Preface

A "main project" is a final project that every student at Gjøvik University College has to carry out at the end of their last year and the main project for the Graphic Arts class of 2001 was to be carried out from January 2001, with a deadline for the project at May 23.

Project background
The project started with an inquiry from GAIN, Graphic Arts Intelligence Network, on the September 26. 2000. GAIN currently has a website that is static, and is not functioning in a satisfying way. The desire is therefore to establish a new dynamic web site with the possibility for the GAIN members to update the page via a browser interface, and maintain their own profiles.

In addition to this they would like a brand new and more functional design. GAIN also wants to explore the possibility for using XML on the web site.

Main Project
The following statement is taken from Gjøvik College's official guidelines for students conducting a main project.

Students are to:

- *Independently complete a significant, multi-disciplinary scientific task.*
- *Produce documentary evidence of task planning and of the solutions found*
- *Understand the advantages and disadvantages of working in a group*
- *Understand the importance of making and following up good project plans*
- *Consider alternatives within a method and problem based working ethic*

The project is to be based upon realistic engineering problems formulated such that their solution will require the use of knowledge from many disciplines.

The Project Group
Our project group consisted of three students from the Graphic Arts Engineering Program at Gjøvik University College, Norway. Originally we were supposed to be four, but the fourth member became ill in the early stages of our project, and he did not recover in time to finish the project. In the early stages of the project we made a progress plan based upon four members of the group. But since the fourth member didn't recover in time to participate to the group at all, we became a group of three. This hasn't lead to any major consequences, other than that we suffered from lack of time at the end of the project.

Employer
This project has been commissioned by GAIN – Graphic Arts Intelligence Network, from now on referred to as the/our employer. Our contact person within GAIN has been located in The Netherlands.

Advisor
Every group of students are appointed an advisor from the school. Our advisor on the project has been associate professor Ole Lund.

Project title
Web site for GAIN

Stig Brænden Stian Gjerde Terje Hansen

Gjøvik, Norway
2001.05.22

Contributions

During the project several people have contributed in various ways to our project. These are the people we feel deserve to be given attention:

Ole Lund (associate professor)
Roelof J. Janssen (contact person in GAIN)
Sven Erik Skarsbø (associate professor)
Torbjørn Høvde (fellow student)

content

1 introduction

2 literature review

3 the prototype

4 discussion

5 conclusion

appendixes

1 introduction

1.1 Author's declaration

In the initial phase of the project we communicated with GAIN through associate professor Sven Erik Skarsbø at our college. He has until recently been a member of GAIN, and it was he who first established the contact between GAIN and our project group. Unfortunately, this led to a serious misunderstanding about the aim of the project. In the correspondence between Skarsbø and us, Skarsbø explicitly and repeatedly expressed that the aim of the project was to develop an "*Internet portal for GAIN"* (and the formal contract between Gjøvik University College and GAIN, which is signed by Sven Erik Skarsbø on behalf of GAIN, carries the title "Internet portal for GAIN").

However, the actual "web site" that we have developed – in accordance with GAIN's wishes (vaguely specified in informal e-mail messages received throughout the project period) – can hardly be called a "portal". Nevertheless, due to the misunderstanding described above, for a long time during the project we focused on "portals". This is the reason why among other things the literature review deals almost exclusively with "portals". We became aware of this misunderstanding too late to rewrite the literature review.

1.2 The Project Group

The students in our group are currently on our third and final year at the Graphic Arts engineering Program, at Gjøvik University College, Norway.

Stig Brænden
Stig was born in 1976 comes from Gjøvik. After high school he spent two years at Gjøvik Technical College, before attending at Gjøvik University College.

Stian Gjerde
Stian Gjerde was born in 1977 and comes from Larsnes. Before attending Gjøvik University College, he spent two years at Ålesund College studying Computer Science.

Terje Hansen
Terje Hansen was born in 1977 and comes from Folldal. He has studied at Gjøvik University College, since finishing high school.

1.3 Employer

Our employer was GAIN – Graphic Arts Intelligence Network, an association of graphic consultants from Europe. The following statement is taken from their old web site:

GAIN is an association of independent consultants throughout Europe with links outside Europe. GAIN focuses on the graphic arts, multi media and computer publishing industries. Members offer complementary specializations, so that GAIN not only covers Europe geographically, but it also penetrates every segment. The consortium conducts projects on a pan-European scale.

GAIN consists of one member from each country. Each member can outsource activities to associated members.

Our contact within GAIN has been Roelof J. Janssen from Amsterdam, Holland. He is the founder of Screens & Pages, a consultancy firm for prepress and multimedia.

Since our contact person is located in Amsterdam, the communication between the group and GAIN has gone via the Internet and e-mail. We established a web page where we posted status reports every week, so that GAIN could follow the progress of the project.

1.4 Competence training

When we first assigned for this project we knew that there were several skills and competencies that were required to carry it out within the project deadline. This included skills in the following areas:

- Web design (the understanding of good design for the Internet)
- Project management
- The use of various software like:
  - Adobe Photoshop (image processing program)
  - Adobe GoLive (WYSIWYG web development tool)
  - Adobe Indesign (desktop publishing)
  - Adobe Illustrator (vector images)
  - Macromedia Dreamweaver (WYSIWYG web development tool)
  - Microsoft Project
- HTML coding
- ASP coding
- Writing satisfactory reports
- Databases

Some of the skills we already possessed, like using the different software tools, HTML coding and some project management. The areas we needed to develop our skills further within was web design and typography, but the main technology we had to learn was ASP (Active Server Pages). The reason why we needed a technology like this was the need for interacting with a database. This technology makes it possible to store member details, reports, news, events, etc. in a database on the server.

To develop our knowledge in these areas we conducted a literature review at the beginning of the project.

Actually we didn't decide to use ASP until we had written the literature review. There were several other different technologies that we considered. But after having undertaken the literature review we decided to use ASP and an Access database. This was because we felt that ASP was a solution that would fulfil both the employers and our own requirements for the web site.

1.5 Problem description

The intention of this project is to develop a web site that combines the easy-to-operate HTML with the technological advantages of newer server based technologies. The web site should interact with databases and provide a foundation for the new GAIN website.

1.6 Project description

Target specification and Accomplishment

The Effect goal
Exploit and develop our competence within the technology required to achieve the project goal, which includes both the technical aspects and the aspects concerning what is considered good design and typography for the Internet.

The Project goal
Through research work, prototyping and analysis the project goal is to build a prototype for a the GAIN web site.

Accomplishment
Straighten out the demands and needs from the employer, and further to model prototypes and propositions in order to receive feedback during the project from the employer. Finishing a final proposal for a final product, in addition to a final report on English.

External conditions

Time conditions
Project start was 8 January 2001 and project deadline was 23 May 2001. See appendix B.

Workload
A main project at Gjøvik University College, Norway consists of 6 points for each student. i.e. about 360 hours or 3 months of work per student. One year of study equals 20 points.

Cost limits
Expenses during the project are to be covered by the students. This could be travelling expenses, printing, various office equipments etc.

Extent

Organization of the project

Responsibility
At the very first stages of the project, we decided to go for a project model without a project manager. We did this because of the small size of our group. Midway into the project we realised that it would probably be a good idea to appoint a project manager in the group.

Other roles in the project group
Other roles in the project group:

Backup of all the project material : Stian
Updating the web site we developed for communication with GAIN : Stig
Keeping the minute book up to date : Terje
Commenting on the ASP code : All of us
Keeping accounts : Stian

Planning, follow ups and report

The project was divided into four main sections

I Pre-engineering

Establish contact with the employer
Pre-engineering report
Conferences with our advisor
Gather resources, technical and personnel

II Research

Conduct a literature review
Gather information about our employer, GAIN
Build up knowledge within the technical aspects of the project
Maintain an active dialog with the employer

III Main section

Develop a functional design
Develop prototypes
Programming
Testing
Trouble shooting
Reporting
Continuous dialog with our advisor
Feedback from employer and advisor
Continuously updating our project web site

IV Project closure

Finalizing our final prototype
Finalizing the final report
Presentation of our project

Status meetings with our advisor every other week
Internal status meetings every Monday.

1.7 Quality assurance

Quality assurance of our project is done in the status meetings, both internal and with our advisor. On the meetings we controlled how the progress had been, and where we stood according to the time schedule and our milestones. Backup has been taken of all the files we produced. These were saved on more than one disk in addition to the original location. The documents were also named with explicit and unambiguous names.

1.8 Progress plan

Milestones
The project consisted of four major milestones. These were corresponding with the four main section of the project. These were the deadlines in our project:

The pre-engineering face:	29 January 2001
The research face:	25 February 2001
The main section:	04 April 2001
The project closure:	23 May 2001

In addition, see appendix x, for Gantt-chart

1.9 Target group

Target group for the project has been our employer GAIN and their nine members.

1.10 Requirement specification

A. Introduction

GAIN wants a web site, which has a better design and better functionality. Their current web site is too static. They want a better system for publishing news, press releases, reports etc for the members. Some of GAIN's reasons for making a new web site are:

– E-commerce, i.e. sell PDF reports
– Make it easier to update and add entries
– New, and more up-to-date design

B. Structure

The structure of the web site will be hierarchic. The start page should contain links to most of the underlying pages i.e. the different sections of the web site.

Home	The start page. Info on GAIN, head lines etc
News	The latest news
FAQ	Frequently asked questions about GAIN and the web site
Events	Information on current and upcoming events
Press releases	Press releases from GAIN
Projects	Overview of projects conducted by GAIN
Services	Overview of services offered by GAIN
Publications	Free reports and reports for sale
Members	Overview of GAIN members and company info

C. Functionality requirements

C.1 General requirements

– The web site should be easy to use and have a functional design
– The typography should be easy to read

C.2 Technical requirements

– Download time should be as short as possible
– The graphic elements should use few colours to keep file sizes small
– Use "web safe" colours
– The web site should function both on Mac and PC

C.3 Internet functionality

– Only the GAIN members should be able to publish news items, press releases and other documents
– The visitors should be able to search for published articles in a database
– Visitors should have the opportunity to purchase reports on the web site
– The navigation panel should be visible on all pages and sub pages

D. The users

D.1 User groups

– GAIN members
– People in the Graphic Arts industry
– Potential customers
– Vendors

D.2 Access rights

Administrator	All rights
GAIN members	Publish, edit and delete

E. System/server environment

The web site should be placed on a WinNT/Win2000 server with IIS (Internet Information Service) and ASP support.

F. Technical limitations

F.1 Hardware

Mac or PC
Display: 800x600 resolution or better
256 colours or better

F.2 Software

Mac – Microsoft Internet Explorer 5 or better

PC – Microsoft Internet Explorer 5 or better
– Opera 5 or better
– Netscape 6

1.11 How we approached the project

Our first step in solving and defining the problem was to conduct a literature review. By writing this literature review we gathered information about technologies to choose among the further directions of the project. We also learned a lot about other aspects of a major Internet project like this, in all this lead to the state where the group possessed the required knowledge in order to do the project.

We then moved on to the next stage where we started working more individually with different tasks. This was mainly a stage, where we programmed and tested several different possible solutions for the final prototype.

The next stage was to complete the final product. This included using all our previously gained knowledge to make a product that we were satisfied with.

The final stage was writing the final report.

2 literature review

2.1 Introduction

When we first assigned for this project, we must admit that our knowledge in the field of Internet portals was just basic. All of us developed several web sites over the years, both in exercises at the college and on our own. But these page were just basic HTML pages with some text and pictures. So to find out more about the topics relevant to Internet portals, we searched on the web, bought books and ordered articles via the college library. We also read some articles published on the web by large Internet development companies. Some of them are also cited in our report. The first question we discussed is the most basic of them all: What is an Internet portal?

2.2 What is an Internet Portal

When we hear the word "portal", most people think of it as another word for a web site or a web page. But is it really? The answer is no. A portal is more like an entrance to other sites on the Web. You could just look up the literal meaning of the word "portal", and then you would find that it is the equivalent of an entrance. On the Internet a portal is a collection of links to and information about other resources on the net. There are two kinds of portals, vertical and horizontal.

Vertical Portals
Vertical Portals are portals that contain information and links, on one single area of interest, for example a portal for the film and entertainment industry, like http://www.warnerbros.com. At warnerbros.com they have gathered a lot of information and links to the film and entertainment industry.

Horisontal Portals
Horisontal Portals are portals that cover several topics, and contain information and links to web sites all around the world, regardless of which type of sites they are. Good examples of such portals are all the so-called "start pages". These are pages that are ment to get you started on the Internet, and they contain links to web sites which may interest you. There may be links to newspapers, car magazines, online travel agencys, comic sites, shopping tips, health education an so on. An example is http://www.lycos.com. This is a typical start page where you can reach almost anything you would like.

There are many ways to categorise the different types of portals on the Internet. We have come across many definitions of the different types of portals, but one of the best is this one, taken from a report made by the Delphi Group:

1. **Publishing portals** target large and diverse communities with disparate interests. These portals tend to follow a fairly traditional metaphor which involves relatively little customization of content, except for online search and limited interactive capabilities, expected by the casual Web user (broadcasting).

2. **Commercial portals** offer the possibility of narrowing content for potentiallly multiple diverse audiences.

3. **Personal portals** deliver specific filtered information for individuals (narrowcasting).

4. **Corporate portals** coordinate rich content within a relatively narrow community which unites around a group mission.

This is the most common way to describe a portal, and also the one we are using.

Publishing portals are as the name indicates, portals for publishing information. This is a type of portal that contains mainly a one-way-traffic of information. There are little or no opportunities for the user to interact with the portal. An example of such portal is newspapers.

Commercial portals can be both horisontal and vertical. Warnerbros.com is an example of a commercial vertical portal.

Personal portals tend to be vertical portals. Containing information for specific user groups. Like for instance a portal for people who like to follow the Hubble telescope, and related information.

Corporate portals are most likely vertical portals, and are most often portals used to promote a company on the web. E.g. information for both stock owners and customers. Corporate portals can also be portals where users can carry out transaction. They may be able to request specific information, buy items, or even set up meetings or appointments with sales staff.

2.3 Building an Internet Portal

The most crucial point in building a portal is planning. You have to develop a strategy for both the portal structure and the portal layout. A structure that makes it easy for the user to get in touch with the information that he/she seeks. There are many portals on the web today that contain a so confusing layout that the user just gets annoyed and leaves. The information that the user seeks may very well be on the portal some place, but what good is that if he can't find it. But to be fair to portal developers, this has recently significantly changed to the better. And I think that developers are more aware of this problem now.

Structure of a horisontal portal
The content of a horisontal portal should be divided into main categories so that the user can easily sort out the information. The user could divide their content into about four to eight main categories. More than that just makes it confusing for the user to get where he wants. When the user has chosen one of the main categories, he can start narrowing down his search. Once the user has entered his main category, there could be quite a few subcategories, because then he knows that he is on the right track, and he won't get confused so easily when searching through the menus. The structure used on most vertical portals are quite similar.

Structure of a vertical portal
As mentioned in the last section, the structure used in vertical portals can often be similar to the structure used in horisontal portals. The main difference is that if you are on a vertical portal, you probably already now what kind of information you are seeking. While horisontal portals are often used as starting points on the Internet, the vertical portals are aimed at a specific user group with specific areas

of interest. For example a travel portal, where you can reach all the resources you need for planning your trip. This could be buying your flight ticket, booking a hotel and make a reservation of a rental car.

The portal layout

It is also always a good idea to have a clear vision of what the content on the portal should be, when you are planning the layout. Different types of content go with different types of layout. For instance if you have an internet portal where you are trying to sell reports etc. You should use colors and a design that makes your portal looks professional and that makes a decent first impression. You don't want to use flashy colours and a layout that looks like the homepage of a twelve year old boy. Do your "homework" before you start working on the design.

2.4 Content of an Internet portal

One of the things that may be used to describe the difference between a an "ordinary" site and an internet portal, is that many of the internet portals contain exclusive membership areas. A membership area contains services and information you have to sign up for in order to gain access. This is common on many of the start portals that exists.

Another important feature of a portal is the possibility to have personalised interface when you're logged in as a member. That means that you can sort out the type of information that you would like to view, and then the next time you log in to your area, only the info of interest are viewable to you. An example is the Norwegian news portal http://enter.vg.no. Here you can choose so the only news that comes up when you are logged in are sports news. In that way you can easily eliminate the types of information that you don't want to view.

Another feature that is available on many portals is the abillity for individual members to contribute information to the other members, for example by uploading of documents, reports, and pictures.

2.5 How to generate traffic on an Internet portal

When you are building a portal on the Internet, you must make sure that future visitors return frequently. Many Internet sites and portals go bankrupt because they can't generate enough visitors to make their sponsors and advisors happy. The host must make the portal attractive. There are different ways and techniques to improve traffic on a portal.

An Internet portal where nothing happens or changes will not attract new visitors. They may come back once or twice, but that's it. Updating the content is therefore crucial. In vertical portals this is more important than in horizontal ones. If a person is very interested in Indian music, he won't go to a portal that is just as comprehensive in food, sports and computers. It is most likely he surfs to a website that specializes on his taste in music. But for a vertical portal, like the fictional "www.enter-the-world-of-indian-music.com", updating the content is a matter of staying alive.

Another way to attract future regular visitors is to include some sort of personal planning. Almost every large-scale portal offers private e-mail accounts and free space for personal homesites. Some also offers a personal organizer where you can keep track of meetings, birthdays etc, often with a reminder system like an e-mail alert or similar. There is no end to the possibilities… This is a trick that encourages the user to return instead of going to other sites that doesn't "know him so well".

Other portals reward their users by collecting bonus points for clicking on ad banners, ordering merchandise through the respective portal, or a discount if you simply use the portal as the start page when you enter the Internet.

A solution often used by pornography portals, is like a labyrinth where it is easy to access, often via banners on other sites or by using the word sex in a search engine, but extremely hard to leave. This technique works best for vertical portals specialized on a given subject. On horizontal portals that focus on a wide term of subjects, this labyrinth model is not an option.

You must also focus on building a functional portal; it should be easy and obvious how to navigate and find what you are looking for. If these techniques are used, the portal is one step closer to success.

2.6 Portals and XML

One of the most hyped technologies today is XML, eXtensible Markup Language. This technology is also successfully used when building corporate portals. But why is XML so important when building a corporate portal, and what is XML?

The benefits of using XML in corporate portals
There are two major benefits when it comes to XML as a data format. The first is that it separates the data from its rendition. This means that XML only describes data and not how it is going to be displayed. The benefit of this is that the rendition of the XML data can be decided later and easily changed. The same XML data can be converted to HTML for delivery in a browser or WML (Wireless Markup Language) for delivery to a mobile phone. The bottom line is that the same XML data can be used for multiple purposes.

The other benefit is that the data can be made explicit. You can "tag" your data and it doesn't affect the way data is displayed. You can add tags with just about any name without affecting the rendition. The tags should describe the content of the data.

HTML vs. XML
HTML was originally developed to describe the appearance of the data. Today this concept is more in a blur, but that was its intention. What it doesn't do is describing the content of the data. XML allows tags describing the content of the data. This is a major advantage. For instance if your data contains serial numbers, just add the tag "serial number" and put the serial number in it.

Why use XML in building a corporate portal
Usually corporate information portals contain two kinds of information: structured documents (Word, PowerPoint etc) and unstructured data. Unstructured data are for instance stored in a database system. When data is converted into XML it makes it more search able. This is because of the separated content and display information. When the content is separated from the display and the information made explicit, the portal can be powerful and assure global access.

Other standards
WebDAV (Web Distributed Authoring and Versioning protocol) is another standard. WebDAV allows client software to write directly to the portal server without any detours. The Microsoft Office 2000 then can publish directly to the portal. This reduces the hassle and there is no need to go through FTP upload.

2.7 Scenario net vs. Internet portal

Recently a new term has caught the attention of Web devel opers around the world, a new way of thinking Web struc ture, that usually goes under the name "Scenario net". A Scenario net can be described as a site where the developers have thought through the likely scenarios their customers will need to perform. That means that they have come up with many solutions to the many likely scenarioes that are related to the task the customers came to their site to perform.

A good example of a web site that is higly scenario based, but not a Scenario net, is the site Autobytel.com. On their site the customers can shop for a new car or a used car, research cars, finance a car, insure a car, and track the service record of a car. Another good example is Biztravel.com, where they have really understood the needs of the frequent business traveller. It is organized around the typical business travel scenario. It lets you book flights, rent cars, reserve hotel rooms for multi segment flights, and itinerary options based on convenience or cost. They offer you the possibillity to go back and change your mind about one or every aspect of the trip as many times as you want, and as you're book ing, you can also optimize frequent flyer miles, select pre-ferred carriers, and keep costs down. But still, even with all these features, autobytel.com is not a Scenario Net. So what is the defintion of a Scenario Net?

A Scenario Net is a number of portals or web sites that cooperates and shares user information. In a Scenario Net you will enter your profile information at the first site you enter, and then both your profile and the information about your requested task will be passed along with you, as you move back and forth between the interlinked sites. In that way the Scenario Net will be able to perform all related

tasks that you request in a way that is much easier and more convenient, that you are able to do with the Web portals today. In short that means that a Scenario Net contains a portable profile that you bring along with you, and makes sure that you are able to perform all the tasks you want that are related to the information you requested. And best of all, this is not limited to a single company. You could move along between the cooperating companies until you have completed your task.

2.8 Technologies

When building a portal there are some technical aspects we need to consider. How are we going to program the portal, are we going to use just an HTML editor or a WYSIWYG tool (What You See Is What You Get) like Adobe GoLive, Macromedia Dreamweaver or Microsoft Frontpage. In the next chapters we discuss several tools and technologies, which are commonly used for devloping advanced web sites, all of which we considered for our project.

ASP

ASP (Active Server Pages) is a Microsoft technology, which provides a lot of new possibilities within Internet publishing. An ASP file does not differ much from a HTML file, it includes text, plain HTML, XML and different scripts, but the scripts are executed on the server, not the client's computer.

How does it work

When the browser request an ASP-file, the server passes the request to the ASP engine, which reads the ASP file line by line, and executes the scripts in the file. Then, the ASP file is returned to the browser as plain HTML. The ASP code cannot be viewed from the browser.

Advantages of using ASP

It is easy to dynamically edit, change or add content on a web page. You can access data or databases to add or search for content like pictures, documents etc., and return the result to a browser. It provides possibilities to customize a web page to make it more useful for individual users, by letting users log on with a username/password to access a personalized content. ASP provides security since the ASP code cannot be viewed from any browser, the code is returned as plain HTML. ASP is also faster and simpler than other techniques like CGI or Perl.

CGI

CGI (Common Gateway Interface) is not a language, it is a protocol that can be used to communicate between e.g. web forms and a program most often written in C++, Perl or Visual Basic. CGI was originally developed to let people all over the world query a database on a server. The host must then create a CGI program that transmits information to the database engine, receive the results, and displays them to the client. In other words, you basically let the world run a program on your server, which isn't the safest thing to do. Therefore, there are certain security precautions that need to be implemented when it comes to using CGI programs.

CGI is the oldest technique to create an interactive environment for WWW. Today, more and more people use ASP, Java or PHP technology instead, but on some servers CGI is still the only way to implement interaction.

PHP

PHP (Hypertext Preprocessor) is a server side HTML encapsulated script language. The PHP script is run on the server and the result is sent to the client. PHP differs from CGI by including the script in the HTML file instead of having a lot of code to get HTML output. The PHP code has special start and end tags to go in and out of PHP mode. Another possibility is to configure the server to treat all HTML pages like PHP and thereby make it impossible for the user to see that data is being processed on the server side. PHP can be run on many different platforms and have support for databases, XML, Java, several Internet protocols etc. The PHP syntax is based on C, Java and Perl.

What PHP can do

PHP can do what any CGI script can do at the most basic level. PHP can for instance send and receive cookies, collect form data and generate dynamic page content. PHP has support for a wide range of databases and it supports services using protocols like HTTP, POP3, IMAP and several others.

Why we decided to use ASP

Our project requires use of a technique that allows GAIN members to easily change or add content on their web site. This may be uploading publications, search for earlier published files from a database, and a log-on system with administration possibilities. ASP is a newer technology than CGI, and it is easier to code, and it executes faster. PHP is based on Unix platforms. It is also most likely the possible provider GAIN will use in the future offers a NT-based platform.

3 the prototype

3.1 Introduction

The prototype is made entirely of ASP files, mainly to implement the dynamic features GAIN required. They also wanted to differentiate between GAIN members and other visitors. We discussed several solutions, and came to the conclusion that the best way was to insert membership information into a database.In this way, members must log in with a unique username and password to access pages where they can influence the content. Only an administrator or webmaster can register or delete new members and assign the login information. The members area contain functions where members have the opportunity to add or delete news articles, press releases, information about events and perhaps most important, reports. The latter section is divided up in "free reports", and "reports for sale".

The report database will usually be a larger database than the other databases, and therefore we have added a search engine to help both members and non-members to find the report they are looking for. It is also possible list the entire database.

Use of ASP files also makes the site more secure, because the code source cannot be viewed from the client's computer. We have also added a function that prevents non-members to access the members area.

3.2 Design

The design, old web site

GAIN also wanted us to improve the existing grapic design. Our first reaction of the current site is the choice of colours. They have used a dark grey as background colour, which disturbs the text. They have compensated this with setting all the text in bold, which is not comfortable to read on pages with lot of text.

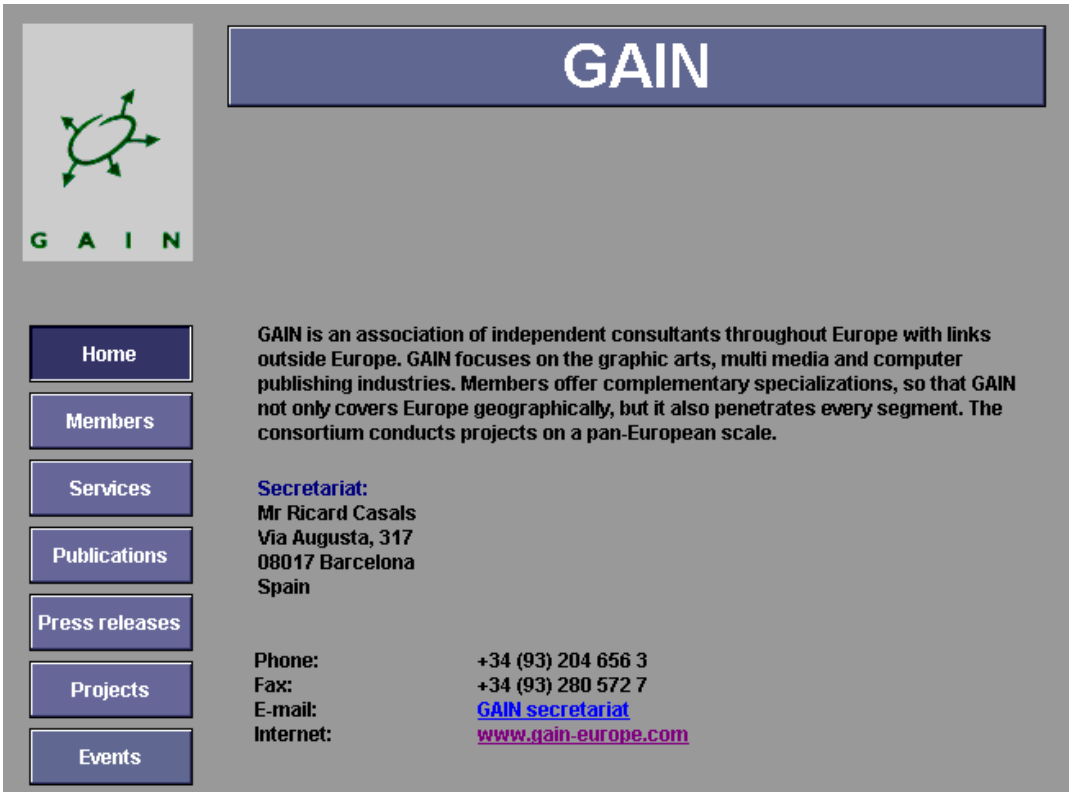


fig. 1 Old GAIN web site

The header and the buttons appear to have been made in a button generator, and the placing of these makes the site look divided. The GAIN logo contains a green colour on a light grey background, which breaks with the rest of the design.

Based on this, we decided to make a total renewal of the site and not re-use anything but the structure itself.

Design, new prototype

Resolution

We decided to make the new site 800 pixels wide even though most people have 1024 x 768 resolution these days. The reason behind that is that a horizontal scrollbar will appear for those very few using 800 pixels resolution on a 1024 pixel wide site. A Macintosh using 1024 x 768 will also have the horizontal scrollbar because the browser window does not fill the whole screen horizontally. When we did our research, we realized that very few corporate web sites are designed 1024 pixels wide. The height is relative, depending on the content.

Colours

We spent some time discussing which colours to use. The background colour should be white, because of legibility reasons. White is also easy to mix with other colours.

First we made a sketch with a green header and navigation menu, because of the green in the original logo as seen on figure 1. None of us really felt for that idea, so we began with something entirely new. The GAIN logo had to be redesigned. We kept the symbolic circle with arrows, but changed the colours. We made the logo white on a blue pattern, mixed with two different gradients of blue.

But still something was missing, the logo didn't attract enough attention.
An orange bar was inserted, and that actually made the whole top of the site look nice. On top of that we added "Graphic Arts Intelligence Network" with a light grey font colour under the blue header, and now we felt that we had something going on.

We used the same blue pattern at the bottom, only separated with a grey field at the right, where we put our project logo.

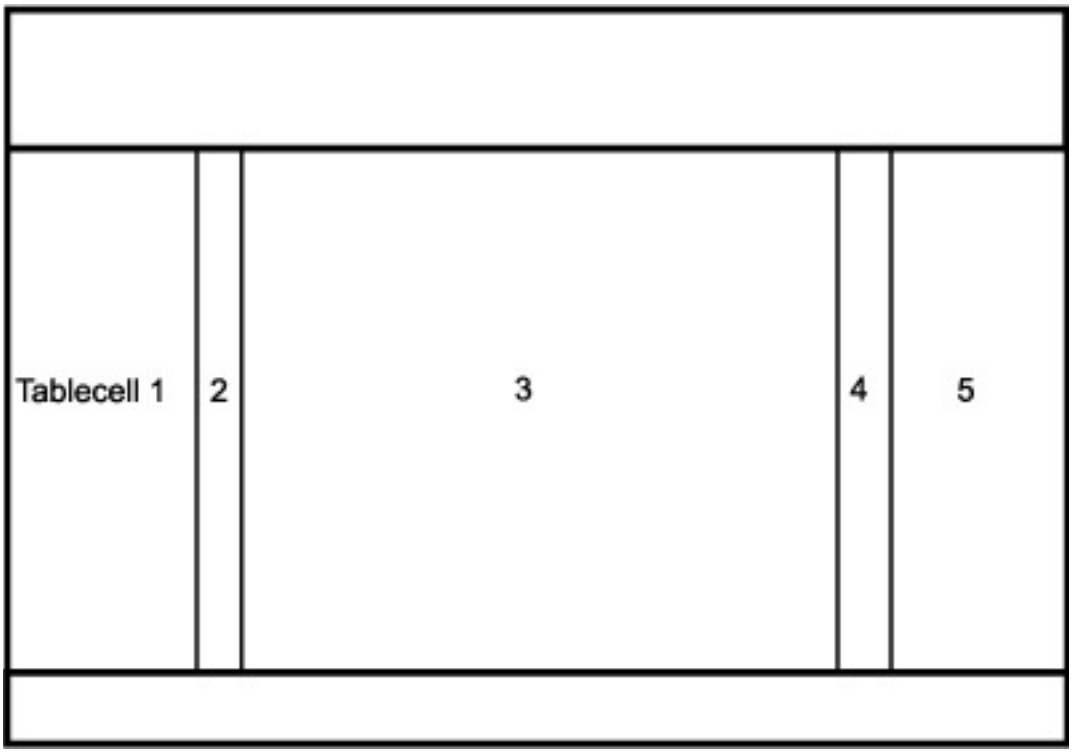
Now we had to make a navigation panel, we tried about 10–15 different design solutions, butwe were not satisfied with any of them. We agreed it had to be rollover buttons to make sure the user understand it really is a button. The last button clicked should also differ from the rest, so the user can see at every time where he is in the web site, and make the navigation easier. We used the same colours as we used on the header.

Another version of the navigation panel vas made with a blue back ground, but then the panel appeared like a blue block with no coherence at all.
The right side of the page should content other links, a search engine and recently published articles/ reports. To separate them from the main area on the page, we filled this table cell with a light grey colour.

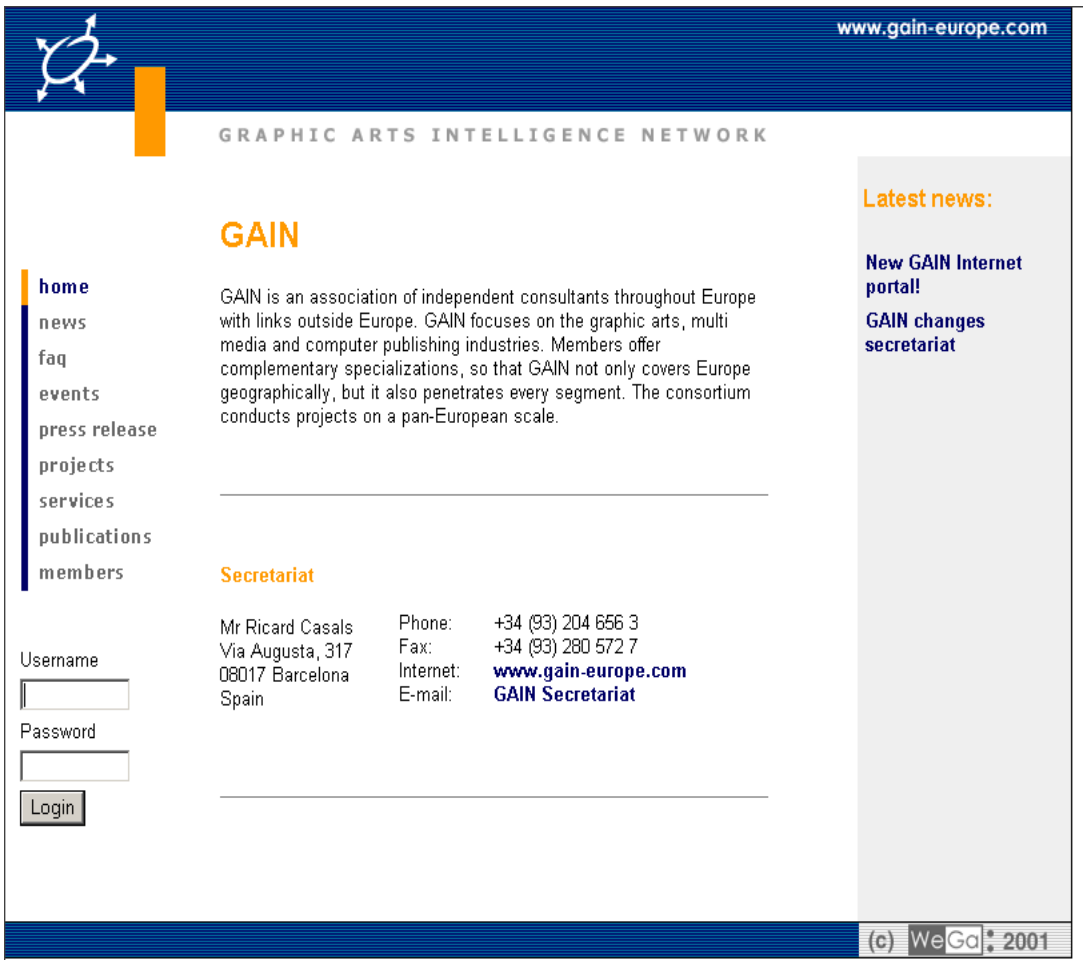
Browser limitations
To make sure the colours look the same even on low-end systems, we used web safe colours from the colour cube.

Site design

Since the vertical hierarchy in this site is made shallow to maintain easy navigation, we went for a table-based site design instead of frames. The structure is very simple, with three tables beneath each other. The centre table is again divided up in several table cells. More precise layout is solved with style sheets and with a new table inserted.



site structure, template



The final template including content from the start page.

3.3 CSS and fonts

We used Cascading Style Sheets to preserve visual resemblance to the text and link layout on every page. This is the style sheet text implemented in the template:

```
<style type="text/css">
<!--
      .standard {font-family: Arial, Helvetica, sans-serif;
      font-size: 10pt; color: #000000}

      a {font-family: Arial, Helvetica, sans-serif; font-
      size: 10pt; color:
      #000066; text-decoration: none; font-weight: bold}

      a:hover {font-family: Arial, Helvetica, sans-serif;
      font-size: 10pt; color:
            #ff9900; text-decoration: none; font-weight:
      bold}
-->
</style>
```

Standard, a and a: hover are the predefined styles. Standard applies to the body copy, i.e. all text except hyperlinks and headings. The text is shown in Arial with size 10 pt black. If Arial is not installed on the client's computer, it displays Helvetica or another sans-serif font face.
Style **a and a:** hover applies to hyperlinks. They are shown in a dark blue colour, with a bold font-weight. When the mouse pointer covers the link, the text turns orange, to accentuate that it is a hyperlink.

We chose to use Arial as the primary font instead of e.g. Verdana, which is a font especially designed for web usage. Verdana may be more comfortable to read, but Arial suits our design better. The TrueType font Arial, although not specifically designed for screen or web usage, as Verdana is, is nevertheless optimized for screen usage by extensive hinting. Both are sans-serif font faces, being highly legible on the screen.

Graphic elements format

The graphic elements on the web site are the logo bar on top, the navigation panel and the bottom bar. The grey table cell to the right is filled with a one pixel high gif with a black pixel repeating itself depending on the height of the cell. This makes a black borderline on the right side. All elements are generated through Adobe Photoshop's "save for web" function, which compresses the file size as much as possible without losing information. Every page contains 8,9k graphic elements and 8,2k HTML. This gives a complete download time of 2 seconds with an ISDN connection, and 3 seconds with a 56k modem, which is very fast. The rest depends on the server.

HTML code

The HTML code is commented on every page, making it easy to edit to for new webmasters/administrators. Meta tags are not yet inserted to prevent visitor coming to this prototype instead of the original GAIN site, but it is shown in the code where to place them.

3.4 Navigation

Before we began to design the navigation structure, the rule of thumb was that all information on the web site should be available within one or two mouse clicks. The navigation panel is present at every page, so the user easily can go directly from one page to another without going via the default "home" page. If you're not logged in, the login field is present on every page, and it disappears on every page after you log in. When you want to log out you will find a logout link at the bottom on every page. On pages including large areas of text, there is always a back to top link. You will also find a link back to the previous level if you are on a sub-level.

Visual feedback
The user interface is self-explanatory. It is designed with the intention that you never should have to guess what to do. There is alwayssome sort of feedback on each function. That may be the headlines on every page, the relevant button on the navigation panel is shown with a different colour. If you log in with incorrect information or try to enter restricted pages, you will always get an error message referring to what you did wrong. Another feedback example is when a publication is deleted. First you select the file you want to delete, and then you're asked if you are really sure that you want to delete e.g. "filname.pdf". Then a confirmation is shown. All the pages where you have a delete option contains also the database listed on the right side, so you can physically see that the file is removed from the database.

All hyperlinks are consistently blue by colour, and turn orange when the mouse pointer hover over them. This prevents misconceptions.

Navigation within the members area versus the non-members area

The main difference between members and non-members pages is the possibility to upload and delete publications and articles in the members area. The login field is also replaced with a log out link, and the home button leads to the page you are directed to when you log in correctly. This page does not include information about GAIN like the start page, but the latest news articles and upcoming events. This is like a "what's up?" page for members, the first thing they read after they have logged in.

Navigation within the Administrator area

The administrator area has a quite different interface, because here you are supposed to access only three different functions only. This is *view and edit membership details*, *register new member* and *e-mail a member*.
You have to log out to access the non-member pages, and log in again to enter members area.

Technical solution

To implement dynamic functions and safeguard the security we, as earlier mentioned, went for an Active Server Pages solution. This chapter explains the different ASP functions like the login/member system, upload and delete publications and articles, the search engine and the security system and the database connection. All ASP code is annotated in the source code. We have also used Java script on some functions.

Database connection
Every time an ASP file receives or puts information into a database, a connection with the database must be established first, and closed at the end of the file.

```
<%
on error resume next
set conntemp=server.createobject("adodb.connection")
katalog_filnavn="DBQ=" & server.mappath("articles.mdb")
conntemp.Open "DRIVER={Microsoft Access Driver
(*.mdb)}; " & catalogue_filename
%>
```

This code establishes a connection with the database named articles. *On error resume next* prevents the database connection to stay open if an error occurs, and continue to the end of the file where the close database script is:

```
<%
conntemp.close
set conntemp=nothing
%>
```

The database design

We chose to use a Microsoft Access 2000 database, because it works quite well on this system considering the low data load.

This system connects with 3 different databases. *Articles.mdb* has a new table for each publication category, news, press releases, events, and reports for sale. *Members.mbd* contains two tables, one with membership and one with administrator information. We use a new database for free reports, because it differs a bit from the articles database. The free report database contains whole reports in pdf or word format, not only text, and it links to a folder where the actual file is saved.

Members : Table

id	Full_name	Country	Username	Password	Join_date	
81	Terje Hansen	Norway	123	123	17.05.2001 11:18:14	terje@h
82	Jan Eskildsen		Eskildsen	Jytte	18.05.2001 00:20:19	je@gai
79	GAIN Member	Europe	test	test	15.05.2001 22:16:13	gain@e
(AutoNumber)						

extract from the members table. fig 2

Java script

In addition to ASP programming, we also use Java script on some functions. The Java script forces all the buttons in the navigation panel to preload to make them all appear at the same time. The script also includes the roll over effect that the buttons have.

```
<script language="JavaScript">
<!--
function MM_swapImgRestore() { //v3.0
var i,x,a=document.MM_sr;
      for(i=0;a&&i<a.length&&(x=a[i])&&x.oSrc;i++)
            x.src=x.oSrc;}

function MM_preloadImages() { //v3.0
      var d=document; if(d.images){ if(!d.MM_p)
            d.MM_p=new Array();
      var
      i,j=d.MM_p.length,a=MM_preloadImages.arguments;
      for(i=0; i<a.length; i++)
if (a[i].indexOf("#")!=0){ d.MM_p[j]=new Image;
            d.MM_p[j++].src=a[i];}}}

function MM_findObj(n, d) { //v3.0
var p,i,x;  if(!d) d=document;

if((p=n.indexOf("?"))>0&&parent.frames.length) {
      d=parent.frames[n.substring(p+1)].document;
                  n=n.substring(0,p);}
if(!(x=d[n])&&d.all) x=d.all[n]; for
            (i=0;!x&&i<d.forms.length;i++)
x=d.forms[i][n];
for(i=0;!x&&d.layers&&i<d.layers.length;i++)
      x=MM_findObj(n,d.layers[i].document); return x;}

function MM_swapImage() { //v3.0
var i,j=0,x,a=MM_swapImage.arguments;
      document.MM_sr=new Array; for(i=0;i<(a.length-
2);i+=3)
if

((x=MM_findObj(a[i]))!=null){document.MM_sr[j++]=x;
      if(!x.oSrc) x.oSrc=x.src; x.src=a[i+2];}}
//-->
</script>
```

We also use the Java script shown below to give the user a feedback in the form of an alert box that pops up if he forgets to fill in required information in forms. This script alerts if you forgot to fill out any of the fields in the "new members" registration form.

```
<script language="JavaScript">
<!--
function CheckRequiredInfo()
{

if (document.register.email.value == "")
{
alert("An Email is required");
document.register.email.focus();
return false;
}
else if (document.register.password.value == "")
{
alert("A password is required");
document.register.password.focus();
return false;

}}
//-->
</script>
```

Member/administrator login

The login form is present on every non-member page. You have to fill out both the username and password fields. When you click the login button, the information is sent to another ASP file, register-action.asp, that controls the information submitted against the member database.

```
<%@ LANGUAGE="VBSCRIPT"%>

<%
username = request("username")
password = request("password")%>
```

These lines retrieves the text from the form and put them into variables.

```
<!--#include file="config-start.asp"--->
```

The include file opens the connection to the members database, and checks both member and administrator table.

```
<%    if username = "admin" then
            if password = "admin" then
            Response.cookies("loggedin")= true
            response.cookies("registered")= username
            response.redirect "../admin/default.asp"
      end if
      else
      end if
 %>
```

If admin is entered as username and password, and it agrees with the fields in the database, the user is directed to the administrator pages.

```
<%checkPass = "select * from members where
username='"&username&"'"
'response.write checkPass
set passSet = canvasDB.execute(checkPass)
if passSet.EOF then
```

If the username entered does not agree with the database:

response.redirect"loginerror1.asp?error=
Username%20not%20found"
else

The username is ok, check the password:

```
real_password = trim(passSet("password"))

      if password = real_password then
      ' pass o.k
      response.cookies("registered")= username
        Response.Cookies("registered").Expires="Sept
      6, 2002"
      Response.cookies("loggedin")= true
      Else
```

Username is ok, but password is incorrect:

```
'response.redirect
"loginerror2.asp?error=Wrong%20or%20unknown%20pass
 word"
end if
end if
%>
```

All information is correct, sends the user to the members area.

```
<%response.redirect "members.asp"%>
<%

'close it off
passSet.close
set passSet= nothing
%>
```

This include file, checks that the whole database connection is closed.

```
<!--#include file ="config-end.asp"--->
```

Security

When you log in correctly, a cookie named *loggedin* is installed on your computer. Every time you enter a new page within the members area, your browser constantly check if the cookie is still there. If the cookie is not found, you're told that you don't have the authorization to enter the page, and are asked to log in again. When you log out, the cookie is removed.If your computer crashes or you close the browser without logging out, the cookie is also automatically removed, and you have to log in again to continue. Nor can you access restricted pages located in the cache by using the browser's back button if you're logged out.

The ASP code located in the beginning of every member-restricted file:

```
<%
dim loggedin, registered
loggedin = request.cookies("loggedin")
registered = request.cookies("registered")

if registered = "" then
Response.Redirect"login.asp"
else
if loggedin = "" then
response.redirect
      "login.asp?error2=%20Please%20login%20again "
end if
end if

%>
```

Important: As a secondary security precaution, another cookie will be installed on your computer the first time you log in, and it will not be removed when you log out. As you can see from the script above, the cookie is named registered, and the script checks that you have both installed. The "registered" cookie expires on a given date, defined in the register-action.asp file. The code line under prevents the cookie to be deleted before the date it is set to expire.

```
Response.Cookies("registered").Expires="Sept 6, 2002"
```

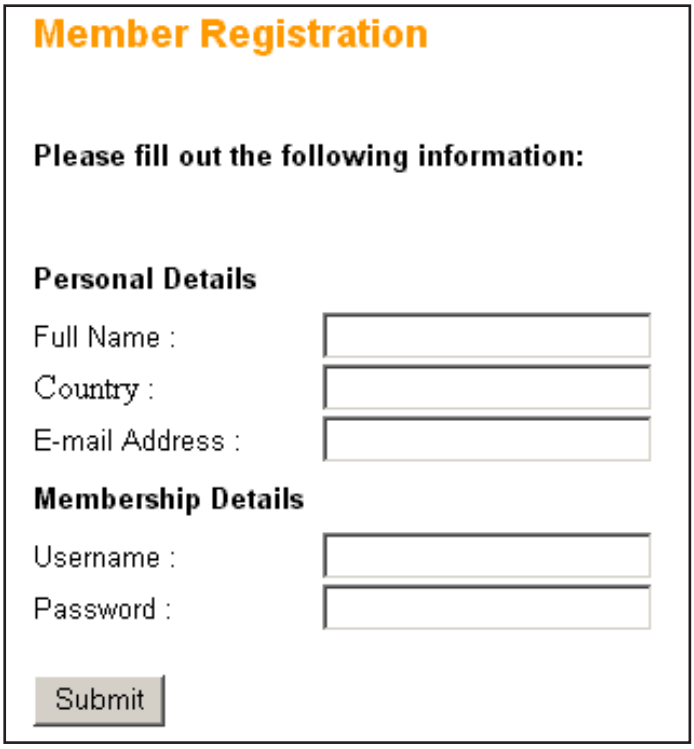
The expiration date can be set to any date, but it must be entered in the same format as above (month day, year).

This cookie is just a pointer, it makes the system recognise the member, and makes the logon procedure faster.

Member/administrator logout

The logout link is present on every member page. The ASP code simply removes the "loggedin" cookie, and redirects you to the home page.

Membership registration
Registration of new members can only be done from the administrator area. You have to fill out a form with the required information, which is submitted to a "register-action.asp" file, that inserts the information into the database.n form.



registration form

All fields must be filled in, or an alert box will pop up. The ASP script checks if the submitted username and password is not taken by any other member, and if the new member already has a member account.

```
<%@ LANGUAGE=”VBSCRIPT”%>
<% ‘on error resume next %>

<%
Dim username, full_name, email, password, country
Dim memberSQL , checkSet
```

The code above declares variables used, and the code below puts form content into the variables, and opens a database connection with the include file:

```
username = request(“username”)
full_name = request(“full_name”)
email = request(“email”)
password = request(“password”)
country = request(“country”)

 %>

<!---#include file =”config-start.asp”--->
```

The next code shows an example where it is controlled that the username doesn’t already exist:

```
<%
set checkSet = canvasDB.execute(“select * from members
where username=’”&username&”’”)
if checkSet.EOF then
else
                        Response.Redirect
”default.asp?error=Username%20Taken%20,please%20
choose%20another”
end if
checkset.close
set checkset =nothing
%>
```

Then the variables are inserted into the respective fields in the members database.
(See fig. 2 page 53.)

```
<%
memberSQL = "insert into members "
memberSQL = memberSQL & "(full_name, username, pass
word, join_date, email, country)"
memberSQL = memberSQL & " values ('"&full_name&"',
      '"&username&"',  '"&password&"', '"&NOW&"',
      '"&email&"', '"&country&"')"
SQLstmt = SQLstmt & "'" & Country & "'"
%>
```

Sets a cookie so the "site knows" if the new member has registered

```
<%
canvasDB.Execute(memberSQL)
'Response.Cookies("registered")= username
'Response.Cookies("registered").Expires="Jan 1, 2002"
%>
```

If the registration processes without any error messages, a confirmation tells that details for 'username' have been saved, and refers to the members area where he can log on to continue.

3.5 Publications

We have made a system for publishing news articles, info about events, press releases and info about reports for sale. The system for free reports is a bit different since actual files are uploaded, and not just text.

The appearance of the different categories is also very similar. If you enter the news section, the five latest news articles are shown with a headline, a short ingress and a *read more* link that shows the whole article. The date of publication is also shown. To make the navigation easier, the ten latest headlines are listed on the right hand side when you're on the page showing the whole article.

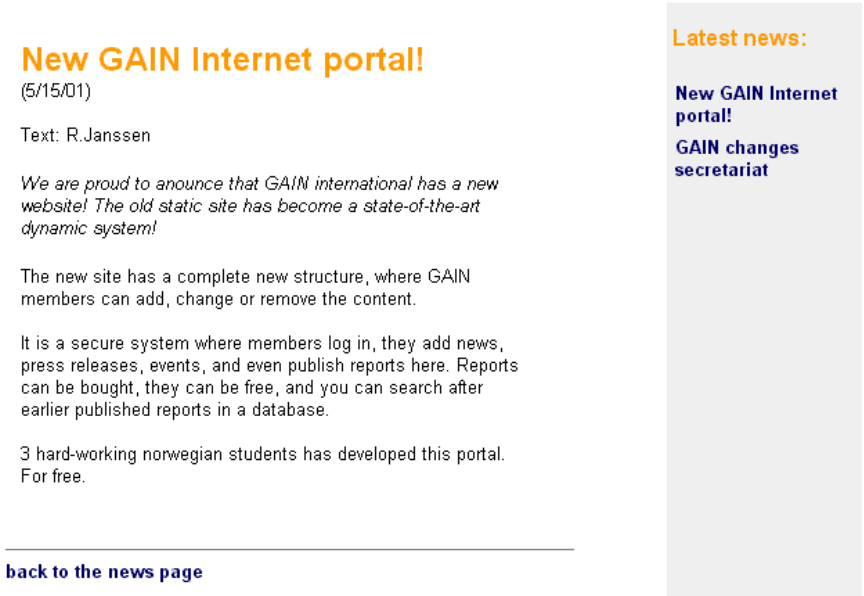


fig.3, extract from the news section.

This structure is repeated on the other publication sections, in some instances with a slight difference.

The page where you publish for instance news is also a form where there are different fields for headline, ingress, author and body copy. The date is automatically generated. When you submit the form, the information is sent to a file similar to the register-action.asp file.

The new information is given a unique id, the last publications id number plus one. This is because the table in the database requires a primary key, and the id is used to list the publications in the correct order, latest publication first:

```
<%
set rstemp=conntemp.execute("select top 1 * from
reports order by Nr DESC") %>
<% do while not rstemp.eof %>
<% Nr = rstemp("Nr")+1 %>
<% rstemp.movenext %>
<% loop %>
```

Then the content from the form is put into variables and inserted into the database.

```
<%
 date = request.form("date")
 name = request.form("name")
 over = request.form("headline")
 ingress = request.form("ingress")
 text = request.form("text")
   %>

<%
insString ="INSERT INTO reports VALUES ('"&Nr&"',
'"&date&"',          '"&name&"', '"&over&"',
'"&ingress&"', '"&text&"')"
 Set insQuery = conntemp.execute(insString)
 set insQuery = nothing
%>
```

You will then see a confirmation that the news article has been successfully published, and the headline is added to the list on the right hand side.

This code shows the ten latest news articles as links, ordered by date:

```
<%
set rstemp2=conntemp.execute(”select top 10 * from
reports order by date DESC”) %>
<% do while not rstemp2.eof %>
<%
 number = rstemp2(”nr”)
 overskriften = rstemp2(”header”)
%>
<a          href=”readnews.asp?
id=<%=number%>”><%=headline%></a>
<%rstemp2.movenext%>
<%loop%>
```

The same code is used for listing the five latest news, here you just add `<%=ingress%>` to show the ingress under the headline. The `<%=headline%>` is removed from the link tag and shown in increased font size.

In addition to close the database connection the defined variables rstemp and rstemp2 must be ended at the end of the code.

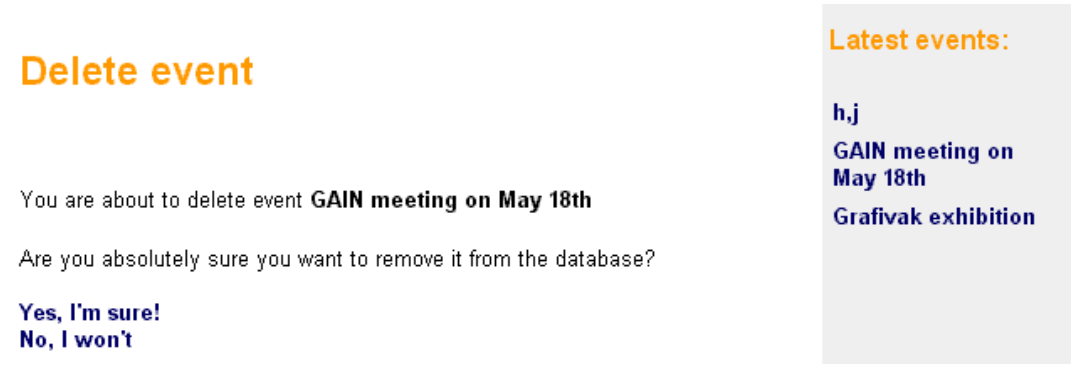
```
<%rstemp.close
set rstemp=nothing
rstemp2.close
set rstemp2=nothing
conntemp.close
set conntemp=nothing%>
```

Every publication section has a link to a help page, which is a short manual, because using HTML tags in the form can influence the layout on the screen.

Free report publication
The difference of uploading free reports and reports for sale is a browse form, where you find the relevant file on your computer, and uploads it in addition to the descriptive text and author. Only the link is inserted into the database, the file itself is saved in a folder named "files". The list shown of the free reports database contains the filename which is the link that downloads the report, the author, and a short description of the report, based on the same ASP code as used in reports for sale section.

Delete publications
A delete option is placed at the same place as the publish link on each section. When you select this option, all the headlines from the relevant table in the database is listed, and you select the one you want to delete. If you want to read the publication before you delete it, the latest events, for instance, is listed on the right hand side so you don't have to leave the delete section to read it. After you selected the link, you're told exactly which publication you are about to delete, and you have to confirm to proceed. Then a "file deleted" confirmation is shown, and you can see on the right hand side that the deleted file does not longer exist. We made the delete procedure like this in order to prevent wrong files being deleted, and to give the member a strong visual feedback.

All files are here listed in reverse order, sorted by the oldest date, since it is most likely that one should delete the oldest file first.



fig, 4, extract from the delete procedure

This code gets all the files from the table named events, sorted by oldest date with the header as a link to the next step, deleventconfirm.asp, shown in fig. x.x

```
<%
set rstemp=conntemp.execute("select * from events
order by date  ASC")
%>
<% do while not rstemp.eof %>
<%
nr = rstemp("nr")
date = rstemp("date")
header = rstemp("header")
%>
<a    href="deleventconfirm
.asp?iddel=<%=nr%>"><%=header%></a>
date: <%=date%>
<%rstemp.movenext%>
<%loop%>
```

This page just show the file you selected, with the confirmation choice.

```
<%iddel=request.querystring("iddel")%>
<%
set rstemp=conntemp.execute("select * from events
where nr="&iddel) %>
<% do while not rstemp.eof %>
<%
        nr = rstemp("nr")
        header = rstemp("header")
%>
```

If you confirm, the delete action will be executed. If you choose no, you will return to the delete overview.

This code deletes the selected event:

```
<%iddel=request.querystring("iddel")%>
sqlstreng = "delete from events where nr="&iddel
set rstemp=conntemp.execute(sqlstreng)
%>
```

It is important not to empty the whole database table, the ASP code presupposes there is an id number different from zero present. If the table is empty nothing can be inserted. If this happens, you have to open the Access database and the relevant table, and then type something in the fields manually and give the "nr" field a value.

The search engine

Since the free report and the report for sale sections may tend to contain several more files than the others, we inserted a search engine here so it will be possible to look up earlier published reports. You can search both by author and file description. If you enter the "%" sign the whole database will be listed. A new engine is inserted on the search results page, so you can execute a new search without going to the previous page.

First the variables are declared.

The query from the search engine:

```
search=request.querystring("SearchText")
```

The URL of this page so the form will work no matter what this file is named.

```
Dim strURL


Dim cnnSearch
Dim rstSearch
```

Path to our Access database (articles.mdb) file:

```
Dim strDBPath  path to our Access database
(articles.mdb) file

Dim strSQL
```

The text being looked for:

```
Dim strSearch
```

Retrieve the URL of this page from Server Variables:

```
strURL =  Request.ServerVariables("http://
prosjekter.hig.no/gain/siste/search.asp")

strSearch = Request.Form("search")
%>
```

This code shows the search results:

```
<%
If strSearch <> "" Then

strDBPath = Server.MapPath("articles.mdb")

Set cnnSearch =
Server.CreateObject("ADODB.Connection")

cnnSearch.Open "Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=" & strDBPath & ";"
```

Build our query based on the input.

```
strSQL = "SELECT nr, date, header, name, ingress,
text " _
& "FROM reports " _
& "WHERE header LIKE '%" & Replace(strSearch, "'",
"''") & "%' " _
& "OR ingress LIKE '%" & Replace(strSearch, "'", "''")
& "%' " _
& "ORDER BY date desc;"
```

Execute our query using the connection object. It automatically creates and returns a recordset which is stored in the variable.

```
Set rstSearch = cnnSearch.Execute(strSQL)

%>
<%
Do While Not rstSearch.EOF
%>
<a href="summary_mem.asp?id=<%=
rstSearch.Fields("nr").Value %>"><%=
rstSearch.Fields("header").Value %></a><br>
<%

rstSearch.MoveNext
Loop
%>
<%
```

Close the recordset and database connection

```
rstSearch.Close
Set rstSearch = Nothing
cnnSearch.Close
Set cnnSearch = Nothing
End If
%>
```

3.6 Administrator area

There are only three functions in the administrator area, *View and edit member details*, *E-mail a member* and *Register new member*. We have already described the last one.

The e-mail function gets the e-mail value from the members database, and generates a <a href="mailto: <%=member%>"> tag.
View and edit membership details simply gets the whole table with all information.This function allows editing, and automatically inserts the new variable into the members table. The administrator can also discontinue information about memberships from here, by removing the entire column from the table.

Delete	Id	Username	Country	Password	Email	Join Date	Full Name
☐	83	zlatni	Norway	zlatni	sti_brae@hig.no SAVE CANCEL	5/21/01 2:39:49 AM	Stig Brænden
☐	79	test	Europe	test		5/15/01 10:16:13	GAIN Member

fig. 5, extract from the *View and edit member details function.*

To enter the administrator page you log in with admin both in the username and the password field. This is just temporary logon information, but the changes has to be made in the member database's admin table, and a variable in login-action.asp. When you log out as administrator, you are redirected to the default start page.

3.7 Browser testing and file structure

Browser testing
The prototype is tested with Internet Explorer 4.0 – 5.5, Opera 5 and Netscape on both pc and Macintosh platforms. Except a small layout bug in Netscape 4.7 for Macintosh, the prototype looked good and seemed to function correctly on both platforms with different browsers.

File structure
Since the prototype contains a lot of asp files, we have tried to give each file a logic name**.** Since the web site is divided between members and non-members, all files belonging to the members area has the extension _mem.asp. All the dynamic functions files are also named after a template.

4 discussion

4.1 How can the product be used

The web site in the state it is now
The GAIN web site we have made is a fully functioning prototype, except for one function. That is, the edit function. It is unfortunately not possible to edit articles when they are already published. The current solution in order to edit, is to delete and republish.

It will probably also need a more user testing and adjustments before it is completely finished. It needs more testing to make sure all the links and functions works, as they should. Because of the number of ASP pages it isn't 100% certain that all of the bugs are fixed. By letting users (in this case the users, i.e. the members of GAIN) test the web site we will find out if the user interface is easy to learn and use. When the users have started using the web site we will also find out if the MS Access database will be a bottleneck or not. We chose to use a MS Access database that may have some restrictions when there is a high load. A high load means several users interacting with the database at the same time. Considering the relatively few members of GAIN this won't be a problem.

Things that needs to be done before GAIN can start using the web site

As mentioned earlier the web site can be used in its current state. But there are some aspects that should be looked into before it can be considered complete in accordance with the requirements. First of all: A thorough user testing should be conducted. We have tested the page quite a lot here at Gjøvik College, but we are still not completely sure of that every bug and usability problem has been identified. And at Gjøvik we haven't had the opportunity to test the web site on slower Internet connections like ISDN or for instance a 28kb modem. This is because we are working on a broad band with very high transfer rates. It is therefore important to conduct some tests on slower connections, just to see that everything is working well.

Since the web site uses ASP, it is essential that it should be located on a WinNT server with IIS. To find an Internet provider that offers these specifications shouldn't be a problem since ASP is a commonly used technology these days. The only drawback could be that some Internet providers charge extra for running IIS on the server.

Since this web site is a prototype and is not to be published on the Internet yet, we have not implemented Meta tags. In HTML code that enables search engines like AltaVista, Hotbot and Lycos etc to find the web site easier when the users search for GAIN, or some other words related to GAIN. We have solved this by implementing the code needed in the .asp files, and the only thing that needs to be changed are the keywords within the Meta tags. These lines of code are located at the top of every .asp file. The .asp files can be opened and edited like an ordinary HTML page, the only difference is that the source code can't be viewed in a browser. It is also very important that the parts of the file with the ASP code remains intact.

An administrator also needs to register all the members in the admin section.

And finally: In the file ../admin/login-action.asp at the top of the code you can specify the username and password for the admin section. It is highly recommended that these are changed into something else, because the username *admin* and the password *admin* is probably the first words any hacker would try if they were to illegally enter the portals admin area. In the code it is described how to do this.

4.2 What we have learned from the project

Technology
To complete this project we had to acquire knowledge in various technical fields. First of all we had to learn ASP (Active Server Pages). This was absolutely essential to fulfil the web site requirements. We also had to learn more about java scripting. The java script was used to ensure that all fields are filled in, in the section for registering new GAIN members.

Typography
To ensure that the appearance of the web site was up to the standards you would expect from a web site promoting Graphic consultants, we had to acquire knowledge in the field of typography for the Internet. We had already been through courses here at Gjøvik College that covered the field of typography, but we still felt that we should learn more.

Human aspects of working in a group
When working together on a project in a group like this, there are several aspects that one must have in mind. First of all we've all experienced during this project that it isn't always that your opinions goes through and you have to let the majority decide. This is perhaps one of the biggest differences between working alone and in groups. When you are working in a group you just have to let the democracy rule, or you would soon end up in chaos.

In this project we also learned soon that most things you plan to do, tend to take more time than originally planned. This is a valuable lesson to bring along for future projects, at least make sure that you have enough time at the end of a project.

We have also learned a lot about the importance of organizing a project like this. Good organization is absolutely essential for completing a project within the deadlines that are set.

4.3 Discussion

Related to the work process in the group

Time plan
One of the largest problems we encountered was the lack of time towards the end of the project. Evaluating our progress plan, we see that we should have started earlier to work on the actual product, the Web site. We realize that in the first stages of this project we spent too much time on the research part. Of course we learned a lot, and that was important enough, but we could have been more effective in our work, and perhaps everyone didn't have to do his research at the same time? In the end of the project this would have given us more time to fix errors and to conduct more thorough user testing.

User testing
The user testing is a very good method, not only to discover errors and bugs in the product, but also to get some external feedback on the user interface in the product. By user testing we mean testing conducted by people that haven't been involved in the project, and are typical users for the product. For instance: Is it obvious to all users where they have to navigate on the web site to publish a free report? Is the way the navigation is constructed on the web site user friendly at all? By allowing external users to test the product, and by observing them at the user interface, we would get some answers and feedback that are very valuable to us as the developers. This is mainly because we are working with the product all the time and will have no problem understanding the structure and what to do where. And therefore they will notice and point out things that don't work or solutions that aren't logic, which we, as the developers, don't see. As mentioned earlier, we didn't get enough time to conduct the user testing as we wished to do it.

When we made the time schedule (progress plan / GANTT) for the project we should have scheduled more time for the user testing. For later projects this is very important. For instance: The testing we managed to conduct ourselves revealed errors and logical failures in the web site, which we hadn't even thought of. And it seemed like the more we tested, the more errors there were. We think we have discovered the majority of the bugs and fixed them, but as mentioned earlier, we should have had more time for the testing, and then we would have been more secure that the product would work properly under any circumstances.

However, at least an expert usability evaluation was conducted by our advisor. He pointed out some aspects concerning the layout and the structure that could have been done better. We then corrected most of the things he pointed out, but some we didn't have the time to fix. These things are described later in this chapter, under the title *The Web site*.

Taking notes

We also realize that during the different stages of the project, we should have been much more consistent in taking notes. Of course we have our minute book, and our status reports, but it would have been helpful now, when we are writing the final report, if the notes we made had been more complete. To illustrate an example: In the earlier stages of the project we discussed many things and problems with our project, which we at that point meant would be worth mentioning in the final report. Such little things could have made the final report even more complete.

Status meetings

Another thing we probably should have done better was the status meeting. We had an internal status meeting every week, but sometimes we forgot it and sometimes we didn't have time to arrange a meeting. So the solution could have been to set the meetings to the same time every week. In this way everyone on the group would have known when the meeting were and we could have arranged our time schedule better. This wasn't actually a big problem, but still worth mentioning, because it could have been done better. The same goes with the status reports. They should have been more thorough, so that we could have had a more precise and accurate apprehension of where we were in the project, and which "loose ends" we still had.

Project manager

We also see now that we should have elected a project manager at the very beginning of the project. In the beginning we believed that we didn't need a project manager, because of the relatively small size of the group. But as we had worked on the project for a while we realized that it would probably be a good idea to appoint a group leader/ project manager. This could have helped us to organize our project work better. For instance: At the beginning of the project we didn't have a clear definition of which group member that had the responsibility for what, i.e. you could say that everybody was in charge of everything. And clearly that wasn't any good solution. So in the midway of the project we appointed a project manager.

Apportioning the work

Especially in the beginning we didn't have a good organization of the work. We simply weren't able to distribute the different work tasks on the different group members. This

is probably related to the fact that we didn't have a project manager. The project would have run more smoothly if we had managed to organize the work among a bit better. However, at the end of our project things have worked better.

Milestones
We also realize that during the initial process of the project we should have made more accurate goals and milestones for the project. In our pre project we made some deadlines, and those were good deadlines, but they were too few and too general. We should have divided them into minor deadlines. This would probably have made it easier for us to keep track on the progress of our project.

The web site

The Layout
One of the things we feel that could have been done different concerning the layout is the location of the links for deleting and publishing events, news, reports and press releases. The links are now placed under their appurtenant link. To illustrate an example we could say that a member would like to publish a new press release. Then he'd have to click on the press release button, and then go to the bottom of the page. There is the links for deleting and publishing located, and he could click on publish press release.

We have received some feedback from various users that have tested the page, and one of the first things they mentioned was that these links was a bit hard to find. This means that they probably should have been located somewhere else. Our project advisor also pointed this out to us. One of the solutions we considered was to place them

in a menu at the left side in the user interface. In that way the members would have seen the links as soon as they log in and enter the members section. That was also the suggestion from one of the users that tested the site. But there were some aspects with that solution that made us turn to the solution with the links lying on the appurtenant page. One of these aspects was that the number of links in the menu would have been close to twenty, and that would have been far to many, with the user interface we have now. We could have made a somewhat different solution with the buttons in the menu, and made them so that the appurtenant links would have popped up when the mouse was hovering over the button. The main problem with this solution is that it would have crashed a bit with the design of the user interface. But still if we had had the time to redo it, this would probably have been the solution.

Another idea would have been to place them in the grey area that we have on the right side of the user interface. That could also have been quite nice to look at, and easy to use. But the drawback with that solution is that on some member pages we have listings of news, events and press release that appear at the top of the same grey area. And it is not a good solution to move the links around; they should be in the same place all the time. It can be confusing for the users if the navigation panels move around depending on where they are on the web site.

In the end it works quite well the way it is now, it is only for new members this could have been a problem. And since the member section is only to be used by relatively users, we believe that this problem will be minimal.

Minor comments on the user interface
In the user interface there are some minor details that could

have been changed in order made the web site look a bit cleaner and more pleasant. For example: In the grey area at the right side of the page, we have, on some pages, a column that lists the last event, news, pressreleases etc. Our advisor commented that this lists could a need bit wider left margin. We agreed, but we didn't have the time to fix it, since this would involve correcting the code in too many files.

We could also have needed some further co-ordination regarding the use of colours on the different text types in the different pages in the web site. Testing and discussion with our advisor has revealed some inconsistency. We did correct the major problems with this, like the headlines in the news, events, and press release section. But still there are some minor differences that we could have adjusted, like the appearance of the search fields in the *free publications* section and in the *publications for sale* section.

The Member section

It would probably also been better if the layout on the member section had made it more clear that they were actually logged in and were in the members section. In the way it is now, the members are welcomed to the member area, just after they have logged in. After that there isn't really anything that says where they are, apart from the fact that the login fields disappear at all time when they are logged in.

The buttons in the menu/navigation panel seemingly stays the same, but the buttons will contain links that are exclusive to the members.

The Admin section

In the administrator area we have changed the design a bit. This was to separate it from the overall design in the member and non-member sections, and to give room for the table that contains the member details. Optimally we probably shouldn't have differentiated the design as much as we did, but this is a minor issue since none of the members will use the admin section much. The section will mostly be used for registering new members and change other member details such as e-mail addresses, usernames and passwords.

The structure/configuration

As the project moved forward the site just grew and grew, and we have to say that the final result became larger than we expected. We produced well over hundred .asp files and all files contained links and graphics that had to be addressed correctly in the source code. To deal with these large number of files, we made a template, which we all used when producing the different files. This saved us for a lot of work, with the different pages. But we realized that we should have made at least two, and perhaps even three different templates, one for the non-member section, one for the member section, and perhaps one for the admin section. This is due to the fact that the non-member section and the member section contain different links, and the login fields are absent on the member-files. And that the admin section is completely different from the other two sections, with an own menu and a bit different layout. Since we only had one template, we did have to change the source code on many .asp files, which took quite some time. With three templates we would have saved time, which we for instance could have used for testing the web site. In addition to the three templates we could also have made a file, which contains the declaration of all the variables used in the asp code.

The declaration file

When developing large web sites with many .asp files, it is a good idea to use a file to declare all the variables This was something we learned during the work with asp, and it would really have helped us lot, and saved us much time. The concept works like this: In a system with many similar files, which uses .asp code, it is possible to use one or perhaps two or three files, where you declare all the variables which are used in all the files. So instead of typing the path gain/members in all the files where the path is used, you could use a variable named pathmem or something similar. Then the file where you declared all the variables would have to be included at the top of each .asp file. In this way it would be easy if you during the project were to change the directory names or move the whole site to another physical location on the server. Then you would just change the variable in the declaration file, and it would be automatically updated in all the other files. To optimise it even further we probably would have made three files with declarations, one for the non-member section, one for the member section, and one for the admin section. If we had learned about this method earlier in the project, we would have done it like that. It would have saved us for a lot of work, since we during the project realized that we had to change the structure of our directories. But instead we had to change the source code for every file that was affected by the structure change, instead of just one, the declaration file. Obviously this is an important experience for later projects like this.

Frames

Another solution to the problem with many files, which had to be updated, could have been to use frames instead of tables to navigate on the web site. With five frames as the basic structure of the user interface, the menus would have stayed static in the left frame as long as the users stayed

in either the member section or in the non-member section. We would have placed the menu in the left frame, the grey area with the listing of news, events etc. in the right frame, and the GAIN banners at the top and bottom frames. That means that the only frames changing would have been the one in the middle and the on at the right side.

As mentioned earlier this would have made the coding much easier for us. However, the reasons why we didn't use frames, was the fact that pages with frames takes longer time to download – at least the first time you dowload the page – and several other well known usability problems with frames. See for example Nielsen 1997.

The Uploaded Files

On the site the users have the possibility to publish news, events and press releases. They can publish an event and then later they can delete it at any time. But there is one drawback with this system as it is now. The users can't edit the already uploaded files. To illustrate an example: A GAIN member has published an announcement on the web site that he will give a lecture about digital printing machines. He then realizes that he has announced the wrong date for the lecture. Then he would have to delete the entire text he published, and publish it again with the correct date. He will not be able to change the existing event. Because he has the option to delete the text, he will still be able to correct it, but with a time consuming and certainly not optimal procedure. This is a feature that we would have liked to implement in the web site, but at the end we simply didn't have the time to get it done.

XML

GAIN wanted us also to explore the possibillity for using XML on the web site. One of XMLs advantages is the

abillity to search the content on a page. Since our prototype uses databases, the search actions are also conducted within the databases, and not on the actual web site. This eliminated the main advantage with XML, and there were no need to explore that option any further.

Selling PDF articles on the web

Om the web today there are many sites where you can purchase products, information and sevices. Many Internet portals today also offer the possibillity to purcahse reports or articles in a PDF format. Often the visitors to the site or portal can search in a database and when they find articles or reports that interests them, there are several several different solutions to manage the purchase. We will here describe two different approaches.

Email request to the owner
When the visitor finds an article that he want to purchase, he can be passed on to a page where he will be informed of the prices and he will then be asked to to fill out a form, where he describes which article he's interested in, and the form will then be sent as an email to the owner of the article. Then the owner will take contact with the buyer and they can complete the purchase intenally. This is not a good solution, if you're selling quite a lot of reports. Then there are other solutions that are much better.

Adobe PDF Merchant
And one of them is the use the software from Adobe, Adobe PDF Merchant. Adobe PDF Merchant is used togehter with transaction servers, and is an secure solution for distributing digital content. The program works in the way that the user can purchase a license to unlock and view

locked PDF documents that the owner has on his web site. Without the license it impossoble to view the documents. The content owner can also specify what rights the buyer will have. That means that he can specify if you are able to print the document, copy the content or edit the document.

5 conclusion

5.1 Conclusion

During this project the intentions was divided into three sections.

I. **The web site should combine the HTML technology with newer server based technology.** Our solution was to use ASP (Active Server Pages) together with HTML. This combination offers all the features, which are needed to fulfil the requirements for the web site. We chose ASP over some other technologies like CGI and PHP because it is a relatively new technology, it is fairly easy to learn for people with basic knowledge in programming, and because it runs on WinNT platforms.

II. **The web site should interact with databases.** We chose to use Access databases because ASP works well with Access, and because we already had the knowledge needed to employ it. The use of Access databases is not an essential element of developing this web site. It might as well have been other similar database platform like MySQL.

III. **The prototype should provide a foundation for the new GAIN web site.** We feel that the prototype we developed is very close to a complete product. The web site can be used in its current state. The only feature that isn't implemented is the possibility to edit the uploaded files (news, events, press releases, and the free reports).

We are satisfied with the results and it is in our true belief that the product will satisfy the demands of GAIN.

WeGa signing off...