Flash som teknisk handbok

- Hur Flash lämpar sig som utvecklingsplattform för presentation av stora mängder HTML-formaterad information?

LUNDS
UNIVERSITET
Lunds Tekniska Högskola

LTH Ingenjörshögskolan vid Campus Helsingborg
Multimedia

Examensarbete:
Marcel Svensson
Peter Björneskog

LTH Ingenjörshögskolan vid Campus Helsingborg
Lunds Universitet
Box 882
251 08 Helsingborg

LTH School of Engineering
Lund University
Box 882
SE-251 08 Helsingborg
Sweden

Tryckt i Sverige
Media-Tryck
Biblioteksdirektionen
Lunds Universitet
Lund 2007

Sammanfattning

Böcker, dagstidningar och lagrad information, t.ex. cd-skivor, fyller inte sin funktion som det självklara valet för en stor del i dagens moderna samhälle där det ska gå fort att få tag i uppdaterad information. Valet för många är Internet, där man kan få tag i aktuell information endast genom att par enkla musklick. Det är inte enbart för individen möjligheterna finns att ta del av den senaste information utan även för företagen. Företag kan enkelt nå ut med den senaste information om t.ex. deras produkter i en enorm uträckning jämförelse med vad man kunnat tidigare. Företagen kan även få direkt feedback från dem som använder sig av deras produkter utan att vända till varje enskilt individ.

SWEP var det företag vi kom i kontakt med och de är världsledande inom forskning och produktion av värmeväxlare. De har sitt huvudkontor i Landskrona och har liksom många andra företag som arbetar med tekniska lösningar mycket dokumenterat material om sina produkter. Detta material innehåller teorier och fakta och olika aspekter på hur värmeväxlarna arbetar och fungerar. De är sammanställda som tekniska handböcker uppdelade på olika område. Dessa handböcker finns i form av häfte men även som en applikation som man kan ladda ner från deras hemsida. Genom möte blev vår uppgift klar, SWEP ville att vi skulle ta fram en ny lösning där informationen presenterades på Internet, tillgängligt för alla. Önskemålet var att vi skulle använda oss av programmet Flash och att användarna skulle kunna interagera med det på något sätt.

Syftet med detta examensarbete är att ta reda på om Flash är en lämplig plattform för att presentera stora mängder information och om Flash har det stöd som behövs för att integrera användaren. All information är sedan tidigare HTML-formaterad och det skall kunna utnyttjas.

Under arbetsgång dras många oväntade problem upp till ytan dessa har självfallet också dokumenterats. Bland annat tolkade Flash inte HTML som en vanlig webbläsare och detta ledde till konsekvenser. Alternativa lösningar har tagits fram och andra har förkastats.

Resultaten blev en grund i Flash som innefattar menyer medans innehållet presenteras i iFrames för att behålla minnet i navigation för respektive bok. Tack vare detta har undersökningen i om hur Flash lämpade sig som plattform besvarats, använd Flash som inslag på sidor som små animationer eller kanske spel eller bildvisare. I en applikation där innehållet är formaterat som HTML bör man använda en plattform som har allt stöd för det.

Nyckelord: Handböcker, Flash, iFrame, ASP.NET, SWEP

Abstract

Books, newspapers or stored information, such as CD's, are no longer the primary tool in today's society where quick accesability of information is a big priority. Internet is the obvious choice where you can reach information that is up to date with only a couple of mouse clicks. And it's not only for the induvidual but also for the companys. It's much easier today for a company to reach out with freash information to it's customers than before. They can also get quick feedback from those who use their products without having to ask each one induvidually.

SWEP was the company that we got in touch with and they are world leading in research and production of heat exchangers. They have their HQ in Landskrona, Sweden. Like many companies before them, who produces techical solutions, alot of documented information on their work. This material contains theories and facts on different aspects around heat exchangers. With that information SWEP created the technical handbooks about heat exchanging each one specified to a certain aspect. These handbooks exist in the form of booklets and also as an application that the user can download. Throughout a meeting our task was formed, SWEP wanted us to create a new solution where the informaion will be available on the Internet. They also wanted us to use the program Flash throughout development and to somehow make it possible for the user to interact with it.

The agenda is to find out wheater or not Flash is a suitable platform for presentate a large amout of information and if it is possible to make the user interact with it. All the information from the booklets is formated as HTML and we tend to use it like that.

Throughout our work many unexcpected problems occured and all of these have been dealt with or another solution was used to go around it. The biggest problem was that Flash didn't interprete HTML as a normal browser would and that brought consequenses.

This resultet in an alternative solution where Flash presentet the navigation system while the content is presented in iFrames to keep track of where in the hierarki the user currently are for each book. Our conclusion is that Flash shouldn't be used in this type of application where the primary objectiv is to display information. Use Flash as illustrations, games or smaller applications such as picture viewers included as elements on websites.

Keywords: Handbook, Flash, iFrame, ASP.NET, SWEP

Förord

Tack till Mats Jönsson för kontaktuppgifter, utan dig hade det här arbetet aldrig blivit av. Vi tackar teamet på SWEP bestående av Henrik Ewetz, Victoria Hamnäs, Anders M Persson, Tobias Persson och Lisa Magnusson, för att ha givit oss denna möjlighet. Ett sista tack går till Christin Lindholm för handledning.

Innehållsförteckning

1 Inledning

1.1 Bakgrund

Böcker, dagstidningar och lagrad information, t.ex. cd-skivor, fyller inte sin funktion som det självklara valet för en stor del i dagens moderna samhälle där det ska gå fort att få tag i uppdaterad information. Valet för många är Internet, där man kan få tag i aktuell information endast genom att par enkla musklick. Det är inte enbart för individen möjligheterna finns att ta del av den senaste information utan även för företagen. Företag kan enkelt nå ut med den senaste information om t.ex. deras produkter i en enorm uträckning jämförelse med vad man kunnat tidigare. Ett sätt företag kan dra nytta av Internet är när de vill tillhandahålla tekniska handböcker till sina kunder.

Tekniska handböcker fungerar som ett bra medium för att lära sig eller uppdatera sin kunskap. Böckerna består av ganska rå fakta och med illustrationer för att visa enkla exempel.

Problemet med böckerna är att de inte anpassas för Internet utan har samma struktur som i tryckt form. Dessutom utnyttjas sällan möjligheterna som tekniken medför. Det naturliga steget blir att göra en mer dynamisk handbok där användaren får en aktiv roll.

1.2 SWEP

SWEP är ett företag som grundades 1983 och är numera ett av de företag som är världsledande inom utveckling och tillverkning av värmeväxlare. Huvudkontoret ligger i Landskrona där det finns både tillverkning och forskning. SWEP har även fabriker runt om i världen, bland annat i Schweiz, Slovakien, USA, Malaysia och Kina. SWEP ägs sedan 1994 av det Amerikanska företaget Dover Corporation. [1]

Värmeväxlarmarknaden är uppdelad i tre delar:

Refrigerting - Kyla

”Kylteknik omfattar alla typer av applikationer där ett köldmedie som används för att alstra kyla eller värme. Typiska användningsområden är luftkonditionering, kyl-och-frys, värmepumpar och lufttorkare.” [2]

Heating – Värme

”Generellt sett avger eller använder alla processer värme i någon form. Att utnyttja den energin så effektivt som möjligt är en huvudfråga inom applikationsområdet. Typiska användningsområden är värmepumpar, värmepannor och värmecentraler.” [3]

Industrial – Industri

”Industriella applikationer är sannolikt det område som har de mest skilda förutsättningarna för värmeväxlingen. Eftersom varje applikation är unik, behöver lösningen i hög grad skräddarsys. Typiska industriella applikationer är kylning av smörjolja i olika motorer eller uppvärmning/kylning av medier i olika processer” [4]

Varje del har en egen teknisk handbok. I SWEPs fall utvecklades både böckerna i tryckt form samt som nedladdningsbar applikation på nätet.

1.3 Mål

SWEP önskar att få sina applikationer uppdaterade så att dessa blir tillgängliga online i form av en hemsida. Istället för att ha alla delar var och en för sig så skall dessa vara samlade på ett och samma ställe. Önskemålet är att dessa skall presenteras och utvecklas i Adobe Flash. Sedan tidigare finns allt innehåll i en databas och det är HTML-formaterat, detta skall kunna utnyttjas då innehållet inte kommer att innefattas i vårt arbete.

Funktioner som användes exempelvis bildvisare och ordförklarare behålls och eventuellt förbättras.

För att integrera användaren kommer ett forum att användas samt en betygssättningsfunktion att utvecklas. Utöver detta har vi i tankarna att utveckla ett eget avsnitt på applikation där användaren skall kunna komponera ihop sin egen text bestående av andra delar från applikationen.

1.4 Syfte/Problembeskrivning

Syftet med detta examensarbete är att ta reda på om Flash är en lämplig plattform för att presentera stora mängder information och om Flash har det stöd som behövs för att integrera användaren. All information är sedan tidigare HTML-formaterad och det skall kunna utnyttjas

Detta leder till följande problem:

- ”Är Flash en lämplig utvecklingsplattform för presentation av stora mängder HTML-formaterad text?”

- ”Har Flash det stöd för den dynamik som krävs för att integrera en användare?”

1.5 Metodik

Arbetet kommer vara uppdelat i följande steg.

1. *Samla information om:*
   - Existerande lösningar, för att skapa en uppfattning om tillvägagångssätt och eventuella problem som kan uppstå.
   - Generella designlösningar/idéer, för att bli inspirerade och influerade.
   - Färgtema och designregler, för att få en genomtänkt och användarvänlig design. (att utgå från SWEP.net:s design och övrigt SWEP material)
   - Flash kommer att användas, därför måste vi samla informationen om detta och se vad som krävs och eventuellt vilka andra program/språk som kan behövas för att komplettera.
2. *Upprätta krav*
   - Efter insamlandet av information upprättas krav. Detta görs delvis i samröre med SWEP.
3. *Skapa en preliminär design*
   - En tidig design tas fram för att stämma av med SWEP så att grafiskt språk och funktionalitet är i den riktning de har tänkt sig.
4. *Programmering*
   - Programmeringen påbörjas med valt programspråk så fort designen är upprättad.
5. *Testning*
   - Efter större ändringar ska användartestning genomföras för att få feedback på funktionaliteten.
6. *Iteration*
   - Steg 4-6 kommer att återupprepas till dess att de mål och krav som är upprättade är genomförda.
7. *Slutgiltigt användartest*
   - Efter det att kraven är uppnådda och vi är nöjda med design och funktionalitet kommer det att genomföras ett större användartest.

1.6 Förväntat resultat

Det förväntade resultatet är att kunna presentera en fungerande applikation/hemsida som innehåller samtliga handböcker.
De tre handböckerna skall utökas till fyra och bindas samman samt att användaren skall få möjlighet att påverka.

Målsättningen är att skapa en plattform/koncept som sen kan återanvändas för andra handböcker. Dessutom skall det vara lätt att utöka befintliga applikationer med nya böcker och kapitel.

1.7 Avgränsningar

- Skapa en editor för framtida handboks-skapande/modifierande
- Uppdatera befintliga texter
- Implementera innehållet för samtliga handböcker

Vi kommer endast att skapa en applikation där delar av innehållet är implementerat, införande av resterande innehåll lämnas som framtida arbete.

1.8 Målgruppen

När SWEP först utvecklade sina egna applikationer med de tekniska handböckerna gjorde man ingen fullgod målgruppsanalys. SWEP är ett buisness-to-buisness företag och handlar således nästan uteslutande med andra företag. Därför kan man utesluta att det är vanliga lekmän eller privatpersoner som använder sig av applikationerna, utan det är snarare en kontorsarbetande inköpare använder sig av dem som undersökning och uppföljning av inköp.

Med SWEPs tidigare upplägg med nedladdningsbara handböcker har det inte gått att få fram statistik om vilka som besöker sidan är, hur ofta de är där och hur länge. Allt detta hade varit intressant för att göra en grundliggare analys och förbättringar.

Kontorsarbete innebär datorvana och det är skärmen som blir det viktigaste redskapet. I kontorsguiden som SIF har skrivit förespråkas användandet av platta skärmar i kontorsmiljö eftersom dessa inte reflekterar lika mycket ljus som en bakprojektionsskärm och att den inte heller utsöndrar lika mycket elektromagnetisk strålning. De platta skärmarna är vanligt förekommande och det blir en aspekt som skall anpassas vid designen av applikationen [5]. Detta hade vi i åtanke när förarbetet till designen genomfördes.

2 Krav

Målet var att skapa en Flashapplikation som ska hämta ut information via en databas. För att kunna få en överblick över hur arbetet skall fortskrida och vilka mål man skall arbeta för bör man uppreda en Kravspecifikation. Denna kan vara uppdelad i olika avsnitt som behandlar krav på olika nivåer såsom funktionalitet, design och säkerhet. Utöver önskemål om Flash och att användaren skall integreras fick vi fria händer från SWEP.

Varje krav skall vara testbart och därför bör de vara korta och konkreta för att begränsa dess omfattning. Samtliga krav är samlade under APPENDIX A – Krav.

Exempel på krav för detta examensarbete:

A.5.4 Varje handbok skall behålla sin status om man byter bok.
A.5.5 Texterna skall vara HTML-formaterade.
A.5.6 Texterna skall ha en gemensam stilmall.
A.5.7 Texterna skall kunna använda sig av JavaScript.

I Appendix A finns en lista över utgångna krav. De finns med eftersom de togs fram som underlag för delar av applikationen som senare av olika anledningar togs bort eller gjordes om. En liten text till varför kan läsas om i 5.2 samt som en liten text i Appendix A – D Utgångna Krav

3 Design

Ett viktig steg att ta innan utveckling på en applikations design påbörjas är att ta reda på bakgrundsfakta; Finns det redan existerande lösningar? Finns det litteratur på området som kan ge riktlinjer? Är det möjligt att utnyttja denna information?

3.1 Bakgrundsinformation

Enligt Jakob Nielsen [6], som anses vara en ledande användarvänlighetsexpert, så skall man optimera upplösningen för 1024x768, men skall fungera på 800x600, och att man skall använda sig av design som inte är fryst och anpassar sig efter upplösningen, t.ex. tabeller med breddattributet satt på %.
Jakob Nielsen förespråkar även:

- Att all nyckelinformation, så som menyer, skall visas utan att man måste skrolla.
- Att informationen som presenteras skall vara lättläst i förhållande till bredden.
- Att sidans objekt är av rätt storlek och har rätt plats i förhållande till skärmupplösningen, t.ex. hamnar bilder i rätt förhållande till texten.

Enligt en fortlöpande undersökning på W3schools.com [7] genomför på deras besökare angående vilken upplösning deras användare har visade de sig att:

- 54% av användarna har en upplösning på 1024x768.
- 26% av användarna har en högre upplösning än 1024x768.
- 15% av användarna har en upplösning på 800x600.
- 5% av användarna har en okänd upplösning.

Tittar man på statistiken på ett år tillbaka ser man en tydlig minskning på användarna som använder sig av upplösningen 800x600. I juni månad 2006 använde sig 17% av upplösningen 800x600 och i januari månad 2006 var denna siffran 20%.

Självklart är W3schools.coms statistik ett utdrag på vad för upplösning en bestämd målgrupp av Internets besökare har. Men eftersom W3schools.com har ett så pass enormt besökarantal så bevisar det ändå att besökarna satsar allt mer på högre upplösningar på skärmarna.

En annan faktor som spelar in på skärmas upplösning är att TFT/LCD-skärmen har blivit en stor försäljnings succé. I ett nyhetsbrev[8] från Samsung som är en av de ledande tillverkarna av TFT/LCD-skärmar, så bytas de gamla CRT-skärmarna ut mot nya platta TFT/LCD-skärmar. Bara detta år (2007) räknar Samsung att cirka 90% av alla skärmar som säljes är TFT/LCD, oavsett tillverkare. Det största antalet skärmar som säljs är 17tums TFT/LCD och dessa i de flesta fall har en minimum upplösning på 1280x1024.

De CRT-skärmar som fortfarande säljs på marknaden har minst en storlek på 21tum och en upplösning på 1280x1024.

3.2 Upplösning och struktur

Applikationen kommer att vara anpassad för en minimumupplösning på 1024x768. Applikationens element kommer inte att påverkas av en mindre upplösning såsom 800x600. Det som kommer att hända är att applikationen kommer få en rullningslista i horisontellt läge, d.v.s. en fryst design kommer att användas.

Anledningen att en fryst design väljs är att textinnehållet struktur är mycket viktig och skall inte påverkas av lägre eller högre upplösning. Använder man en högre/lägre upplösning på en dynamisk design skulle texterna och bilderna behöva täcka ett större/mindre område och då skulle designen/formatering av texterna bli olika beroende på upplösningen.

En annan faktor som spelar in på upplösningen är att de redan existerande handböckerna är skapade för en upplösning på 1024x768. De redan existerande användarna skall inte behöva ändra upplösningsinställningar, utan de skall kunna fortsätta använda denna tjänst utan problem.

3.3 Storyboard

Vi utvecklade en rad olika designer som SWEP fick välja mellan och komma med synpunkter på, gemensamt för dessa var att de hade en liknande struktur. Storyboardet visar positioner för var de olika elementen kommer att befinna sig i förhållande till varandra.

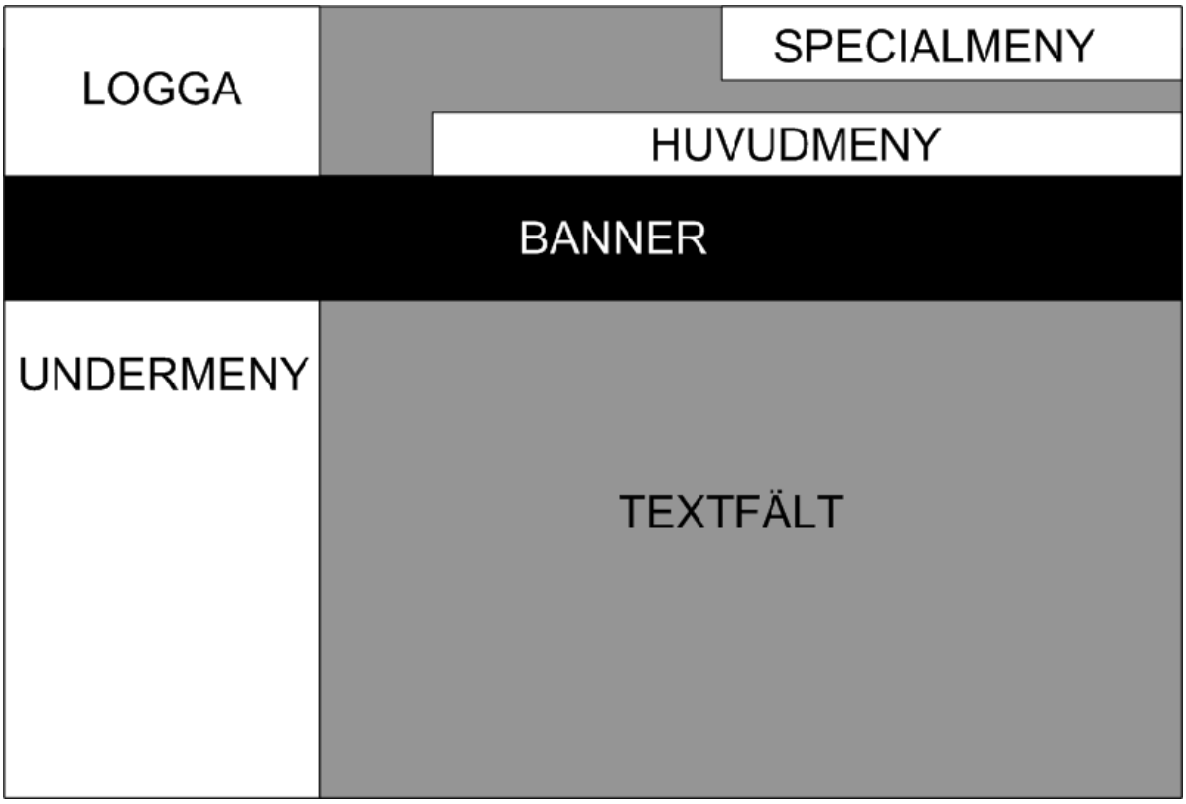

Fig. 3.1 - Storyboard

Specialmeny

Detta är en meny som består av olika val användaren kan göra som ligger lite utanför möjligheterna att titta på innehållet i böckerna. Dessa har delvis bestämts i samröre med SWEP, de som skall finnas med här är en sökfunktion, en ordlista med samtliga ord förklarade som finns med i innehållet, möjlighet för användaren att ändra typsnitt och storlek på texten, en sitemap för överblick av applikationens innehåll samt kontaktinformation till SWEP.

Huvudmeny
Består utöver de böcker som fanns innan av en länk till startsidan, en länk till ett forum samt en länk till myBOOK.

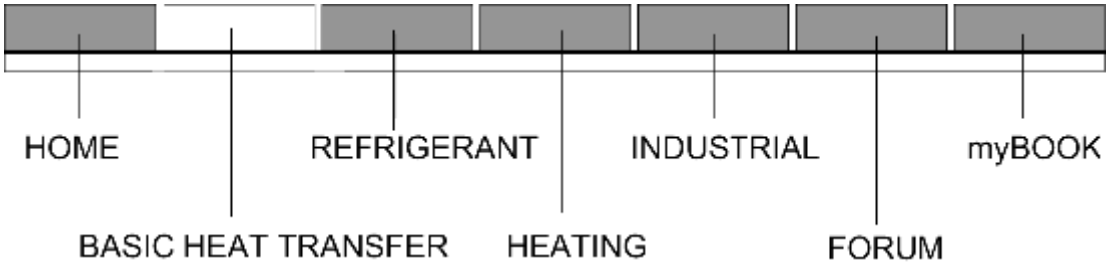

Fig 3.2 – Huvudmeny - Storyboard

Logga
SWEPs logga kommer att presenteras

Banner
För att gränsa av huvudmeny med innehållet används detta grafiska element.

Textfält
Här kommer allt material från handböckerna att presenteras.

3.4 Färgschemat

När man skall välja färgschema till vad man än gör, från applikationer till affischer, så är ett viktigt steg att undersöka om företaget redan har etablerade färger, färger som människor redan förknippar med företaget/applikationen. I SWEPs fall så har dem en väl etablerad logotyp.

Fig. 3.3 – SWEPs logotyp

Denna logotyp har varit med ända sedan företaget skapades och är något som identifierar SWEP. Det röda representerar värme och det blåa representerar kyla i en värmeväxlare. Att ändra denna logotyp eller färgschemat är något som man inte bör göra.

Att utgå från dem existerade färgerna på SWEP är därför det smartaste valet. Färgerna logotypen är uppbyggd av är blå, röd, svart och grå. Tittar man på

logotypen så ser man att nyanserna på färgerna blå och röd inte är aktuella för att skapa ett passande färgschema, för att välja rätt färg att komplettera med kan man använda sig av färghjulet och de olika modeller som man kan tillämpa.

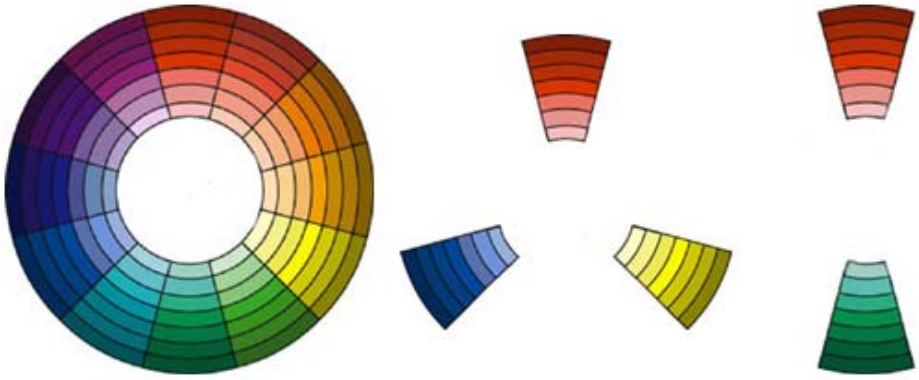

Fig. 3.4 – Färghjulet och några färgmodeller

Färghjulet representerar de färger som finns, den kan delas in i ännu mer detaljrikedom. Ett enkelt sätt att välja färger är att ta de mittemot varandra, de är varandras komplementfärger. Färgerna som återfinns i logotypen lämpar sig bäst med triadicfärger som ger en ytterliggare en färg att arbeta med. Istället för att använda sig av en helt vit bakgrund så matchar man in lite gult i färgen för att på så sätt knyta ihop modellen. Utöver dessa tre färger användes en monokrom skala från vitt till svart för text och konturer. [9]

För att lättare matcha in färger kan man använda andra nyanser av färgerna blå och röd som är mer matta. Tanken på vad färgerna representerar finns fortfarande kvar men är anpassade för färgschemat på en applikation.

3.5 Fördelning av utrymme

Horisontellt led

Att tänka på är att även om en användare har upplösningen 1024x768 garanterar det inte att webbläsaren har möjlighet att visa material på en yta som är 1024 pixlar då användaren kan anpassa webbläsarfönstret. Man kan dock i de flesta fall alltid maximera webbläsarfönstret i horisontellt led. Eftersom applikationen kommer att vara anpassad för upplösningen 1024x768 så ger det 1024 pixlar minus rullningslistans 18 pixlar i sidled att arbeta med. Fördelningen av utrymme i horisontellt skall vara fördelat i två delar, en meny och ett textfält där texter skall presenteras. Textfältet skall vara tillräckligt brett för att rymma bilder och ge en behaglig textlängd för läsbarhetens skull. Formatet skall även vara utskriftsvänligt på en A4. Menyn skall rymma de längsta orden, som 35 tecken, utan göra en

radbrytning. En pixelbredd på hela applikationen kommer därför att vara 900 pixlar med ett textfält på 650 pixlar och ett menyfält på 250 pixlar.

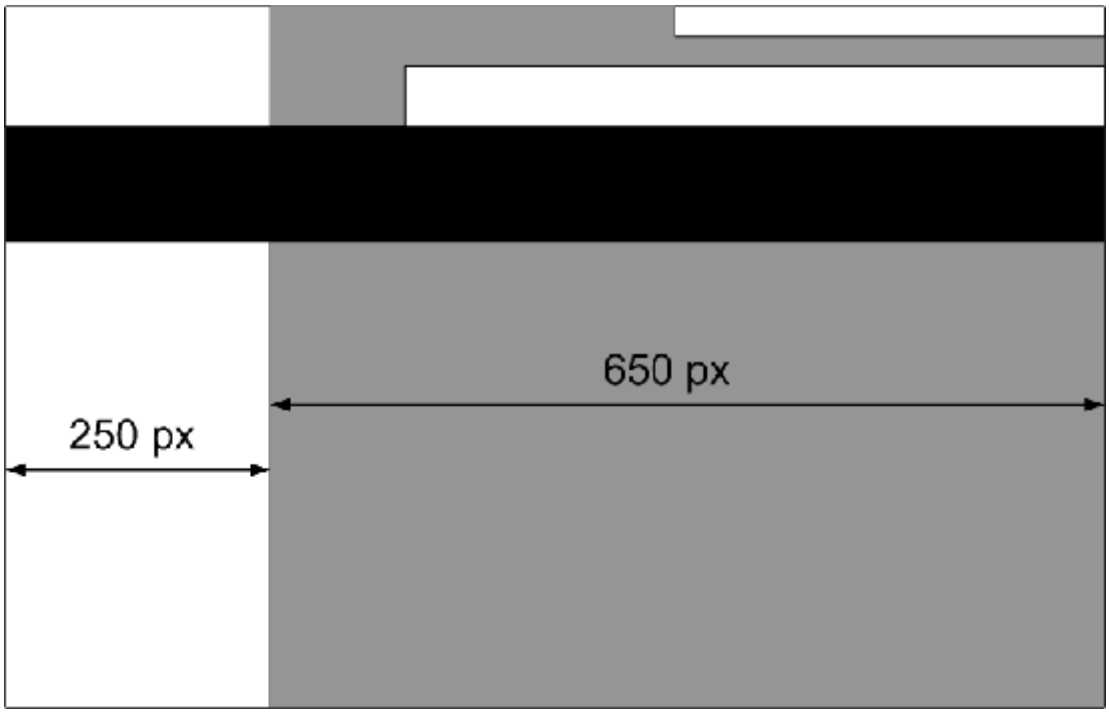

Fig. 3.5 – Utrymmesfördelning horisontalt

Vertikalt led

Utrymmet som är ledigt att visa material i vertikalt led är mycket svårare att uppskatta eftersom utrymmet är olika mellan de olika webbläsarna. De olika standardfälten som navigerings-, bokmärkes och verktygsfält är valbara och skiljer sig i design mellan. Det finns även en mängd olika andra fält som man kan lägga till som inte är standard, som exempelvis Googles sökfunktion, aktuellt väder osv. som tar utrymme.

I vertikalt led kommer det finns tre områden, ett fält med en företagslogga, huvudmeny samt en specialmeny, ett fält med en banner samt ett fält där undermenyn och textfältet är placerad i.

Applikationen kommer därför att ha en automatisk anpassning i vertikalt led med hjälp av en rullningslista. Förstasidan skall man dock inte behöva skrolla på om man använder sig av standard fält i Mozilla Firefox och Internet Explorer. Specialmenyfältet med en logga samt handboksbilden kommer att ta upp 150 pixlar vilket lämnar resterande till textfältet.

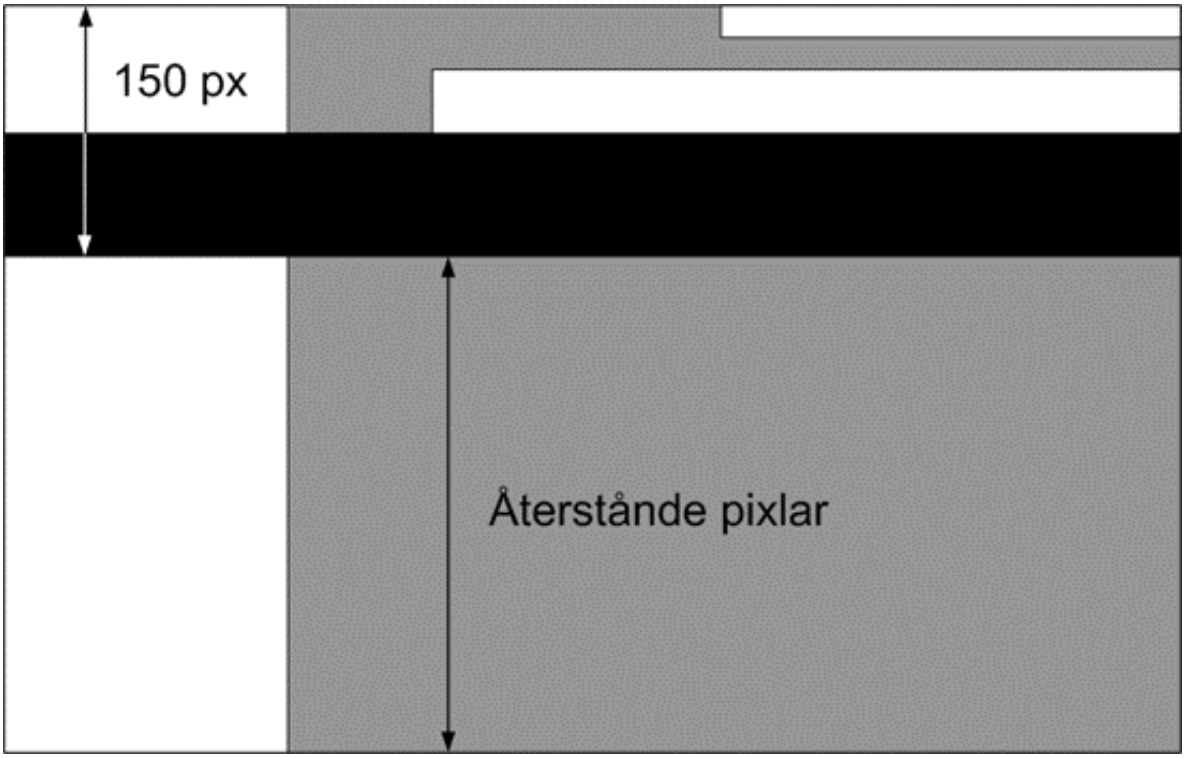

Fig. 3.6 – Utrymmesfördelning vertikalt

3.6 Slutdesign

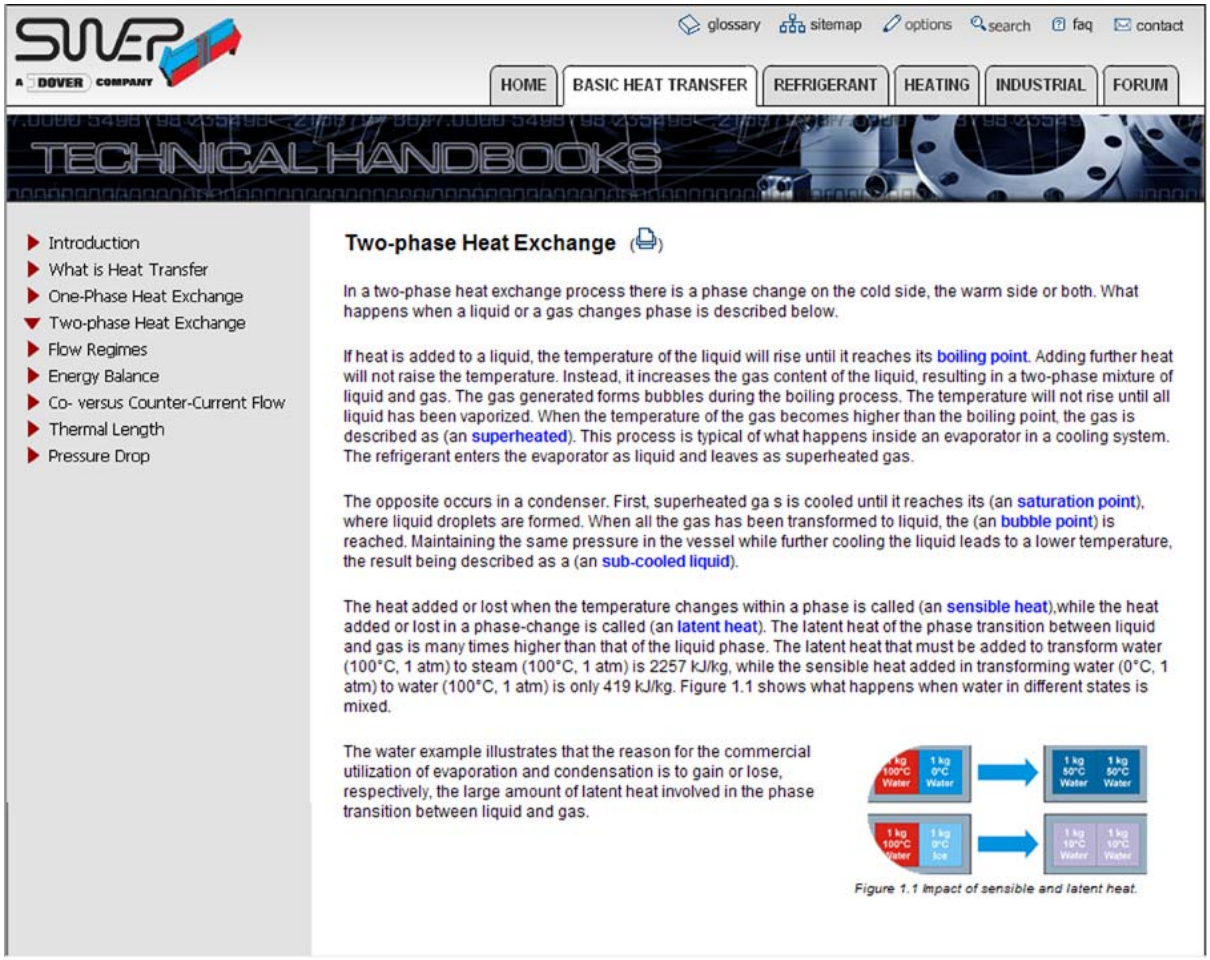

Fig. 3.7 – Slutdesign

Specialmenyn

Specialmenyn skall bestå av fem olika länkar med tillhörande ikoner. Denna meny skall alltid vara synlig och tillgänglig för användaren. Menyn kommer att vara uppdelad i två olika sorter; en som öppnar en pop-up, och inte påverkar vart man befinner sig och en som öppnar i det vanliga textfältet. Specialmenyn är uppbyggd av följande länkar, delarna beskrivs mer ingående i kapitel 5.4.

Fig. 3.8 – Specialmenyn

Search – Sökfunktion som söker igenom alla texter i applikationen och returnerar var de ligger och en länk till dem. Det finns vanligtvis två olika typer av användare, de som söker information via sökmotorn och de som letar efter information genom att klicka sig fram. En applikation bör aldrig begränsa sig till en av dessa grupper och därför finns sökfunktionen.

Options – Valmöjligheter för att kunna anpassa textstorlek och typsnitt. Trots att varje webbläsare, t.ex. Mozilla Firefox, Internet Explorer, har en inbyggd funktion att ändra teckenstorlek så saknas funktionen att ändra typsnitt. Denna funktion inkluderas i applikationen eftersom olika användare är olika vana att läsa text i olika typsnitt så skall detta ej begränsas.

Glossary – ordlista av alla ord som behöver förklaring.

Sitemap – översiktskarta av hela applikationen med länkar till tillhörande sidor. Användaren skall snabbt kunna bilda sig en överblick över applikationen utan att börja navigera sig fram i olika nivåer av sidan.

FAQ (frequently asked questions) – en sammanfattning av de vanliga frågorna och deras svar. Anledningen att applikationen har denna funktion är att de vanligaste frågorna inte skall behövas svaras på flera gånger.

Contact - Här kommer kontaktinformation till SWEP presenteras, såsom adress, telefonnummer och fax.

Dessa länkar grupperas högst upp i det högra hörnet med huvudmenyn så användaren inte skall behöva flacka med blicken när denna navigerar.

Huvudmenyn

Huvudmenyn är skapad som ett fliksystem, där användaren genom att klicka på något av länkelementen aktiverar den valda fliken. Anledningen applikationen använder sig av ett fliksystem för huvudmenyn beror på att det är ett praktiskt taget standardiserat menysystem som är mycket enkelt att förstå sig på, det har sitt ursprung från flikar i t.ex. pärmar och almanackor där det används flitigt. Menysystemet gör det också enkelt att skapa kontrast för att visa vilket menyval, flik, som är valt. Det är även möjligt att gruppera/gömma information som inte är väsentlig för användaren i det aktuella stadiet. Skulle inte ett fliksystem använts så skulle användaren få filtrera informationen i en större grad som gör att navigationen försämras och tiden som användaren måste lägga ner för att finna den rätta informationen skulle öka. Kanske skulle detta komma till den graden då användaren anser att det inte är möjligt att finna informationen som söks eller för ansträngande så användaren vänder sig till en ny webbplats/applikationen och dessa ses såklart som ett stort nederlag.

Värt att tänka på är att rubrikernas namn måste innehålla information som är relaterad till rubriken, görs inte detta kan det leta till att förvirring uppstår. Man bör således inte skiva om frystemperaturen för olika ämne i en flik som har rubriken "värme".

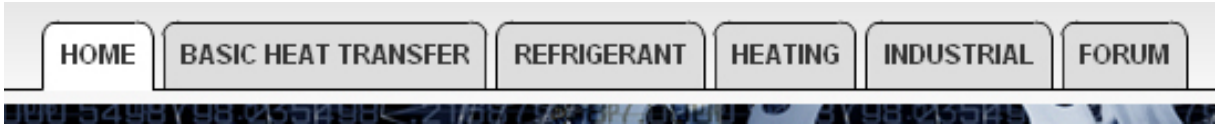

Fig. 3.9 - Huvudmeny

Sidomenyn

Sidomeny är skapad som en form av en nerfallande meny. Menyn är uppdelad i två olika nivåer. Först visas en lista på huvudmenyvalen, nivå ett, klickar man på dessa öppnas undermenyerna, nivå två, som man sedan kan klicka på. En nerfallande meny kan liknas med flikarna då denna meny också grupperar och gömmer information som inte är väsentlig för användaren i olika stadier.

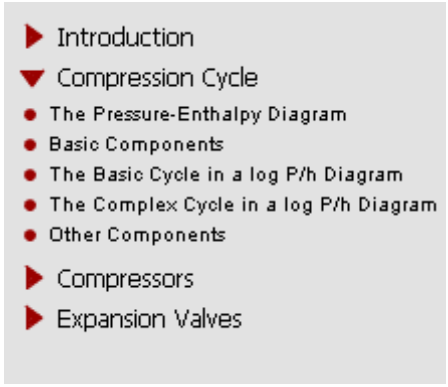

Fig. 3.10 - Undermeny

Anledningen att detta menysystem används är att man kan få plats med mycket information på liten yta eftersom alla menyval inte visas direkt. Man skall dock tänka på likt fliksystemet att använda sig av tydliga huvudmenyval eftersom användaren skall veta på ett ungefär vad denna finner under ett huvudmenyval. Skulle inte en nerfallande meny användas och alla information skulle placeras under vart annat skulle detta leta till att användaren måste i större utsträckning leta efter informationen. Konsekvenserna blir likt de konsekvenserna i att inte använda ett fliksystem/gruppera information.

Kombinationen av huvudmeny och sidomeny.

Kombinerar man fliksystemet som toppnivån och nerfallande menyn som undernivå placerad i fliken så får man ett navigationssystem som i första nivå

filtrerar bort det användaren inte är ute efter och som sedan ännu en gång filtrerar bort som inte användaren är ute efter. Vid användandet av denna kombination måste man vara ytterst försiktig så man inte filtrerar bort för mycket, d.v.s. använder för mycket flikar och för många nivåer av nerfallande menyer. Vid varje flik/nerfallande meny så måste användaren själv tolka vad som kan finnas under menyn och stämmer inte detta överens med applikationen uppstår navigationsproblem. Vid namngivningen av flikar och nerfallande menyer så skall man inte använda sig av namn som en användare kan förknippa med varandra, t.ex. kyla och fryspunkter. Görs detta så kan förvirring uppstå vad som kan finnas under varje meny.

Textfältet

Materialet som presenteras här har liknande struktur som de tidigare lösningarna och innehållet är ingenting vi har arbetat med, skillnaden är att vi fält in bilder som låg i marginalen och på så sätt presentera bilder och text i samma fält.

3.6.1 Designproblem

Under designarbetet skulle vi även utveckla utseende och position för myBOOK, detta visade sig vara svårare än vi från början tänkt. Formatet passar inte att parallellt ha den vanliga informationen bredvid den egna texten i myBOOK. Då skulle vi vara tvungna att införa en horisontal rullningslista vilket vi vill undvika. Beslutet togs att skjuta upp den delen och istället börja på programmeringen. Senare togs myBook bort helt vilket var följden utav problem i Flash som går att läsa om i 5.2.

Utöver detta bryter vi medvetet om en inarbetad funktion som många använder har vant sig vid, nämligen att man kan klicka på företagslogotypen i det övre vänstra hörnet för att komma till startsidan. Eftersom det är SWEPs logga känns det inte logiskt att koppla denna till startsidan för applikationen

4 Programspråk

Här följer en genomgång av möjliga programspråk som användes. SWEP har meddelat att de med 99% säkerhet kommer att använda sig av en Microsoft Server även i framtiden och på den använda sig av .NET, därför har PHP uteslutits. Önskemålet var att Flash skulle användas.

Undersökningen här görs för att se om Flash behöver kompletteras med något annat språk samt vad det finns för likheter och skillnader, bakgrund samt för- och nackdelar kommer att gås igenom. Programspråk som kommer att gås igenom är:

Adobe Flash (ActionScript 2 & ActionScript 3)
ASP.NET (C# & Visual Basic)
JavaScript

4.1 Adobe Flash

Flash kan ses som pionjärerna för dynamiska hemsidor. Det i en tid då det fanns animerade gif-bilder eller enkla JavaScript för att gör något som bröt mot den annars statiska HTML sidorna. Med Flash frigjordes möjligheten att enkelt och smidigt presentera animationer och dynamiska element för att liva upp statiska sidor.

Det var Macromedia som skapade och publicerade programmet och har sedan dess uppdaterats ett antal gånger. Sedan har Adobe köpt upp rättigheterna och släppt version CS3.

Programmet i sig är ett gränssnitt som bygger på en scen dit man kan dra eller scripta andra element (movieClips, Graphic eller Buttons) till. Dessa lägger sig i olika lager. Fördelen med detta är att man kan gå in och ändra eller göra individuella animationer för samtliga objekt.

I Flash finns sedan en tidslinje som spelar upp saker och ting i ordning. Detta gör att man utan programmeringskunskaper kan göra egna animationer. Genom att rita en bild i en bildruta och sedan rita en annan längre fram på tidsaxeln så kan Flash interpolera fram bilderna emellan och på så sätt göra en animation. Vill man göra mer avancerade applikationer kan man programmera för hand istället.

Flash använder sig av ett objektorienterat programspråk som heter ActionScript. Liksom Flash så kommer ActionScript i olika versioner. I den senaste versionen som finns att tillgå idag är ActionScript 3, med denna kan man kommunicera med en databas utan att ta hjälp av en serversite som exempelvis PHP eller ASP. Tyvärr har ActionScript 3 inte har samma syntax som de äldre versionerna. [10] [11]

Fördelar med Flash

- Flash använder sig av vectorgrafik, vilket innebär att upplösning och dylikt inte spelar någon roll så länge man inte använder sig av externa bilder
- Är utvecklat för webben och kan således kommunicera med andra program utan extra tilläggsprogram
- Är delvis ett WYSIWYG* program som underlättar vid grafisk design

Nackdelar med Flash [11]

- Flash kräver att webbläsaren man använder har en plug-in för att kunna köras. Dock så är Flash väldigt utbrett och vanlig att de flesta webbläsare redan har denna.

4.2 ASP.NET

.NET är i grund och botten en plattform för att skapa, sprida och underhålla webbapplikationer och tjänster. Det är ett Microsoft utvecklat verktyg som man kan kombinera med ett antal andra olika språk för att på så sätt skapa kraftfulla och dynamiska hemsidor.

Microsofts serverorienterade programmeringsspråk ASP (Active Server Page) anses vara bra och kraftfullt men kräver väldigt mycket programmering. Genom att skapa .NET har Microsoft försökt göra det enklare för användaren att skapa applikationer och liknande.

ASP.NET är en kombination som är Microsofts uppföljare till Active Server Page. ASP är ett fristående språk och den senaste versionen av det är 3.0 sedan har det gått över till en kombination med .NET. ASP och ASP.NET är alltså inte samma sak. Även om de har saker gemensamt så går det inte att använda ren ASP på en ASP.NET server utan tillägget ASPX.

Genom att göra det lättare för användaren att programmera så minskar mängden kod och hela arbetet blir mer effektivt och produktivt. Genom att skilja design och script så kan exempelvis ett utvecklar team jobba bättre parallellt. Med .NET får man tillgång till ett stort bibliotek med hjälpklasser och annat som underlättar ytterliggare vid programmering. Dessutom är det möjligt att uppdatera filer samtidigt som servern är igång. [12] [13] [14]

Fördelar med ASP.NET [15]

- Stödjer många andra programspråk däribland, C, C#, C++, JavaScript och Visual Basics.
- De flesta HTML objekt går att styra via scipt som om de vore ASP.NET objekt
- ASP.NET objekt använder sig av XML som underlättar vid ändringar och informationsupphämtning.

Nackdelar med ASP.NET

- Inte helt bakåtkompatibelt med vanlig ASP.
- För att köra ASP.NET på Linux krävs det ett tillägg (MONO) fördelen med det är att det är gratis.

4.3 JavaScript

JavaScript är ett välkänt språk för att programmera funktioner och annat i en webbläsare. Script är en förenklad version av programmering, koden tolkas direkt av webbläsaren och behöver därför inte exekveras. Det kan ses som ett komplement till HTML för att göra det mer dynamiskt. I och med detta ligger scriptet på klientsidan och inte på servern, även om försök har gjorts för att anpassa det till server-sideprogrammering men här är för stor konkurrens av PHP, ASP och Perl.

JavaScript introducerades 1995 i Netscapes Navigator version 2 och Microsoft tog efter och införde en egen implementation Jscript i version 3 av Explorer.

En webbläsare som arbetar med ett JavaScript modifierad HTML-dokument lyssnar efter de händelser som användaren kan tänkas göra (musklickningar, knapptryckningar osv.). JavaScriptet utför det man scriptat det att göra om villkoren uppfylls. [16] [17] [18]

Fördelar med JavaScript

- JavaScript är väldigt utbrett och vanligt.
- JavaScript kan hantera cookies.

Nackdelar med JavaScript

- Vissa webbläsare stödjer inte JavaScript och användaren kan välja att stänga av funktionen på grund av säkerhets eller annat.

4.4 Sammanfattning av Programspråk

Eftersom SWEP ville att vi skulle utveckla applikationen i Flash så kommer det att användas. Vi skulle dra nytta av att innehållet för applikationen redan var formaterat med HTML och hämta ut detta från en databas. Utifrån detta har vi valt ut följande språk:

- Adobe Flash (ActionScript 2)
- JavaScript
- ASP.NET (C#)

För att skapa grafik och design har Flash störst fördel eftersom det är ett animationsprogram i grund och botten. Flash har även stöd för HTML-formaterad text. Med ActionScript 3 görs det möjligt att kommunicera med databas utan mellanhand. Vi hade tyvärr ingen erfarenhet av ActionScript 3 och tog beslutet att använda oss av ActionScript 2 istället. Som mellanhand för uthämtning från databas valde vi att använda oss av ASP.NET. Vi kommer att använda oss av språket C# när vi programmerar i ASP.NET eftersom dess syntax och struktur mer liknar andra språk vi arbetat med tidigare än jämförelsevis Visual Basics. För att använda oss av funktioner inifrån innehållet kommer vi att utnyttja JavaScript.

5 Utförande

Efter våra undersökningar och förberedelser hade vi kommit fram till att Adobe Flash med ActionScript 2 skulle användas för att göra gränssnittet. Det är i förstahand navigationen och presentation av text som är det viktigaste så detta bör fungera innan man implementerar specialfunktioner så som bildvisningsfunktioner och sökfunktioner.

5.1 Arbete i Flash

Arbetsupplägget vi använt oss av här är att börja göra designen utseendemässigt lika och sedan göra den mer och mer dynamisk, exempelvis att det skapas knappar efter det antal som finns i databasen och likadant med undermenyer. Man överför designen från Photoshop till Flash steg för steg. När utseendet i Flash speglar den modell som arbetats fram i Photoshop tar man itu med det viktigaste för navigationen, nämligen huvudmenyn.

5.1.1 Huvudmenyn

När det handlar om saker som upprepar sig som exempelvis en meny brukar vi arbeta efter en metod som går ut på att man först får en bit att fungera, sedan två och sist tre. Har man fått tre stycken att fungera dynamiskt så är skillnaden att göra 8 eller 3 ingen större process, då har man arbetat fram sin algoritm som gör att det fungera dynamiskt på flera utan några större ändringar.

Knappen som helhet, består av en knapp samt ett dynamiskt textfält.

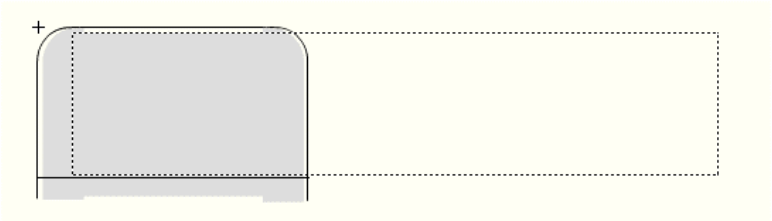

Fig. 5.1 – Sammansatt menyknapp

Det första som gjordes var att göra designen på menyknapparna uppdelad så att det går att programmera varje del individuellt. Knappen är nu uppdelad i tre movieClips där varje movieClip består av en ram och en bakgrund.

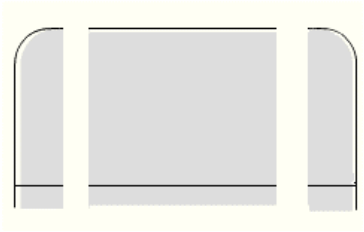

Fig. 5.2 – Uppdelad menyknapp

Bland annat kommer man åt bakgrunderna så att man kan ändra dessa vid önskade tillfällen. Man hade kunnat göra hela knappen som ett objekt men då kommer man inte få samma avrundning på samtliga knappar om de varierar i storlek.

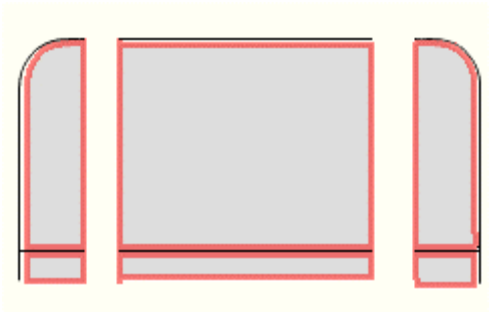

Fig. 5.3 – Bakgrundsytor för menyknapp

Istället sträcker man ut den mittersta biten som endast har räta vinklar i sig. Den biten anpassas genom att läsa av längden på texten som skall stå där, sedan placeras den sista biten intill mittbitens högerkant och knyter på så sätt samman knappen till en enhet.

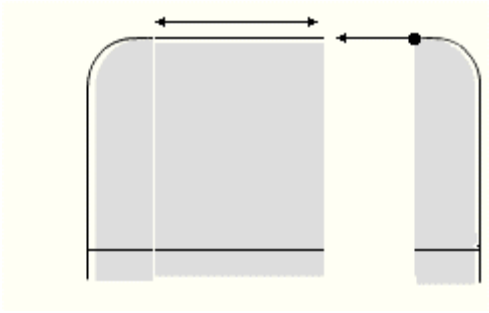

Fig. 5.4 – Sammansättningsprocessen för menyknapp

Placering av knapparna blir lite speciell eftersom de byggs ifrån höger, med andra ord att den sista knappen alltid skall ligga längst till höger, ungefär som en högerställd text i en ordbehandlare. Därför måste man skala om knappen och sedan mäta ut dess längd för att kunna subtrahera det från den positionen i x-led där tidigare knapp är placerad.

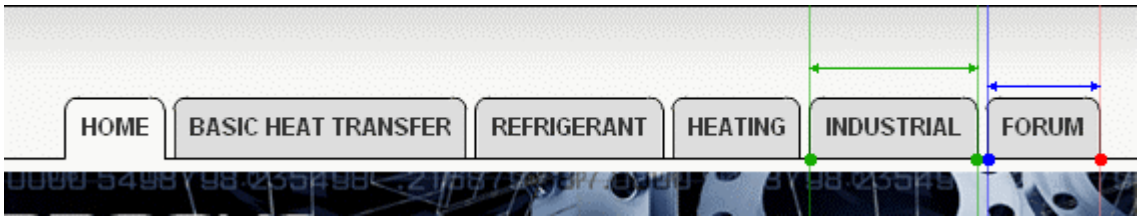

Fig. 5.5 – Placering av samtliga menyknappar

Den röda punkten i figur 5.5 representerar startpunkten. Det är en global variabel som kan ändras efter behov. I applikationen ligger den på 880. För att då veta exakt var sista knappen skall börja måste man räkna ut dess bredd vilket den blå pilen representerar. Denna bredd subtraheras från den röda punktens position vilket ger den blå punkten.

Samtliga delar hänvisar till figur 5.5:

Röd punkt – blå pil = blå punkt
880 – 57 = 823

Den blå punkten sparas som den nya startpunkten. Nästa knapp görs på nästan samma sätt. Bredden på den (grön pil) räknas ut, men eftersom det skall finnas ett avstånd mellan knapparna måste det läggas på en buffert. Avståendet för det är en global variabel som adderas på bredden innan den subtraheras från tidigare start punkt.

Generell formel från de efterföljande knapparna blir då:

Tidigare startpunkt - (Knappens bredd + avståndet mellan knappar)= Ny startpunkt

Det man bör tänka på här är ordning man läser av värdena som skall stå på knapparna. Enligt vår lösning lagras de i en array, en sorts lista. Fördelen är att man kan med en loop gå igenom varje listpunkt antingen bakifrån eller framifrån. Då kan man själv välja om man skall börja med det som skall stå sist och öka eller börja sist och gå baklänges. Det beror lite på hur man hämtar ut sina värden från databasen.

Alternativ 1:

_global.mainmenucontent = new Array("FORUM", "INDUSTRIAL", "HEATING", "REFRIGERANT", "BASIC HEAT TRANSFER", "HOME");

Sedan kan man gå igenom den med en vanlig for-sats som börjar på 0 och ökar (*i*++).

Alternativ 2:

_global.mainmenucontent = new Array("HOME", "BASIC HEAT TRANSFER", "REFRIGERANT", "HEATING", "INDUSTRIAL", "FORUM");

Då kan man köra igenom en for-sats som börjar på arrayens längd (*_global.mainmenucontent.length*) och minskar (*i*--).

5.1.2 Flikar med minne

En av fördelarna med Flash är att det går att göra ett enkelt system för att kunna alternera mellan många olika flikar och ändå få dem att behålla sin status när man tittar på någon annan. Det är en fördel då man snabbt och enkelt kan jämföra olika böcker som har liknande lösningar.

När användaren trycker på en knapp i huvudmenyn för första gången skapas en bakgrund (ett movieClip) som täcker hela fältet under bannern och huvudmenyn. Det är även i detta movieClip som eventuella undermenyer hamnar samt allt innehåll från databasen laddas in hit, och det är endast en bakgrund som visas åt gången. Det åstadkoms genom att ändra visability (synlighet).

5.1.3 Undermenyer & Innehåll

Eftersom (nästan) varje flik på sidan motsvarar en teknisk handbok så kommer deras avsnitt innehålla en hel del undermenyer i varierande mängd. För att på ett dynamiskt sätt kunna hämta ut menyraderna måste det ställas en fråga till databasen. Svaret man får tillbaka är en lång rad med en väldig massa extra tecken. Man måste då kunna sortera ut vad som är namnet på undermenyn och vad som är skräp. Man vill exempelvis inte ha med en massa extra tecken, utan varje menyrad bör ligga, som det skall stå, i en array för enkel åtkomst.

Undermenyer

Ponera att användaren klickar på fliken med namnet ”REFRIGERANT”, från Flash skickas en variabel (med functionen SendAndLoad) till en adress (…/sidemenu.aspx?submenu=REFRIGERANT), ASP.NET-dokumentet tar emot variabeln ”submenu” och ställer därefter en fråga till databasen där rubriken på varje kapitel hämtas ut. Mellan varje namn läggs en avskiljare för att i Flashfilen dela upp dem som separata element.

Tillbaka i Flashfilen tas en lång URL-tolkad sträng emot som delas upp och ersätts med vanliga tecken. Det görs med split- och splicefunktioner som letar upp separatorn som känns igen som ”%3C%25Zseparator%3E”, i ASP.NET-dokumentet skrivs detta som ”<%Zseparator>”, liknande görs med mellanslag, bindestreck osv. Efter denna process har man en array med rubriker istället för en lång sträng med udda tolkade tecken.

Nästa steg blir att presentera undermenyerna och ange dem som länkar. Arrayen gås igenom och varje rubrik placeras som ett eget objekt i Flashfilen, ett stycke kod appliceras som säger att om användaren skulle trycka på den så skall undersöka ifall den har en undermeny under sig i hierarkien samt ladda in innehållet i rätt movieClip.

Undermeny nivå 2

Vid klickning på ett menyobjekt åkallas en funktion som återställer undermenyn till sitt ursprungliga utseende följt av att expandera ut den igen om det finns en andra nivå. Informationen hämtas in på liknande sätt som på den första nivån. Flash beräknar antalet menyposter och listar upp dessa under dess förälder samtidigt som den flyttar ner övriga.

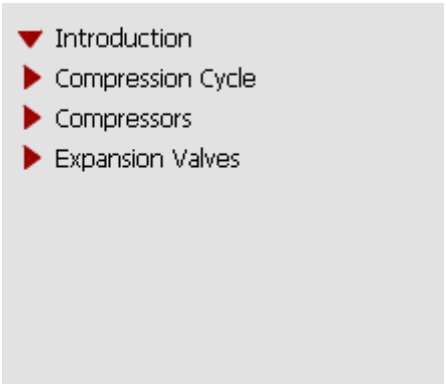

Fig. 5.6 - Undermeny

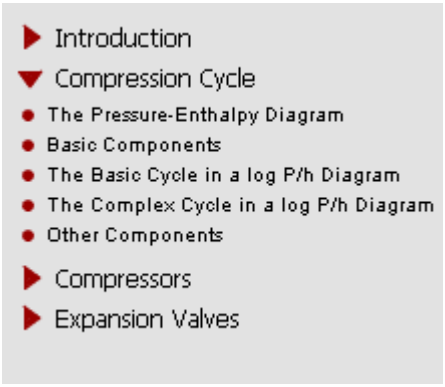

Fig. 5.7 – Undermeny nivå två

Begränsningen här är att man bara kan ha en andranivåmeny synlig åt gången. Hämtning av innehåll fungerar likadant för alla menyposter.

Innehållet

En ny sendAndLoad-funktion utförs där det som skickas är: rubriken och vilken bok den tillhör. ASP.NET-dokumentet hämtar ut innehållet som stämmer överens med rubriken och boken som skickades, sedan får Flashfilen tillbaka en HTML-formaterad text som presenteras i ett textfält.

Här uppstår stora problem. Skillnaden mellan hur en webbläsare och Flash tolkar innehållet är uppenbart inte likadant. Efter att ha experimenterat med HTML-koden så fick vi fram ett utseendemässigt bra resultat, men fortfarande med sina brister, dels är den inte stabil, vilket blev väldigt tydligt då man förde över muspekaren på en länk och all text som befann sig under denna hamnade som en smal kolumn, och dels kan går det inte att integrera några funktioner som exempelvis JavaScript.

Med detta som resultat kan vi svara på frågan om Flash var en lämplig plattform för utveckling av stora mängder HTML-formaterat material, svaret blev nej. Men för att ändå kunna presentera något för SWEP så bestämde vi oss för att utveckla en alternativ lösning.

5.2 Alternativ lösning

Flash stödjer HTML, men inte all HTML, långt ifrån. Istället för att ge ett felmeddelande så struntar Flash i att försöka tolka det den inte förstår. Bland annat så blir en vanlig p-tagg (<p>) ett inline-objekt vilket innebär att den inte ger radbrytning innan och efter sig själv. En numrerad lista blir en vanlig punktlista och rubriker (<h3>, <h4> osv.) formateras inte alls. Detta problem gör bland annat att man inte kan använda sig av JavaScript i HTML-formaterade textfält, även vanliga tabeller står i samma kategori. Med andra ord så fungerar Flash inte så bra om man vill presentera ett dynamiskt HTML-formaterat innehåll genererat från databas.

Fördelen här att kunna utnyttja Flash för att behålla de olika flikarnas tillstånd intakt när man växlar mellan dem blir en oväsentlig funktion om materialet inte är presenterat på ett strukturerat sätt. Om det inte går att göra en bra layout för text och bilder i Flash måste det finnas en annan lösning. Problemet blir att på något sätt få det HTML-formaterade innehållet presenterat på en HTML-sida med kommandon skickade från Flash, går det att behålla tillstånden från varje flik är det en bonus.

Först och främst för att presentera en Flashfil på Internet så måste man baka in den på en HTML-sida. Genom att använda en iFrame som har ett djup över Flash filmen (högre z-index) kan man presentera material så att det ser ut som en och samma sak. Detta eftersom där innehållet presenteras ligger ovanpå Flashfilen, dvs. ”närmare” betraktaren. Från Flash kan man kommunicera med sin förälder, HTML dokumentet som den ligger i, genom kommandoraden getURL. Här kan man skicka information om vilken sida som skall visas till en iFrame med ett bestämt id.

Genom att skicka iväg adressen till en ASP.NET-sida med önskade variabler så kan man presentera HTML-formaterad text från databas till den iFrame med korresponderande id.

Processen är lättast att följa genom ett exempel från Flash till HTML med hjälp av JavaScript.

Flash:

getURL(4, "getContent.aspx?Xname=Basic Components&book=Refrigerant&css=12a.css");

Det innebär att den iFrame med id 4 skall ladda in sidan med adressen ovan. ASP.NET-sidan i sin tur hämtar ut innehållet från databasen där namnet är Basic

Components från boken Refrigerant och presenterar det hela med stilmallen 12a.css (12 punkter Arial).

HTML:

Den iFrame som är aktuell uppfattar kommandot och innehållet laddas in. En del problem kvarstår dock fortfarande. Höjd på en iFrame uppdateras inte efter innehåll utan behåller den höjden man skrivit in i HTML-koden. Det i sin tur leder till att man kommer att få en extra rullningslista på sin iFrame, och rent funktionellt så är två rullningslistor (en på sin iFrame och en i webbläsaren) inget att föredra, sedan ser det inte tilltalande ut heller.

För att lösa detta kan man använda sig av JavaScript, på Internet kan man hitta olika lösningsförslag som fungerar antingen till den ena eller andra webbläsaren eller så kan man hitta script som fungerar för både Mozilla Firefox och Internet Explorer som enligt W3Schools är de två vanligaste webbläsarna. Av de lösningsförslag som undersöktes ansåg vi att lösningen skapad av Dynamic Drive var den som fungerade bäst i både Mozilla Firefox och Internet Explorer. Andra lösningar har diskuterats fram på forum-sidor en lösning skapades av användaren ”MajorFracas” på Ozone Asylum. Av dessa två var Dynamic Drive bättre eftersom det var en färdig lösning och fungerade direkt. Den var dessutom lättare att förstå sig på för att den var bättre strukturerad. [19] [20] [21]

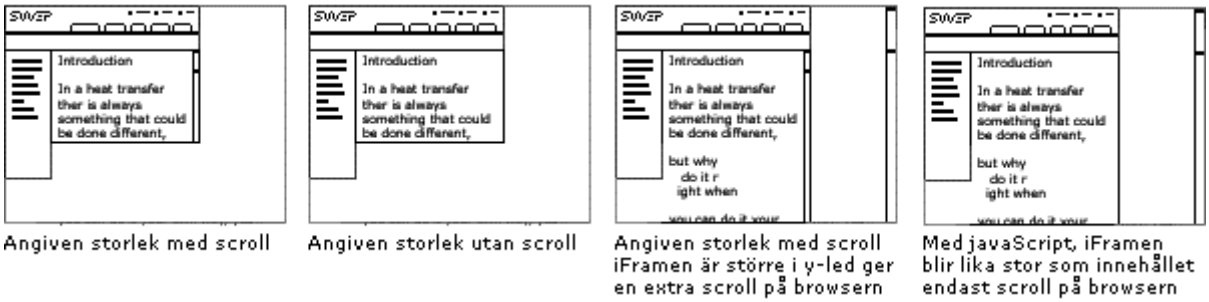

Fig. 5.8 – Dubbla rullningslistor

Problemet med tillståndsminnet kvarstår dock, lösningen vi har använt oss av innefattar att använda sig av en iFrame för varje flik och JavaScript för att skilja på vilken som skall visas.

Genom att placera flera stycken iFrames på samma position fast med olika z-index blir det som att de staplas ovanpå varandra. Varje iFrame får sitt egna unik id som lätt går att sikta på från Flash när det ska skicka adresser till den. På samma sätt som man gjorde på en iFrame så använder man sig av getURL i Flash. Skillnaden här blir att bara få den iFrame man skickat till att visas.

Det är här JavaScriptet kommer in i bilden, i det yttersta HTML-dokumentet finns en funktion som vi har skrivit som tar emot ett nummer (samma nummer

som id på en iFrame). En loop går igenom samtliga nummer som det finns iFrames, en if-sats undersöker om varje nummer stämmer överens med det mottagna numret. Om det är samma så skall den iFrame som matchar visas (display: block) och om inte så skall den inte synas (display: none). Fördelen här är dessutom att webbläsarens rullningslista enbart anpassas åt det som syns. Funktionen för detta åkallas från Flash strax innan den skickar adressen till en iFrame, även här använder man sig av getURL.

Med getURL kan man inte bara skicka adresser till iFrames utan åkalla JavaScript detta görs med följande kommando:

getURL('javascript:fromFlash(4)');

iFrames staplas ovanpå varandra, varje iFrame får ett unikt id som används både för att skicka adresser och ändra synlighet på. Som index är det bara startsidan som syns resten har en display: none.

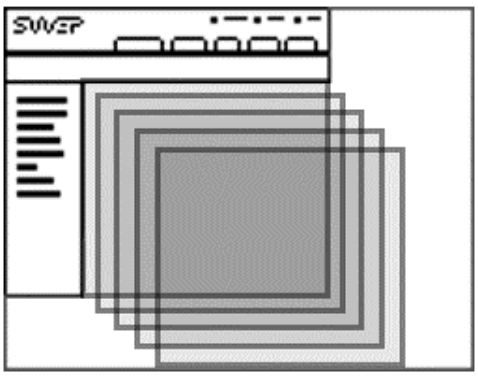

Fig. 5.9 – Staplade iFrames

När ett anrop till fromFlash() görs går det igenom en loop med alla iFrames för att jämföra dessa med det mottagna numret. Om de överensstämmer så skall den iFrame bli synlig (display: block) annars ska den bli osynlig (display: none).

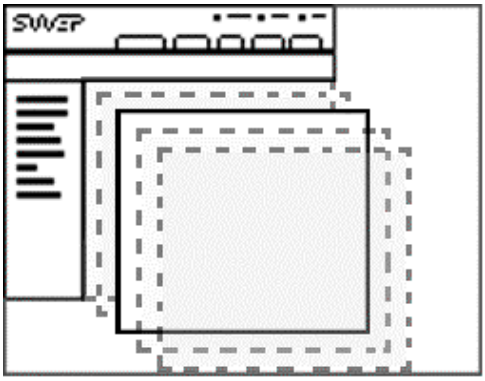

Fig. 5.10 – Dölj/visa iFrame

Detta har gett oss en funktionell sida med Flashmenyer och HTML-formaterat innehåll som presenteras i olika iFrames för att behålla tillståndet, det går dessutom att använda sig av JavaScript obehindrat. En nackdel med att använda sig av iFrames är att "tillbaka"-knappen inte fungerar, det hade den förvisso inte gjort heller om man gjort hela sidan i Flash. Det beror på att det är endast HTML-dokumentet som innehåller alla iFrames och Flash applikationen som går tillbaka, vilket är ganska logiskt. Tillbakaknappen kan inte veta vilken iFrame den ska flytta tillbaka.

5.3 Arbete i JavaScript

Nu när innehållet är skilt från Flash har möjligheterna att använda sig av JavaScript frigjorts. Det är ett vanligt sätt att göra sidor mer dynamiska än man kan med vanlig HTML. Vi kommer främst att använda os av JavaScript för bildvisning och ordförklarare, men även aktiva som användaren inte kommer att märka av som de vi gått igenom tidigare med storleksändring och samspelet mellan Flash och HTML.

5.3.1 Bildvisning

”En bild säger mer än tusen ord…” – det är ett gammalt talesätt som Georg Stenberg från Beteendevetenskapliga Institutionen vid högskolan i Kristianstad gjort ett projekt om. Undersökningen visade att bildminnet är mer lättåtkomligt än ordminnet, men även att man kan lagra falska minnen utifrån de associationer man gör till bilden eller orden. Därför kan en kompletterande text vara ett bra komplement, så att man associerar bilden till de orden och inte låter hjärnan bearbeta bilden själv. [22]

I de ursprungliga handböckerna presenteras bilderna som tumnaglar (miniatyrbilder, från engelskan *Thumbnails*) lite sidan om texten så att de lätt går att hitta via referenser från texten. Detta är ett klassikt sätt att använda sig av illustrationer i böcker, men när man presenterar det på elektroniskt vis är det slöseri med utrymme. Det stödjer dessutom inte något utskriftsformat om man vill ha både text och bild på ett och samma papper.

Därför togs beslutet att införa bilder i texten, både med tanke på att få större brytningar mellan olika stycken men även för att det blir lättare att skriva ut. En text som innehåller mer bilder och illustrationer upplevs som mer lättläst än en med enbart text. [23]

Utöver placeringen på de tidigare böckerna så är funktionen med tumnaglar ett ganska smart och vanligt sätt att presentera bilder som annars kan vara för stora och otympliga för att på ett smidigt sätt integrera den med texten. För att se den stora bilden så måste användaren trycka/föra över muspekaren eller liknande. De tidigare böckerna frammanar en ruta som går att flytta och stänga ovanpå den befintliga texten. Det anser vi vara bättre lösning än att komma till en ny sida för att sedan tvingas gå tillbaka för att fortsätta i texten.

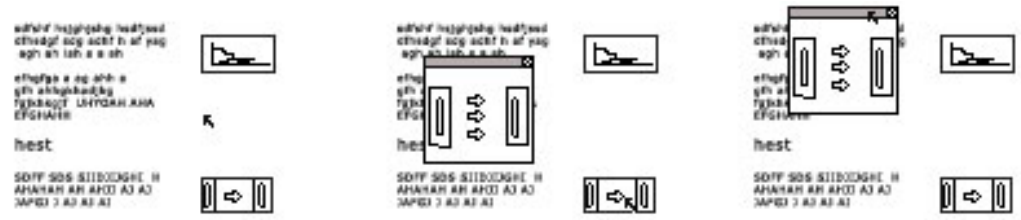

Fig. 5.11 - Bildvisning

SWEPs tidigare bildlösning.

1. Tumnaglarna är placerade vid sidan om texten.
2. Vid knapptryckning uppenbarar sig hela bilden
3. Bilden går att flytta på ifall man vill kunna läsa texten under

5.3.2 Lightbox

För att åstadkomma en liknande lösningen som SWEPs tidigare så har vi har använt oss av ett JavaScript som kallas Lightbox.

Lightbox är ett JavaScript som visar bilder som en ny ruta i webbläsaren. JavaScriptet skalar om rutan i realtid (om så önskas) så att den passar just den bild som skall visas. Användaren kan gruppera bilder och då skapas ett bildspel där användaren kan navigera mellan dessa med knappar i boxen. I rutan presenteras en ”caption” (bildtext) som förklarar vad bilden föreställer.

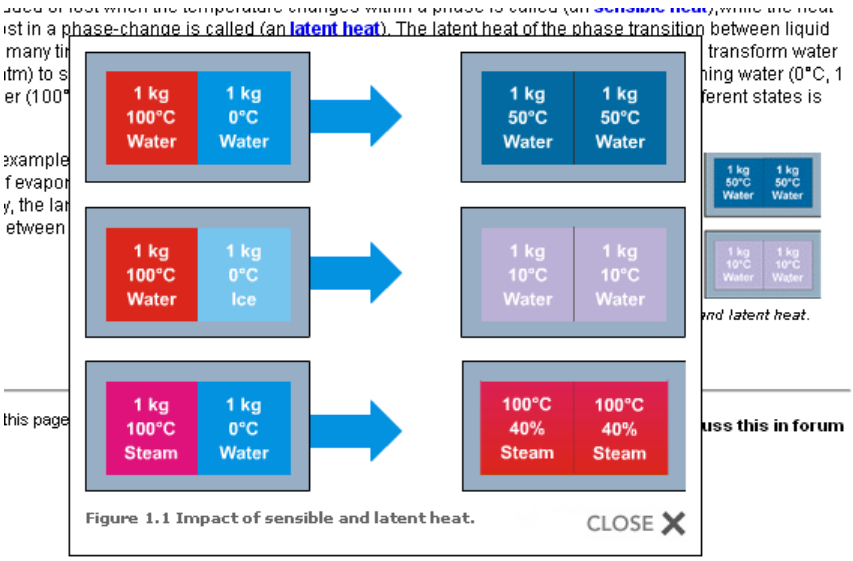

Fig 5.12 – Lightbox

Fördelarna med Lightbox är många, först och främst är det gratis och får användas av alla som vill det, det andra är att det fungerar i många olika

webbläsare, bland annat Mozilla Firefox och Internet Explorer. Med medföljande stilmall och bra kommentering i JavaScript-filen är det relativt lätt att alternera utseendet på själva Lightboxen. Den sista och avgörande fördelen är att man kan presentera mer än bara bilder, exempelvis Flashfilmer och även iFrames.

Det finns många olika versioner av Lightbox, skillnaden är inte så stor utan det handlar om buggfixning, webbläsarkompabilitet och vilka typer av filer som stöds. Vi har valt att använda Lightbox++ av två anledningar: dels dess Flashkompabilitet den andra av positionerings skäl. [24]

5.3.2 Positionering av Lightbox

Det största problemet med Lightbox var positioneringen. I scriptet så placeras det i mitten av webbläsaren. Eftersom Lightbox presenteras inuti en iFrame så uppstår ett problem. Eftersom det redan används ett annat JavaScript för att skala om den iFrame som syns så att den anpassas efter innehållet kommer Lighbox alltid placeras i mitten av den. Det är ett stort problem om det är en lång text, trycker man på en bild i början eller slutet av texten måste man skrolla för att kunna se den, då är lösningen inte speciellt användbar.

För att kunna avgöra var Lightboxen skall placeras använder man muspekarens position i y-led. Med ett aktivt JavaScript som hela tiden läser av musens y-position kan man lagra resultatet i en osynlig div med ett bestämt id. I JavaScriptet för Lightbox byter man ut den befintliga positionering i y-led mot att hämta ut y-koordinaten med document.getElementById().innerhtml från den osynliga diven. Nu blir Lighboxen placerad där muspekaren befinner sig.

Tänk på att bilderna i Lightboxen inte får vara bredare än den iFrame som presenterar den då dessa kommer att glida utanför ramarna och hela bilden kommer inte att vara synlig.

5.3.2 Ordförklararen - Overlib

I faktatexter uppstår det väldigt ofta facktermer som är nödvändiga för att kunna beskriva innehållet i texten på ett önskvärt sätt. Det kan vara svårt för en person som kanske inte använder sig av dessa termer i vardagen att förstå dess innebörd och således behövs det ett förklarande komplement någonstans. Dessa brukar vara samlade på ett och samma ställe och så är även fallet i SWEPs tidigare applikation, de ligger under namnet *glossary*.

Orden som behöver en förklaring länkar i sin tur till glossary-sidan. Nackdelen här är att användaren tvingas bort från sidan denna var på. En lösning på problemet är att ha som i böcker och lägga länkarna som en fotnot längst ner på

sidan, men det hade fortfarande tvingat användaren bort från det ställe i texten där denne befann sig.

En annan lösning som vi anser vara mindre krävande av användaren är att förklaringen kommer upp som en ruta vid muspekaröverföring. Med ett gratis JavaScript som kallas Overlib kan man åstadkomma detta. [25]

Overlib blir som en ”pop-up”-ruta som kommer fram vid aktivitet och där kan man presentera sin förklaring. Overlib begränsar sig inte enbart till text utan kan användas på en mängd olika sätt, antingen att på ett statiskt sätt där man i direkt i HTML-koden skriver in vad som skall visas, eller få den att visa bilder eller även iFrames. Genom att använda iFrames öppnas möjligheten att ladda in data från en databas. Varje ord får ett liknande script där endast namnet på ordet skiljer sig, den skapar en iFrame med en aspx-sida som tar emot namnet på ordet och hämtar ut dess förklaring från en databas.

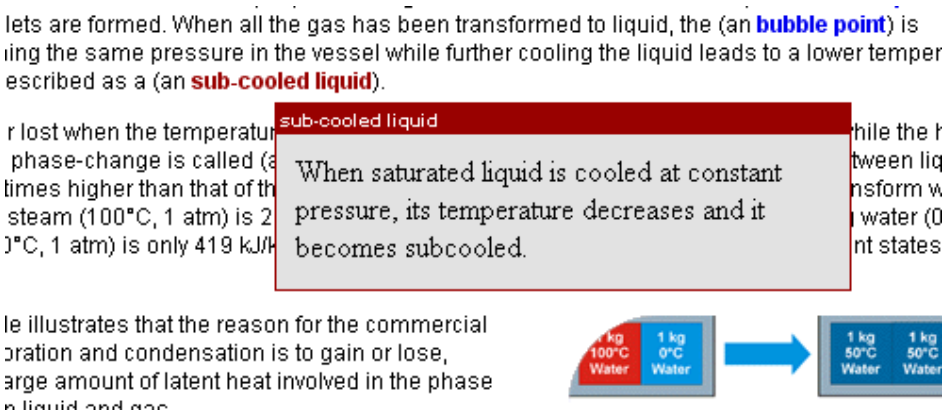

Fig 5.13 - Overlib

Med den lösningen får användaren förklaringen till ordet direkt bara denne för muspekaren över det och slipper således skickas vidare till en annan sida. Dock kan det vara bra att behålla en sida med en ordlista som samlar samtliga ord på ett och samma ställe. Skulle därför användaren välja att klicka på ett ord i fråga så kommer denne att flyttas till ordlistan och skrollas ner till det aktuella ordet.

Snabbaste sättet att ta sig tillbaka är att använda sig av en tillbakaknapp. Tyvärr är det så att när man använder sig av iFrames eller Flash fungerar inte den inbyggda i webbläsaren. Det beror på att den vill flytta tillbaka HTML-dokumentet som innehåller Flash och alla iFrames och inte innehållet i en iFrame, samma sak med Flash. Därför kan det vara bra att tillverka en egen tillbakaknapp, om inget annat så för att underlätta för användaren. När man trycker på ett ord i texten skickas ett värde till knappen så den vet var man skall hamna när man går tillbaka, knappen placeras dessutom invid det ord man tryckt på i ordlistan.

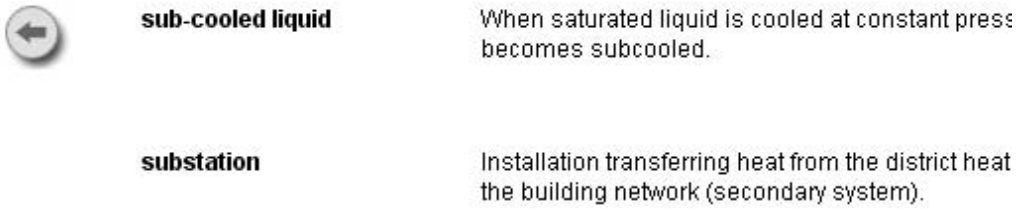

Fig 5.14 – Glossary, med tillbaka-knapp

5.4 Att integrera användaren

5.4.1 Feedback och Forum

När det här examensarbetet startades fanns det önskemål från SWEP om att göra det möjligt för användaren att kunna interagera i så stor utsträckning som möjligt. Mer specifikt än så var det inte, vi fick fria händer. Det första vi kom att tänka på var ett forum, där användarna kan diskutera handböckerna i mer eller mindre fria ämne. Ett drömscenario för SWEP hade varit att få en så pass seriös och insatt besökarbas som gärna besöker sidan ofta och skriver i forumet.

Forum

Att skriva ett forum från grunden är enbart det ett stort projekt som vi inte ansåg oss ha tid till. Därför valdes ett gratis forum som finns på Internet. [26] Forumet har de vanliga funktionera att kunna registrera sig och skapa nya trådar och dylikt. Det för att kunna strukturera upp de olika ämnena och så att användarna ska kunna ha bra kontroll över de dem själv har diskuterat.

Forumet integrerades på två olika sätt. Först och främst så har forumet placerats som en del i huvudmenyn, användaren som trycker på den fliken kommer till forumets startsida. Det andra är att integrera den på varje sida som innehåller text. Här har det underlättats för användaren genom att undersöka om någon redan har skrivit om ämnet ifråga, då kommer man till forumet med matchande rubriker, i annat fall kommer man till inloggning med följande skriv inlägg.

Forumet kommer att öppnas i ett nytt fönster eftersom om man ska diskutera om ett stycke i texten så vill man kunna hänvisa och läsa texterna samtidigt som man skriver i forumet. Hade man placerat forumet som en del på sidan så hade man inte kunnat ha båda öppna samtidigt utan varit tvungen att växla mellan dem. Dessutom kan en användare komma direkt till forumet utan att behöva gå in på sidan om de tekniska handböckerna, vilket kan vara användbart i framtiden om SWEP bestämmer sig för att ha ett heltäckande forum.

Rate This Page

För att ge SWEP en chans att förbättra texter och bilder så infördes ett frågeformulär efter varje text där besökaren får möjlighet på att svara på om han/hon tyckte att sidan var användbar. Svaren lagras i databasen tillsammans med motsvarande text, vid röstning presenteras det även hur många procent av dem som tidigare röstat som varit nöjda med texten.

Man bör ha i åtanke att varje gång man ger besökare möjlighet att påverka och göra sin röst hörd så finns det alltid risk att någon kommer att missbruka det genom att lägga in oseriösa inlägg i forum eller ge dåligt betyg bara för att man kan det, särskilt om det är en stor och kommersiell sida med många besökare.

Fig 5.15 – Rate-this-page & forum

5.4.2 Sökfunktion

Det finns två typer av användare. De som klickar sig fram till det de letar efter samt de som söker efter det de letar efter. Därför är en lättillgänglig sökfunktion ett användbart redskap för att öka användbarheten på applikationen.

Användaren ges möjlighet att skriva in ord denne vill söka på och resultatet presenteras på en flik med information från vilken bok och kapitel ordet eller orden hittades i, en länk till den samma ges även. Länken öppnas tillfälligt i ett fönster (egentligen en div som går att flytta), på så sätt kan användaren snabbt bläddra mellan de olika resultaten.

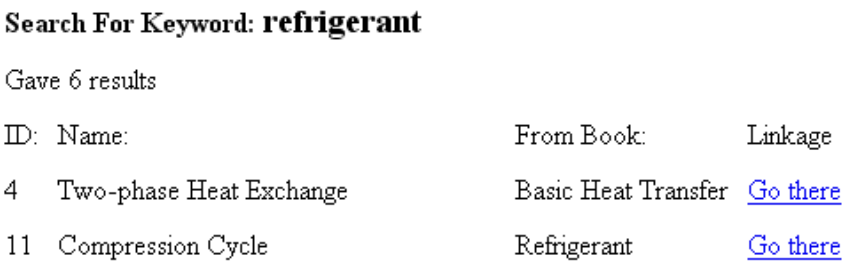

Fig 5.16 – Sökresultaten listas upp

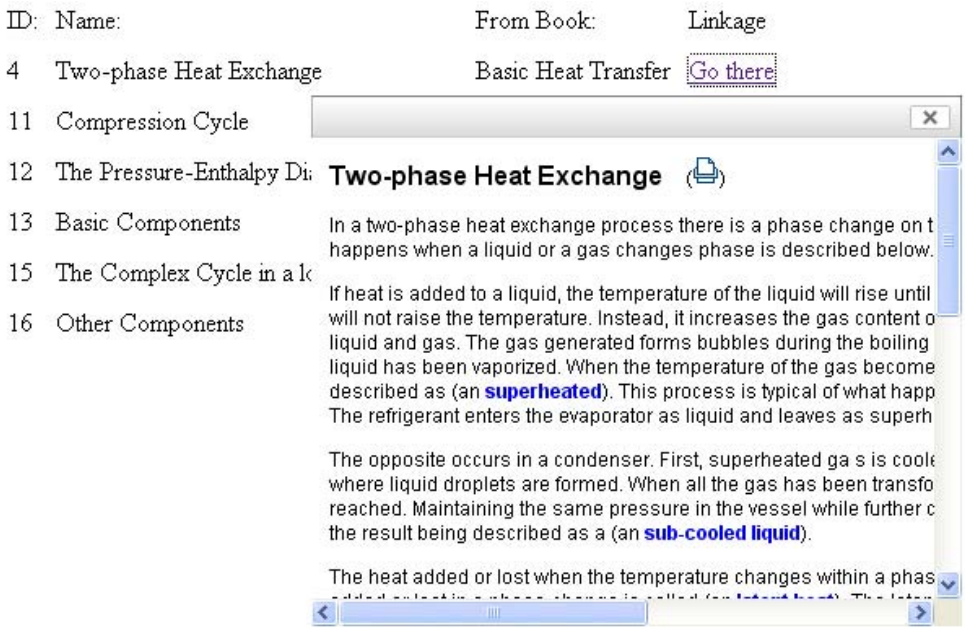

Fig 5.17 – Användaren kan kolla på resultatet

Knappen för sökfunktionen ligger i Flashfilens ”specialmeny”, det innebär att användaren kan komma åt den överallt när denne använder sig av applikationen. En ruta med ett textfält där användaren skriver in ord samt en knapp finns i rutan. När användaren trycker på knappen skickas en adress till den iFrame som ska visa sökresultatet. Adressen innehåller de sökord användaren skrivit in samt den aktuella stilmallen som skall användas, kan se ut som följande:

…/getSearchResult.aspx?keyword=refrigerant&css=12a.css

ASP.NET-dokumentet upprättar därefter en anslutning till databasen och ställer frågan till databasen där den matchar ord i rubriken och texten med det ordet eller de orden som togs emot. Med formatet myISAM på databasen görs det möjligt att använda sig av kommandot MATCH som helt enkelt matchar ord med ord

Begränsningar på sökfunktionen är att den inte kan söka på ord som är mindre än 4 tecken, dessutom matchar den ord för ord, så om användaren skriver in en fras kommer resultatet innefatta samtliga område som innehåller något av orden.

5.4.3 Ändra stilmall

En välbeprövad funktion som många sidor använder sig av är att låta användaren själv bestämma utseende på text. Detta är inte enbart för personligt eller estetiskt utseende utan fyller en viktig funktion. Faktum är att olika personer ser olika bra, framför allt äldre personer som får problem med synen har svårt att se vad som står med en storlek som någon annan tycker är självklar. [27]

Fig 5.18 – Options, ger användaren möjlighet att ändra typsnitt

Från den ständigt åtkomstbara specialmenyn finns alternativet OPTIONS, där kan användaren alternera utseendet. Med tre val på både storlek (12, 14 och 16) och typsnitt (Arial, Times New Roman och Century Gothic) kan man få texten att se ut som man önskar. En gyllene regel säger att man inte bör använda serifer då man presenterar text på skärmar eftersom en text med Serifer kräver flera bildpunkter, men vi har valt att ha med ett typsnitt som ändå använder serifer [28]. Detta av två anledningar: en användare kan föredra typsnittet samt att serifer är att föredra då något är tryckt på papper och eftersom det finns möjlighet att skriva ut kan man välja det typsnittet av den anledningen. Dessutom har det gamla sättet att använda sig av Sans-Serifer blivit mer ifrågasatt [29]. Genom att låta användaren själv bestämma utseende har man garderat sig.

AaBbCc Sans-Serif

AaBbCc Serif

Fig. 5.19 – Serifer

När användaren gjort sitt val och tryckt på knappen (apply eller tillämpa) sparas namnet på den aktuella stilmallen i en global variabel. Namngivningen sker som så att den tar storleken och initialerna från typsnittet och lägger till ”.css” i slutet:

Om användaren valt 16 bildpunkter Times New Roman blir namnet ”16” + ”tnr” + ”.css” = ”16tnr.css”, standard är 12 bildpunkters Arial (12a.css).

Utifrån detta kan två saker hända beroende på vad användaren gjort tidigare. Om det är det första användaren gör så måste denna trycka på ett val i undermenyn för att stilmallen skall börja gälla. I övriga fall kommer texten att ändras direkt på knapptryckning.

Det beror på att varje iFrame redan laddar in introduktionssidorna för de olika böckerna när sidan laddas, det görs med standardstilmallen. Information som skall ha en specialiserad stilmall måste laddas in via Flash och det görs först när man klickat på ett val i undermenyerna. Varje menyknapp har en egen ”bakgrund” som undermenyn finns i, här finns en lokal variabel som jämförs med den globala variabeln där den nya stilmallen är sparad, om dessa inte är lika så skall materialet laddas om fast denna gång med den nya stilmallen.

5.4.4 Uteblivit material

Utöver dessa funktioner hade vi en till i åtanke som inte blivit utvecklad. Den gick under arbetsnamnet myBook och tanken var att användaren skulle kunna kopiera över text och bilder från de befintliga texterna och därefter editera efter eget behov.

Detta föll på många punkter, främst två: formatet som diskuterats i 3.4 som mynnade ut ett eget krav att den skulle vara lätt att använda och lika översiktligt som en vanlig texteditor. Dessutom presenteras numera allt material i en iFrame och då kan man utan problem kopiera den till en vanlig texteditor, chansen att användarna gör så istället är överhängande. Från början var det dessutom tänkt att allt skulle vara i Flash och då kan man inte kopiera text på samma sätt. Den sista faktorn som spelade in var att längden på en del av texterna inte hade gjort det lätt och översiktligt att editera även om vi hade gjort funktionen. Därför valde vi att inte utveckla den.

Enligt metodiken skulle vi ha utfört regelbundna användartestningar genom programmerings olika faser, tyvärr har vi fått prioritera bort dessa då vi blev tvungna att hitta en alternativ lösning när Flash inte kunde tolka HTML-formatet som vi hade tänkt oss.

6 Användartestning

Användartestning är ett effektivt sätt att hitta problem och fel som utvecklarna själv missat. När man jobbar med en och samma sak under en längre tid så får man en viss blindhet för sin egen design och användbarhet. Det beror på att man blir så pass fäst och insatt i hur det hela fungerar så man tycker allt är självklart. Steve Krug är en användbarhetsexpert som skrivit boken ”Don’t Make Me Think”, den behandlar webbdesign från en användares perspektiv och visar på fakta för att underlätta navigation och samförstånd mellan design och användare. Han har beskrivit användartestning med följande punkter:

”Att testa på en är bättre än att inte testa alls” – Det kvittar om testet är väl genomfört eller om användaren inte var tillräckligt engagerad, man får alltid ut något att förbättra.

”Att testa på en person i början av sitt projekt är bättre än att testa 50 stycken vid slutskedet” – Ju tidigare man involverar användartestning på sin design desto enklare, billigare och snabbare går det att ändra någonting.

”Det är överskattat att testa enbart på sin målgrupp” – Visst är det bra att testa sin design på sin tilltänkta målgrupp, men fler tester på slumpvis valda personer kan identifiera andra typer av problem en de som är av samma målgrupp.

”Testning är en iterativ process” – Testningen bör inte ske endast en gång, utan ett test, utvärdera testet och fixa problemet – testa igen. Upprepa.

Utifrån detta lär man sig att man bör testa under hela utvecklingsfasen redan innan programmeringsfasen, helst under designfasen för att underlätta eventuella förändringar.

Själva problemet med vårt arbete var att vi var tvungna att hitta en ny lösning när Flash-applikationen började ta form, således prioriterades det bort att testa användbarheten på menyer och sådant under utvecklingsprocessen. Men för att inte underskatta användartestningen så genomförde vi några test i slutfasen för att få underlag för framtida förbättringar. Även om det stred mot Steve Krugs påstående så stämmer det fortfarande att det är bättre att testa på en än ingen alls. [30]

6.1 Hur testar man?

Det finns en del olika test som man kan genomföra för att få fram information från olika delar av sin design och sina funktioner. För att undersöka hur en användare uppfattar överblicken kan man genomföra ett så kallat *”Trunk Test”*. Det innebär att man navigerar in en användare på en sida och denna skall snabbt kunna identifiera var denne befinner sig, vilket företag det handlar om och gärna få en bild om vad det handlar om. Ett annat sätt att testa mer specifika funktioner är att låta användaren genomföra olika testfall. Där har man själv som utvecklare skrivit en uppgift med korta beskrivningar som skall få användaren att använda sig av funktioner för att se om de är lätta att förstå sig på och om de går att förbättra. I vårt fall kommer vi att använda testfall för att testa olika funktioner på applikationen. [31]

6.2 Hur många ska man testa på?

Jacob Nielsen och Tom Landauer har tillsammans kommit fram till att det perfekta antalet att göra ett användartest på är fem personer. De kom fram till en formel som pekar på att test på fem stycken personer täcker 85% av alla användbarhetsrelaterade problem.

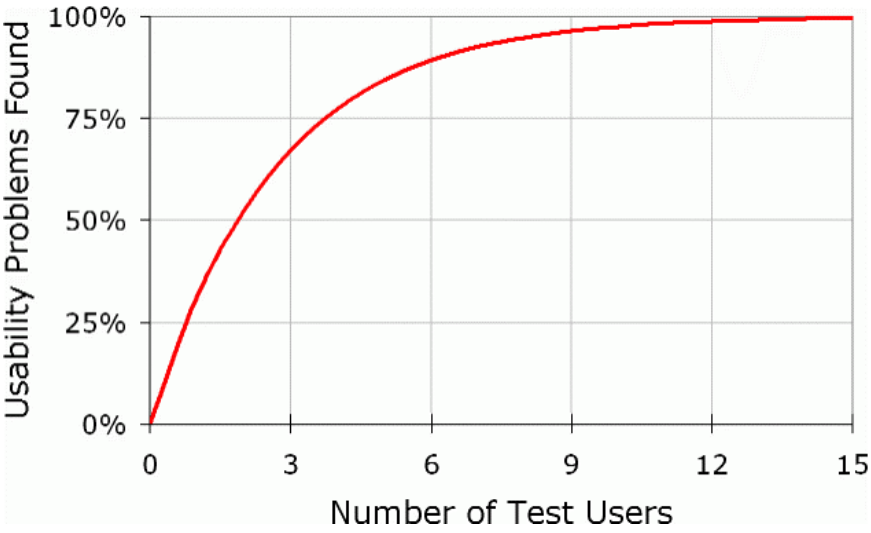

Fig. 6.1 – Användartestningskurvan

Kurvan visar ett test på att 15 personer kommer att täcka de flesta problemen, men som Steve Krug nämnt är testning en iterativ process som skall göras om flera gånger. Istället bör man göra 5 test med olika personer för att identifiera de

flesta problemen, utvärdera och göra om design och funktionalitet. Därefter görs 5 nya test och en ny efterföljande utvärdering. Tredje gången har man täckt in de flesta problem och testat de nya lösningarna. Jacob Nielsen menar att den här metoden är både billigare och effektivare än att testa med 15 personer direkt och sen göra en utvärdering och omarbetning av sin lösning. Dessutom kommer ett test på 15 personer ge väldigt lika resultat om de tidigare problemen inte görs om. [32]

6.3 Testfall

Testfallet är utformat på ett så omfattande sätt för att låta varje testperson utnyttja samtliga funktioner som finns i applikationen. Testfallet genomfördes på Campus i Helsingborg och vi testade på fem personer. Testarna fick direktiv och vi antecknade vad de gjorde samt ställde frågor beroende på de val de gjorde. De numrerade punkterna är de instruktioner som gavs till testpersonerna.

1. Gå in på fliken Refrigerant
2. Välj Compression Cycle
3. Välj Basic Components
4. Tryck på figur 2.7
5. Läs texten om 2.7
6. Diskutera ämnet i forumet
7. Gör en sökning på ordet, ”cold”
8. Betygsätt en av sidorna som resulterade från föregående punkt
9. Gå in på fliken Basic Heat Transfer
10. Välj Thermal Length
11. Undersök betydelsen av ett ord
12. Ändra utseende på applikationen
13. Skriv ut sidan (Thermal Length)

Efter att testet var avslutat, lät vi användaren ge sina synpunkter på applikationen och ställde följdfrågor för att få ett så precist svar som möjligt. Resultatet kan användas för framtida förbättringar.

6.4 Testresultat och utvärdering

Testet gjordes på fem personer med varierande kunskap. Samtliga hade datorvana till vardagligt bruk, medan två av dem hade stor kunskap om programmering.

Den första testpersonen hade minst datorvana av samtliga på testet, steg för steg gick hon igenom direktiven och ett av de första problem hon hittade var att det var brister för de numrerade bilderna. När användaren skulle läsa texten bakom

bild 2.7 använde hon möjligheten att flytta denna, vilket hon kände var logiskt. För forumet använde hon länken längst ner under innehållet vilket ledde direkt in på ämnet. Efteråt tyckte hon att det var en estetiskt tilltalande applikation och gillade funktionerna med ordförklaring och bildvisning, negativt var att Menyn i Flash inte riktigt hängde med när man använde sökfunktionen.

Andra personen hade stor datorvana och gick igenom direktiven snabbt och navigerade sig själv runt för att se hur det fungerade. Synpunkter han hade var att forumet (via huvudmenyn) kunde öppnas i många exemplar och att detta borde begränsas. Han tyckte att sökresultatet inte presenterades på ett önskvärt sätt samt att det blev konstigt emellanåt med att Flashmenyn inte hängde med i samband att man hade gjort en sökning eller besökt forumet. I övrigt tyckte han att det var bra med en tillbaka-knapp då man besöker glossary genom att trycka på ett ord i texten.

Tredje personen hade inget mer att påpeka än de tidigare två, men berömde även han funktionen med ordförklaringen.

Fjärde personen täckte även hon upp delar av de problem som de två första redan tagit upp men vill använda sig av webbläsarens tillbaka vid något tillfälle, vilket inte fungerar som vanligt när man använder Flash och iFrames.

Femte personen var aningen stressad och testet gjordes snabbt, användaren observerade inte att menyn inte uppdaterades vid ett tillfälle och gav som kommentar efteråt att det såg bra ut.

Den gemensamma nämnaren när testresultaten summerades blev att menyn inte riktigt fungerade som den skulle. Detta inträffar ibland då man använt sökfunktionen eller besökt forumet via huvudmenyn. Problemet blev vi medvetna om och ligger förmodligen i både Flash och JavaScript. Det stora problemet är att Flash inte kan ta emot information om det inte hat begärt det själv först.

Småfel se där vi numrerat bilder fel har redigerats innan leverans till SWEP. Ingen lösning för att begränsa forum-fönsterna har tagits fram men kan finnas i åtanke för framtida förändringar, det är inte säkert att SWEP kommer att använda sig av detta forum.

7 Resultat

7.1 Återblick

Kontakt togs med SWEP och det beslutades att vi skulle uppdatera deras tekniska handböcker och presentera dessa tillgängliga för alla med Internet. Önskemålet från SWEP var att Flash skulle användas som grund och hämta HTML-formaterad text från och databas för att på så sätt kunna fungera på ett dynamiskt sätt där det ska vara lätt att lägga till nya böcker och kapitel.

Kan Flash (med ActionScript 2) användas som plattform för HTML-formaterade texter och dynamiskt generera nya flikar och kapitel?

Den dynamiska biten fungerade bra och att hämta in från en databas fungerade med en ASP.NET-sida som mellan hand, men att presentera innehållet med HTML-formatering fungerade inte. En alternativ lösning har tagits fram som presenterar innehållet i iFrames istället för i Flash. Applikationen fungerade nu tillfredställande och ett användartest genomfördes. Testets utfall pekar på att samspelet mellan Flash och innehållet inte riktigt håller måttet.

En kontroll mot kraven visade att samtliga var uppfyllda utom A.1.1 som handlade om att startsidan skall innehålla information om samtliga böcker. Detta har inte uppfyllts då vi inte skulle stå för innehållet och eftersom det är SWEP som skall använda applikationen får de själv disponera ytan på startsidan.

7.2 Slutsats

Den här applikationen hade som största prioritet att presentera innehållet och därför lämpar sig Flash inte för uppgiften. Det fick vi befara då ett av våra största misstag var att inte helt undersöka till vilken grad tolkningen av HTML i Flash var korrekt. Därför spillde vi tid på att göra en lösning i Flash som inte fungerade och sedan hitta en alternativ lösning.

När det beslutet hade tagits hade vi besvarat frågeställningen om Flash var en lämplig plattform för presentation av stora mängder HTML-formaterad text, vilket det inte var. Den andra frågan handlade om Flash hade stöd för den dynamik som krävdes för att integrera användaren. När beslutet togs att inte utveckla en funktion, myBOOK, så kunde denna fråga inte besvaras med rättvisa. Delar av applikationen använder sig av Flash för att ha ett samspel med användaren men de flesta är i gjorda med JavaScript.

Som slutsats kan vi säga att:

Flash lämpar sig bäst som spel och bildvisare eller mindre applikationer där innehållet formateras inne i Flash och inte hämtas in via databas. Om Flash är grunden i en hemsida har man eliminerat webbläsarens inbyggda funktioner som exempelvis tillbaka-knappen.

Använd Flash som komplement och inslag på sidor och inte iFrames och JavaScript för att komplettera Flash.

7.3 Framtidsutsikter

Med testresultat och en slutsats på papper skulle vi vilja lämna över arbetet till SWEP med en del synpunkter. Sidan fungerar i användbart skikt som det är nu men för att eliminera de fel som uppstår så bör menysystemet ändras så att det består av HTML. Då frigör man även de funktioner som är inbyggda i webbläsaren. Vi är ändå nöjda med resultatet eftersom vi fick tänka ut en alternativ lösning.

APPENDIX A: Krav

A Systemkrav

A.1 *Navigering*
A.1.1 På startsidan skall det finnas en kort introduktion till de andra delarna.
A.1.2 Menyn skall vara uppdelad i 3 olika delar:
A.1.2.1 En huvudmeny som skall innehålla:
- A.1.2.1.1 Home
- A.1.2.1.2 Basic Heat Transfer
- A.1.2.1.3 Refrigerant
- A.1.2.1.4 Heating
- A.1.2.1.5 Industrial
- A.1.2.1.6 Forum

A.1.2.2 En undermeny som skall innehålla de olika kapitlen i handböckerna.
A.1.2.3 En specialmeny som skall innehålla:
- A.1.2.3.1 Glossary
- A.1.2.3.2 Option
- A.1.2.3.3 Search
- A.1.2.3.4 Sitemap
- A.1.2.3.5 Contact
- A.1.2.3.6 FAQ

A.2 Har utgått, se D Utgångna krav

A.3 *Forum*
A.3.1 I handböckerna ska det finnas länkar för att skapa/läsa motsvarande topic.

A.4 Har utgått, se D Utgångna krav

A.5 *Handböckerna*
A.5.1 Ska innefatta Basic Heat Transfer, Refrigerant, Heating samt industrial.

A.5.2 Skall hämta sin meny från databas vid knapptryckning.
A.5.3 Vid tryckning på en menyrad skall minst 1 av följande fall inträffa:
- A.5.3.1 Texten som motsvarar knappen skall presenteras.
- A.5.3.2 Om menyn har en undermeny skall denna kollapsa samtidigt som krav A.5.3.1

A.5.4 Varje handbok skall behålla sin status om man byter bok.
A.5.5 Texterna skall vara HTML-formaterade.
A.5.6 Texterna skall ha en gemensam stilmall.
A.5.7 Texterna skall kunna använda sig av JavaScript.

B Datakrav

Samtliga krav har utgått se D – Utgångna krav

C Säkerhet

Samtliga krav har utgått se D – Utgångna krav

D Utgångna krav

Av anledning för eventuella framtida bruk samlas de utgångna kraven här, en kort introduktionstext ges och förklarar varför de blev borttagna.

myBook var arbetsnamnet på en funktion som aldrig utvecklades. Tanken var att den skulle fungera som en klippbok där användaren skulle kunna plocka ut delar från andra sidor i applikationen samt redigera denna. Pågrund av att formatet inte kunde bli lika bra som en vanlig text-editor och att tiden rann iväg valde vi att inte genomföra denna funktion.

A.2 *myBook*
A.2.1 Text från handböckerna skall gå att lägga in på myBook.
A.2.2 Text från forum skall gå att lägga in på myBook.
A.2.3 Allt i myBook skall gå att redigeras på följande sätt.
  A.2.3.1 Man skall kunna ändra storlek på text blockvis.
  A.2.3.2 Man skall kunna ändra stil (kursiv/fet) på text blockvis.
  A.2.3.3 Man skall kunna omorganisera textblock och bilder.
  A.2.3.4 All text i myBook skall gå att editera
A.2.4 Strukturen i myBook skall vara utskriftsanpassad för A4.

För att kunna spara sina egna sidor och skriva i Forumet måste varje användare ha ett eget konto. När myBook togs bort och beslutet att inte göra ett eget forum var kraven på dessa inte nödvändiga att ha kvar.

A.4 *Användarkonton*
A.4.1 Det skall finnas personliga konton.
A.4.2 Kontona ska vara aktiva i 2 månader.
  A.4.2.1 Vid inloggningstillfället förlängs giltighetstiden till 2 månader från inloggningstillfället.

A.4.2.2 Påminnelse skickas ut om man vill förlänga sitt konto efter 50 dagars inaktivitet.
A.4.2.3 Kontot tas bort automatiskt efter 2 månader inaktivitet.
A.4.3 Det ska gå att logga in.
A.4.4 Det ska gå att logga ut.
A.4.5 Vid inloggning hämtas eventuella ändringar man gjort på myBook.
A.4.6 Ändringar som görs under inloggad period sparas temporärt var 5:e minut.
A.4.7 För att göra sina temporära ändringar permanenta så måste man själv bekräfta detta.
A.4.8 Vid inaktivitet under inloggad session ska man loggas ut automatiskt efter 15min.
A.4.8.1 Vid automatisk utloggning sparas användarens temporära ändringar.
A.4.8.2 Vid nästa inloggning skall en förfrågan ges om man vill återgå till senast sparade inställningar eller senaste temporära ändringar.

När användarkontona togs bort så fanns det ingen anledning att ha kvar Datakraven och säkerhetskraven då båda dessa behandlar och beskriver hur registreringen och informationen skall hanteras.

B Datakrav

B.1 *Regristrering*
B.1.1 När en användare registrerar sig skall följande data registreras:
B.1.1.1 Namn (efternamn, förnamn)
B.1.1.2 Användarnamn
B.1.1.3 Lösenord
B.1.1.4 E-post
B.1.2 Om användaren tappar bort sitt lösenord och begär ett nytt via onlineformuläret skall ett nytt lösenord bestående av 6 slumpmässiga tecken [A-z] [0-9] genereras.
B.1.2.1 Ett mail skall skickas till användaren som måste bekräfta att denna ansökt om nytt lösenord.
B.1.2.2 Det nya lösenordet i B.1.3 skall mailas ut till den registrerade e-mailen.

C Säkerhet

C.1 *Lösenordshantering*
C.1.1 Lösenord skall krypteras.
C.1.2 Lösenordet skall sparas som en hashsumma i databasen.
C.1.3 Vid inloggning skall lösenordet hashas.
C.1.3.1 Det hashade lösenordet jämförs med hashsumman i databasen.

APPENDIX B: Referenser

[1] http://www.swep.net/index.php?tpl=page0&lang=se&id=322 10/7 2007
[2] http://www.swep.net/index.php?tpl=page0&lang=se&id=79 10/7 2007
[3] http://www.swep.net/index.php?tpl=page0&lang=se&id=78 10/7 2007
[4] http://www.swep.net/index.php?tpl=page0&lang=se&id=111 10/7 2007
[5] Kontorsguiden, sida 24, ISBN: 91-7968-384-3 framtagen av SIF oktober 2007
[6] http://www.useit.com 28/11 2007
[7] http://www.w3schools.com/browsers/browsers_display.asp 28/11 2007
[8] http://www.techworld.idg.se/2.2524/1.120095 &http://www.samsung.com/se/presscenter/samsunglocalpress/samsunglocalpress_20050221_0000098668.asp 28/11 2007
[9] http://swescrapbook.blogg.se/280806120622_ngra_grunder_i_frglra.html 28/11 2007
[10] http://www.techdocs.ku.edu/docs/flash_mx2004_introduction.pdf 10/7 2007
[11] http://www.w3schools.com/flash/introduction.htm 10/7 2007
[12] www.w3schools.com/aspnet/aspnet_intro.asp 28/11 2007
[13] http://www.codeproject.com/aspnet/aspnetintro.asp 28/11 2007
[14] http://msdn2.microsoft.com/en-us/library/4w3ex9c2(vs.71).aspx 28/11 2007
[15] http://www.pcquest.com/content/search/showarticle.asp?arid=47162 10/7 2007
[16] http://www.scit.wlv.ac.uk/~jphb/javascript/intro.html 10/7 2007
[17] http://www.w3schools.com/js/js_intro.asp 28/11 2007
[18] Learn How To Program Using Any Web Browser, sida 9, ISBN: 1-59059-113-5 författad av Harold Davies 2004
[19] http://www.w3schools.com/browsers/browsers_stats.asp 28/11 2007
[20] http://www.ozoneasylum.com/9671. 28/11 2007
[21] http://www.dynamicdrive.com/dynamicindex17/iframessi2.htm 28/11 2007
[22] http://www.vr.se/huvudmeny/forskningvistodjer/humanioraochsamhallsvetenskap/slutfordaprojekt20012005/darforardetsalattattminnasbilder.4.aad30e310abcb9735780004948.html 28/11 2007
[23] http://www.pat-burt.com/web-usability/how-to-optimize-your-body-text-for-readability/ 28/11 2007
[24] http://www.codefidelity.com/blog/?page_id=7 28/11 2007
[25] http://www.bosrup.com/web/overlib/
[26] http://www.phpbb.com/downloads/ 28/11 2007
[27] http://www.hp.com/ergo/pdfs/297660-102.pdf avsnittet om ögon 28/11 2007
[28] http://web20kurs.wetpaint.com/page/Anv%C3%A4ndarv%C3%A4nlighet?t=anon 28/11 2007

[29] http://kurafire.net/log/archive/2005/07/23/typography-serif-vs-sans-serif 28/11 2007
[30] Don't Make Me Think, sida 140-146, ISBN: 0-7897-2310-7 skriven av Steve Krug
[31] Don't Make Me Think, sida 87-89, ISBN: 0-7897-2310-7 skriven av Steve Krug
[32] http://www.useit.com/alertbox/20000319.html 28/11 2007