of 160 characters, no verbose reports can be transmitted.

8. Conclusion

Intranets can be a very effective solution for distributing information within an organisation. We have found that they are easy to maintain and modify according to the users wishes.

Many of the problems concerning the limited functionality in HTML can be solved with applet technology, which will increase the number of areas in which Intranet technology can be used.

We believe that applications written in platform independent languages such as Java will rapidly increase in importance, especially in organisations where different computers run on different operating systems.

Using Intranet technology for actively monitoring and controlling a production process has not yet been widely tested, but we believe these kinds of solutions will grow in importance.
Literature cited


