

Datalogi 2B - Bachelorprojekt

Webdesign for Copenhagen STL

af Magdalena "Lenka" Otap

Datalogisk Institut, Københavns Universitet
Copenhagen STL Rapport 2001-12, maj 2001

Indholdsfortegnelse

1	Indledning	1
2	Nuværende situation	2
2.1	Egne tanker omkring projektet	3
3	Behovsanalyse	4
3.1	Hvorfor inddrage brugeren?	4
3.2	Hvem er brugeren?	4
3.3	Samtaler med brugeren	4
3.3.1	Møde d. 5/2-2001	4
3.3.2	Møde d. 19/2-2001	5
3.3.3	Forberedelser til interviews	6
3.3.4	Spørgsmål til interview	6
3.3.5	Interview med Lars Yde d. 26/2	8
3.3.6	Interview med Jyrki Katajainen d. 26/2	9
3.4	Andre udtalelser	10
3.5	Delkonklusion af behovsanalysen	10
4	Designanalyse	11
4.1	Database eller ej	11
4.2	Struktur	12
4.2.1	Main Page	12
4.2.2	News	13
4.2.3	Downloads	14
4.2.4	Source Code	15
4.2.5	Design Documents	16
4.2.6	Contributors	16
4.2.7	New Contributors	17
4.2.8	Links og Literature	17
4.2.9	Bug Reports	18
4.2.10	Mailing List og Contact Us	18
4.3	Administration	18
4.3.1	News	18
4.3.2	News via. mail	18
4.3.3	Modules	20
4.3.4	Contributors	20
4.3.5	Links	20
4.4	Database	20
4.4.1	News	20
4.4.2	Contributor	21
4.4.3	Modul og person/modul-relationen	22
4.4.4	Links	22
4.5	Redskaber	23
4.5.1	Scriptsprog	23

4.5.2	Database	24
4.5.3	Doxygen	24
4.5.4	CVS	24
4.5.5	Bug Tracker	25
4.6	Præsentation	25
4.7	Delkonklusion af designanalysen	26
5	Implementation	27
6	Afprøvning	28
6.1	Korrekthed: HTML	28
6.2	Korrekthed: scripts	28
6.3	Hastighed: load tid	28
7	Konklusion	30
A	Bilag - De tre designeksempler	33
B	Bilag - Det færdige Design	36
C	Bilag - Afprøvning	39
C.1	Test af korrekthed: HTML	39
C.2	Test af korrekthed: scripts	40
C.3	Test af hastighed - load tid	41
D	Bilag - "Program"-udskrifter	42
D.1	index-header.html	42
D.2	index.php	46
D.3	index-footer.php	47
D.4	news.php	48
D.5	archived_news.php	49
D.6	downloads.php	50
D.7	source.php	51
D.8	docs.php	52
D.9	contributors.php	54
D.10	newcontrib.php	57
D.11	links.php	58
D.12	literature.php	59
D.13	bugs.php	60
D.14	mailing.php	61
D.15	contact.php	62
D.16	admin/index.php	63
D.17	admin/contributor.php	64
D.18	admin/update_person.php	68
D.19	admin/module.php	70
D.20	admin/update_module.php	72
D.21	admin/news.php	73

D.22 admin/update_news.php	76
D.23 admin/links.php	77
D.24 admin/update_cat.php	81
D.25 admin/update_link.php	82
D.26 tables.sql	84

1 Indledning

Dette er en besvarelse af bachelorprojekt-opgaven på datalogi 2B. Opgaven er en selvstændig opgave, afleveret d. 17. maj 2001.

Formålet med opgaven er at lave webdesign for Copenhagen STL. Opgaven indebærer bl.a. at finde frem til hvilke behov Copenhagen STL har, og derefter analysere sig frem til, hvordan disse behov skal dækkes med et design. Tredje del af opgaven vil være at implementere det design, som der er fundet frem til vha. behovsanalysen.

Behovsanalysen skal udføres vha. møder og interviews med medlemmer af Copenhagen STL.

Designanalysen skal bygge på behovsanalysen og bruges til at overveje hvordan det egentlige design skal implementeres.

Designet skal afspejle de valg, som der findes frem til, i designanalysen.

De to analyse-dele er det primære formål med opgaven, mens implementationen af designet er et sekundært formål.

2 Nuværende situation

Hjemmesiden er på sin nuværende form [1], blot en meget “lang” side, med indholdsfortegnelse øverst på siden, som indeholder hyperlinks, som peger ned til overskrifter længere nede på siden.

Indholdsfortegnelsen indeholder på nuværende tidspunkt (d. 18. februar 2001):

- Project outline
- Window to our CVS repository
- Copyright
- Contributors
- Related links
- Literature

Desuden findes der her links til et kursus (“Performance Engineering”), samt links til Copenhagen STLs ugentlige informationsmøder.

Project Outline

“Project Outline” indeholder en projektbeskrivelse, dvs. en forklaring af hvad STL og Copenhagen STL er, samt en forklaring af hvad formålet med projektet er.

Window to our CVS repository

Vinduet til CVS repositoret giver en besøgende mulighed for at følge med i udvikling af modulerne.

Copyright

“Copyright” siger vist sig selv: med en enkelt sætning står der hvem, der har copyright på Copenhagen STL.

Contributors

Her kan man finde en liste af alle bidragyderne, med oversigt over hvilke komponenter de har været eller er med til at lave. De bidragydere der har en hjemmeside, har fået tildelt et link til denne. Under “Contributors” kan man desuden se en udskrift af, hvilke komponenter, der ikke er taget af nogen endnu.

Related links

Under “Related links” er en liste af relaterede links delt op efter emne.

Literature

Her kan man finde en litteratur-liste over de bøger som kan være en hjælp til C++ programmering og STL.

2.1 **Egne tanker omkring projektet**

Hjemmesiden på nuværende form er

- lidt kedelig at kigge på, og kunne trænge til lidt form og farve.
- en enkel statisk HTML-side, hvilket gør det mere besværligt at vedligeholde bl.a. “Contributors”-punktet og andres punkter, som kræver vedligeholdelse.
- mangler ekstra information for en besøgende, som gerne vil vide mere om projektet.
- CVS repositoret er forvirrende at finde rundt i.

Først og fremmest er det vigtigt at fastlægge, hvad formålet med siden er, samt hvem hjemmesiden henvender sig til. Se kapitel 3 for dette. Dette er vigtigt, da det giver et udgangspunkt, man kan gå ud fra, når hjemmesiden skal designes. Afhængigt af formålet, kan projektet præsenteres med forskellige informationer og afhængigt af målgruppen kan informationerne præsenteres på forskellige måder.

3 Behovsanalyse

En behovsanalyse er nødvendig for at kunne designe en hjemmeside, som passer til de involveredes (brugernes) behov. I dette afsnit beskrives de samtaler og møder jeg har haft med brugerne, dvs. bidragyderne til Copenhagen STL, og som senere vil bruges, som et grundlag for designanalysen.

3.1 Hvorfor inddrage brugeren?

Der kan være flere grunde til at inddrage en bruger i udviklingen af et system [SYS, s. 169]. I tilfældet af udvikling af en hjemmeside (som vel egentlig godt kan betegnes som en slags system), eller mere specifikt, Copenhagen STLs hjemmeside, er det en god idé at inddrage brugeren pga.:

1. Det er brugerens hjemmeside og derfor skal brugeren kunne have indflydelse på designet.
2. Brugeren er domæneekspert i den forstand, at brugeren ved hvilke informationer der skal være på sitet (og hvilke der kommer til i fremtiden).

3.2 Hvem er brugeren?

“Brugeren” er i dette tilfælde bidragyderne til Copenhagen STL. Man kan sige at der her både er tale om en operativ bruger, en berørt bruger og en domæneekspert [SYS, s. 175].

- Den operative bruger, fordi det i sidste ende er til disse brugere, produktet skal laves. Det er disse brugere der, efter at designet er færdigt, vil stå for at administrere hjemmesiden og videreudvikle den.
- Den berørte bruger, da det bl.a. er disse brugere, der også vil komme til at benytte sig af hjemmesiden, til bl.a. at holde styr på møder og udviklede komponenter oa.
- Sidst, men ikke mindst, er brugerne også domæne-eksperter, da det er dem der er “eksperter” i hjemmesidens emne.

3.3 Samtaler med brugeren

I dette afsnit beskrives de møder og interviews, som jeg har haft med Copenhagen STLs udviklere.

3.3.1 Møde d. 5/2-2001

Til dette møde var vi 4 deltagere: Jyrki Katajainen, Lars Yde, Bjarke Buur Mortensen og jeg. Vi tog diskussionen fra bunden af, og diskuterede formål med siden og hvilken målgruppe den sigtede efter. Der blev, efter lidt snakken frem og tilbage, enighed om, at sidens målgruppe både var tekniske udviklere,

undervisere og studerende, og formålet var at give information og dokumentation af C++ STL til disse.

Vi kom ind på brug af databaser, som ville være smart at have for at holde siden dynamisk og nem at vedligeholde. Jeg foreslog databaser til bl.a.:

- alle tekstfilerne
- alle bidragyderne
- komponenter
- hovedpunkter
- links
- litteratur

Databasen til bidragyderne kunne især være brugbar, hvis der besluttedes og til at lave en lille personificeret hjemmeside med simple informationer for hver bidragyder.

Jeg nævnte kort at jeg forventede at bruge php som scriptsprog. Vi kom også ind på CGI-scripts.

En ide om at have bug tracking kom op. Her blev nævnt bl.a. Bugzilla [2] og RequestTracker [3].

3.3.2 Møde d. 19/2-2001

Foruden Jyrki Katajainen og mig, var der 5 deltagere til dette møde. To af deltagere var de samme som nævnt i forrige kapitel, altså Lars Yde og Bjarke Buur Mortensen.

Det var meningen, at jeg til dette møde blot skulle finde nogle frivillige til interviews, så derfor havde jeg ikke forberedt nogle spørgsmål. Vi kom dog alligevel ind i en diskussion om design af hjemmesiden, men denne var temmelig ustruktureret, som man jo også kunne forvente. I denne fase har jeg stadig brug for brainstorm, så al input var velkommen.

Diskussionen faldt denne gang igen på brug af databaser. Der var flere i mod dette, da det virkede som et unødvendigt indgreb. I stedet blev jeg rådet til at bruge tekstfiler, som jeg hentede informationer fra, vha. scripts - medmindre jeg havde nogle rigtig gode argumenter for at benytte mig af en database.

I stedet for at aftale interview-tidspunkter med deltagerne til mødet, blev vi enige om, at jeg skulle sende en mail ud til hele Copenhagen STLs mailingliste. Det gjorde jeg så.

3.3.3 Forberedelser til interviews

Det lykkedes at skaffe en aftale om et interview med to involverede i Copenhagen STL: Lars Yde (3.3.5) og Jyrki Katajainen (3.3.6).

For at lave brugbare interviews, har jeg fundet nogle gode regler [6] at holde sig til:

- Stil ikke ja/nej spørgsmål
- Stil åbne spørgsmål
- Stil spørgsmål som vil blive forstået ens af alle udspurgte
- Stil samme spørgsmål på samme måde til alle
- Definer hvilke slags svar der vil være acceptable

Objektiver (et mål til spørgsmålene):

- Lav en rød tråd og skriv derefter spørgsmålene.
- Hvilken slags information skal spørgsmålet bruges til?
- Hvad er formålet med spørgsmålet?

3.3.4 Spørgsmål til interview

Nogle af disse spørgsmål er allerede diskuteret til møderne, men for at få det hele på plads samtidig, har jeg valgt at tage de spørgsmål, som vi allerede har diskuteret, med i dette interview.

Formål med siden?

Objektiv: Formålet vil afspejle sig i den information der skal ligge på siden.

- Sidst fik jeg at vide at formålet er at informere og dokumentere - uddybes?
- Evt - hvilket formål vil der være i at sætte personlige hjemmesider op med beskrivelser?
- Tiltrækning af nye udviklere? Er det et formål?

Målgruppe?

Objektiv: Udviklere og brugere tænker forskelligt. Så design og udseende er afhængigt af målgruppen

- Sidst fik jeg at vide at målgruppen var undervisere, tekniske udviklere og studerende. Er der andre?
- Hvordan vil I gerne virke på målgruppen - Seriose? Spændende? Neutrale? Sjove?

Behovsanalyse:

Hvad er der brug for? Objektiv: behovene vil afspejle designet

- Har du tænkt over hvilke (ekstra) informationer der skal være på siden? (Hvilke informationer skal der være plads til? (fx,
 - skal der være flere hovedpunkter? Eller evt. tilføjes underpunkter?
 - Hvordan regner du med at siden vil udvikle sig i fremtiden?))
- Skal informationen være delt op som den er nu? Hvilken opdeling → Hvilken navigation? (jfr. hoved- og evt. underpunkter)
- Hvem skal videreudvikle siden?

CVS:

Objektiv: hvor meget skal jeg sætte mig ind i CVS?

- CVS til udvikling af web, hvordan skal det bruges?
 - kun til udvikling af siden?
- CVS repository
 - skal den doxygen¹-genererede side ind-’opereres’ i resten af designet? (eller bare lades være?)

Teknisk:

Objektiv: Hvilke redskaber skal jeg bruge?

- Database vs. tekstfil? (afhænger vel af fremtidig udvikling?)
- Hvilket scriptssprog? Betyder det noget?
- Bugzilla [2] - Bugrequest.. hvordan skal det bruges? nntp?
- Kan det jeg vil, laves på serveren?

Udseende/navigering:

Objektiv: Hvordan skal designet se ud?

- Hvilke af de 3 eksempler² tiltaler dig/tiltaler dig ikke, og hvorfor/hvorfor ikke.

¹Læs mere om Doxygen i afsnit 4.5.3

²Se Bilag A - forslagene er lavet som simple billeder (.jpg) som har fået en browserramme udenom, for at det skulle være lettere at sætte sig ind i, hvordan det færdige resultat ville se ud.

3.3.5 Interview med Lars Yde d. 26/2

Interviewet blev båndet og derefter klippet og klistret sammen til nedestående referat.

Om formålet (foruden alt det, som vi har talt om til de forrige møder):
Siden skal også bruges til at tiltrække nye udviklere.

Om målgruppen:
Tekniske udviklere, undervisere, studerende. Udseendet skal derfor være seriøst, men heller ikke børs- eller bankrådgiveragtigt. Ikke for farverigt.

Om Doxygen:
Doxygen [7] er fastlåst. Men man kan påvirke hvordan det ser ud i headeren. Dokumentationen skal ikke varieres, men kan skræddersys. *“Vi har haft diskussioner om det skulle se sådan ud”*. Doxygen-siderne skal integreres i resten af designet.

Om de tre forslag (se bilag A):

1. Layout: fornuftigt, men ikke let at udvide.
2. Ikke vild med - virker for blokopdelt.
3. Smart skrifttype til cphstl-logo, måske for “smart”. Ikke god angivelse af hvem vi er. Mht. skrifttype - “Impact” er godt. Diskret, ikke for fancy, men afdæmpet og troværdig.

Generelt om de tre forslag:

Godt at indikere hvor man befinder sig på siden.

Umiddelbart er forslag 1 bedst, men der mangler bare noget nederst for at bryde siden og støtte letlæseligheden. Det kunne fx være copyright-teksten.

“Det ville være rart om designet med en forholdsvis lille bjælke øverst på skærmen kunne bibeholdes, men samtidig give plads til et variabelt antal menupunkter. Dette kunne f.eks. gøres ved at bruge en slags “bladkarrusel” princip eller måske et antal små faneblade.”

Om indholdet:

Der skal også være sider for de enkelte bidragsydere, hvoraf der fremgår kortfattede biografiske oplysninger samt data om hvilke bidrag de har ydet til cphstl. Disse sider skal vedligeholdes af de enkelte bidragsydere. Formålet med disse sider, er at fortælle hvem der står bag. *“Folk er nysgerrige”*.

Om fremtidig situation:

Når der kommer “stable releases”, skal der også være en download-sektion. Punktopdelingen skal vist være nogenlunde den samme, men det er stadig til diskussion. Copyright skal dog ikke forblive som et punkt for sig selv.

Komponenterne skal heller ikke ligge under "Contributors". Der skal desuden være plads til links til sponsorer på hovedsiden.

Når den besøgende ønsker at se dokumentationen skal hun/han evt. have mulighed for at vælge mellem åbning af nyt vindue og åbning i eksisterende vindue.

Om at have en database:

Ville måske gøre det lettere at vedligeholde.

Om bugtracking:

Kan klares enten med fx nntp eller RequestTracker Snak evt. med EDB-afdelingen.

3.3.6 Interview med Jyrki Katajainen d. 26/2

Interviewet blev båndet og derefter klippet og klistret sammen til nedestående referat.

Om Formål (foruden alt det som vi har snakket om til de forrige møder):

Siden skal primært tjene som undervisningsmateriale. Det er også et formål at tiltrække nye bidragydere, men der er ikke så stort behov, da de fleste komponenter er lavet.

Om fremtidig situation:

Det er vigtigt at finde frem til hvad det er vi forsøger på. Disse punkter skal med:

Program - modulerne

Doxygen skal indopereres - strukturen skal ikke ændres, men skal være mere læsevenlig. Visse filer generes automatisk, visse er statiske. Der må gerne ændres på udseende.

Det er ikke klart, hvad der skal vises til omverden, og hvad der skal være hemmeligt.

Designdokumentation - rapporter

Der er to slags designdokumenter: Skriftlige projekter (rapporter) og artikler.

Udviklere

Personlig side - evt. med foto. Korte informationer + link til hjemmeside.

Bug report

Er der stort behov for. A.la. EDB-afdelingens.

News

Hvad der er sket siden sidst. Det skal være nemt at tilføje nyheder.

Download

"Stable Releases".

Mailingliste

Er ved at blive lavet. Kun til intern brug på DIKU.

Der mangler altså Bugtracking, News og Download. Men det hele skal omstruktureres og gøres dynamisk/let at vedligeholde.

Om designet

Designet må gerne være meget enkelt. Det skal være let at ændre. Må ikke være statisk. Der skal være plads til eventuelle underpunkter. Må ikke være opdelt i frames. Må ikke tage for lang tid at load. Test hurtighed ved brug af database. Kan godt lide klare farver - "*siden skal smile*". Må ikke være småkedelig.

Om navigering

Topmenu er for besværlig. Der kommer nok underpunkter.

Om at have en database

Afhænger af behovet - hvordan udvikles siden i fremtiden? Ville være fint med administrationsmodul. Siden vil blive vedligeholdt af "*alle eller ingen*".

3.4 Andre udtalelser

Løbende har jeg fået mere at vide efter de to møder og de to interviews. Disse udtalelser er i kort version forsøgt samlet her.

1. maj 2001: Jyrki Katajainen, ansvarlig for Copenhagen STLs udvikling: Det vil være mest praktisk, hvis man kunne sende en mail med en nyhed, som automatisk vil blive udskrevet på nyhedssiden.
1. maj, 2001: Jyrki Katajainen, ansvarlig for Copenhagen STLs udvikling: De rapporter (design dokumenter), der skal ligge under mappen `cphst1/Reports/`, kommer til at hedde metodiske navne a.l.a. 2001-xx, som angiver hvornår og i hvilken rækkefølge rapporterne er skrevet. Rapporterne vil blive omdannet til `.pdf`-format, men findes på nuværende tidspunkt primært i `.ps`-format.
3. maj, 2001: EDB-afdelingen har efter egen udtalelse brugt RequestTracker [3] til deres bug report, men har foreslået at Bugzilla [2] måske ville være noget for Copenhagen STL.

3.5 Delkonklusion af behovsanalysen

Det er ikke lykkedes at holde mig til min interviewplan, og de to interviews er nærmest to helt forskellige samtaler, hvor vægten blev lagt på helt forskellige emner. Jeg synes dog ikke, at dette nødvendigvis er dårligt (selvom det ikke var efter planen), da jeg fik en del gode oplysninger fra begge (slags) interviews. I næste afsnit overvejes det egentlige design.

4 Designanalyse

I dette afsnit diskuteres designovervejelser på baggrund af behovsanalysen. Slutresultatet af designet kan ses i bilag B. Det færdige design ligger desuden på min lokale server (hjemmeside [4] og administrationsmodul [5]), hvor det er muligt at klikke sig rundt på siden.

Det er normalt at dele løsningen op i tre dele: data, logik og præsentation. Den information man har at arbejde med, er data-delen. Logik-delen er selve programmet eller scriptet, som behandler informationen, og præsentationsdelen er det der bliver vist i browseren (genereret af HTML) i sidste ende. I dette afsnit diskuteres primært lagring, strukturering og præsentation af data, men alle delene griber mere eller mindre ind i hinanden, så det har stort set ikke været muligt at diskutere data og præsentation uden at tage stilling til logikken, der binder disse to led sammen.

I afsnittene 4.1, 4.2, 4.3 og 4.4 behandles problemstillingen med lagring og strukturering af data. I afsnit 4.5 analyseres de overvejelser der er mht. valg af redskaber til udviklingen og videreudviklingen af Copenhagen STLs hjemmeside. I afsnit 4.6 behandles selve præsentationen af data, dvs. udseendet (eller det som de fleste webdesignere vil kalde for designet).

4.1 Database eller ej

Det har været til diskussion og overvejelse hvorvidt der skulle bruges en database til at holde styr på informationerne på siden eller ej. Her er en kort opsummering af fordele og ulemper ved brug af en database til Copenhagen STLs hjemmeside:

Fordele ved at have en database:

- Det vil være lettere at holde styr på bidragydere, nyheder, moduler mm.
- Det vil give muligheden for hurtigt at få forskellige informationer om det data der ligger i databasen. Fx kunne man få at vide hvor mange der arbejder på et bestemt modul, eller hvilke bidragydere, der er allerede har afsluttet deres modul(er), oa.
- Man slipper for at gemme samme oplysninger flere steder.
- Det vil gøre udviklingen af hjemmesiden mere konsistent, da hver data-del (entitet) får bestemte attributter, og disse attributter kan præsenteres på en entydig måde.

Ulemper ved at have en database:

- Alle informationerne ligger samlet i en database-fil, og hvis noget sker med denne fil, går al information tabt (medmindre man har en backup - men ændringer, der blev foretaget siden sidste backup vil stadig gå tabt).

- Det kan gå ud over load tiden af hjemmesiden, da information skal hentes fra databasen, hver gang en side loades.

Da jeg mener, at fordelene opvejer ulemperne, har jeg valgt at lægge diverse data ind i en database. Database-designet er beskrevet i afsnit 4.4 og overvejelser for valg af Database Management System er beskrevet i afsnit 4.5.2.

4.2 Struktur

Noget af det første der skal klarlægges, er hvordan strukturen af siden kommer til at se ud. Fra mit møde med Jyrki Katajainen (se afsnit 3.3.6), har jeg fået en ide om, hvilke punkter der er vigtige, og hvordan siden generelt skal omstruktureres.

Den struktur vi løbende er kommet frem til, ser således ud:

- Main Page
- News
- Download
- Source Code
- Design Documents
- Contributors
- New Contributors
- Related Links
- Literature
- Bug Reports
- Mailing List
- Contact Us

Hvad disse punkter dækker over, vil blive forklaret i de følgende underafsnit.

Copyright af siden vil blive vist nederst på hver side. Eventuelt skal der nederst på siden også stå angivet, hvornår denne side sidst er opdateret.

4.2.1 Main Page

Hovedsiden skal indeholde stort det samme, som den indeholder på nuværende tidspunkt, eventuelt skal det gøres lidt kortere. Jeg havde en idé om, at denne forside også skulle indeholde en kort forklaring af hvad de forskellige hovedpunkter dækker over, men jeg er ikke sikker på, om det vil kunne gøres tilstrækkeligt kort, så måske er det bedre at lade hovedpunkterne i menuen tale for sig selv.

4.2.2 News

Nyhedssiden skal vise de nyeste nyheder, der har med Copenhagen STL at gøre. Det kan være nye releases, det kan være en ny udvikler eller det kan være annoncering af nyt på siden. Nyheder, som bliver forældet, skal kunne arkiveres (gerne automatisk) og fra nyhedssiden skal der være adgang til at læse i arkivet.

Det skal være nemt at kunne tilføje nyheder, og der er flere måder det kunne gøres på.

En mulighed er, at man i administrationsmodulet skal kunne tilføje nyheder, hvorefter de automatisk dukker op på nyhedssiden. På samme måde skal man kunne ændre i de nyheder der allerede er lagt op (eventuelle stavfejl eller andet man gerne vil rette til) og man skal kunne slette en nyhed helt eller kunne arkivere den. På denne måde har man som administrator, fuld kontrol over hvilket indhold der er på nyhedssiden.

Det er også blevet foreslået (se afsnit 3.4), at man skal kunne sende en mail, hvorefter nyheden automatisk ville blive tilføjet på nyhedssiden. Her skulle kunne ses afsender af mailen, og en sådan nyhed vil typisk være en kort introduktion til nyheden med et link til hvor man kan læse resten af nyheden eller en "press release" (Et eksempel på dette kan bl.a. ses på PHPs officielle hjemmeside [8]).

Da valget af, hvilken metode man vil benytte sig af til at sætte nye nyheder op, afhænger af hvilket slags nyhed det drejer sig om, ville det være smart, hvis man havde mulighed for at vælge mellem de to metoder.

Nyhedesadministration vha. et administrationsmodul, er beskrevet i afsnit 4.3.1, og nyhedstilføjelse via. e-mail kan læses om i afsnit 4.3.2.

Noget der også skal overvejes, er hvor mange nyheder der max. skal være på siden før disse bliver arkiveret. Arkiveringen kan selvfølgelig gøres manuelt, men det nemmeste vil være, hvis en nyhed bliver arkiveret enten når den bliver forældet (tid) eller når der kommer for mange nyheder på siden (antal). Der er fordele og ulemper ved begge arkiverings-faktorer.

Fordele ved tid: Det er "naturligt" at en nyhed forældes. Bare fordi der er gået en måned eller lignende uden at der er kommet nye nyheder, betyder ikke at gamle nyheder stadig er nyheder.

Ulemper ved tid: Hvis ikke der kommer mange nyheder generelt, kan man risikere, at nyhedssektionen ofte står tom.

Fordele ved antal: Det er måske meget rart at se hvad der er sket sidst, selvom der er gået et stykke tid, siden det skete.

Ulemper ved antal: Vi kan risikere at der kommer rigtig mange nyheder på meget kort tid, og derfor kan man gå glip af en 2 dage gammel nyhed fordi der er kommet mange andre nyheder samtidig/siden. Eventuelt kunne man vælge at gøre antallet tilstrækkeligt stort, således at sandsynligheden for at der kommer for mange nyheder til at de kan stå på siden tilstrækkeligt længe bliver formindsket. Problemet er bare, at det ikke altid er lige heldigt for overskueligheden, at have en for lang liste af nyheder stående på nyhedssiden.

Om man vælger at arkivere nyheder efter et bestemt stykke tid eller efter et bestemt antal nyheder, ser altså ud til at afhænge af hvor ofte der kommer nyheder og hvor mange der kommer. Men det er ofte forskelligt hvor mange nyheder der kommer, og det er svært at gisne om.

Kombination af antal og tid

Problemet kunne løses ved at lave en kombination. Således kan nyhederne blive arkiveret hvis der er mere end et bestemt antal nyheder *eller* hvis nyhederne bliver forældet. Fx kan man lave en grænse på max 10 nyheder og max en 1 måned gamle nyheder. Men da det ikke er til at vide på forhånd hvor grænsen i antallet og tidslængden ligger, vil det være smart, hvis man nemt vha. administrationsmodulet kan ændre begge dele.

Da nogle af nyhederne kan være længere “press releases” eller lignende, kan det diskuteres, hvorvidt det er praktisk at have hele den lange nyhed (eller flere) stående på en nyhedssiden. En mulighed for at gøre det mere overskueligt, er at lade nyheden have en “appetitvækker”, som er en slags forord, som indbyder til at læse mere af nyheden, samt et link til resten af nyheden, a.la. [læs mere her](#). Dette link skal sende et nyhedsid videre med URL'en a.la. `news.php?nid=<nyhedsid>`, hvor der i `news.php` vil være en funktion som udskriver den ene nyhed med dette id, hvis der er angivet et id.

Hvis der derimod kommer en del kortere nyheder, er det en dårlig ide at dele nyheden op og lægge den på en side for sig. Dvs. ved forekomster af nyheder af forskellige længder, skulle administrator af nyheden have et valg mellem at lægge en nyhed op som en længere, opdelt nyhed, eller en kortere nyhed uden en ekstra side for sig.

Jeg har indtil videre valgt, at holde mig til at have alle nyhederne på samme side, men ovenstående er til evt. fremtidig overvejelse.

4.2.3 Downloads

Denne sektion vil ikke blive taget i brug før om et par måneder, eller hvor længe det nu vil tage at udvikle “stable releases”. På siden skal der være mulighed for at kunne downloade de nyeste stabile versioner af modulerne. Da der skal ligge så lidt arbejde i at vedligeholde hjemmesiden som muligt, skal der laves

et funktion, som på en eller anden måde kan udskrive de pågældende moduler automatisk, så de kan downloades. Der skal eventuelt også være en samlet tar-pakke, som kan downloades.

Hvis funktionen automatisk skal skanne og udskrive de færdige versioner, skal de udskille sig fra mængden på en eller anden måde. Fx kan de færdige versioner lægges i en bestemt mappe, eller de kan kaldes for noget som unikt kendetegner dem for stabile versioner. Man kan også udnytte CVSs mulighed for at sætte mærker eller "tag" på forskellige versioner af source-koden, og mærke de stabile versioner med et "stable"-mærke.

4.2.4 Source Code

Det nye punkt "Source Code" svarer til det gamle "Window to our CVS repository". Her ligger de Doxygen-generede programstykker, samt dokumentation (ikke rapporter, snarere en automatisk genereret struktur-analyse af C++-koden).

En ting, der gør denne afdeling forvirrende i den nuværende situation, er navigeringen. Der er en fast top-menu til de forskellige grupper af klasser, men denne er ikke konsistent med den menu der er på klasse-gruppens hovedside (som også hedder header-index). Det ville være naturligt, hvis de links der pegede på samme sted, også hed det samme. Eksempelvis er "modules" godt nok det samme som "components", men hvis ikke man holder sig til én betegnelse, vil den besøgende hurtig få den idé, at det drejer sig om to forskellige ting, eller måske to sider af samme sag, mens hyperlinket i virkeligheden peger samme sted hen.

Source-kode-analysen bliver genereret af et Doxygen-program, som er beskrevet nærmere i afsnit 4.5.3. Da det hele bliver automatisk genereret, er det ikke lige til at rette i strukturen. Headeren kan rettes til, så udseendet kommer til at passe til resten af designet. Alt andet skal rettes i doxygen-filer, og da jeg ikke har brugt Copenhagen STLs server til at designe hjemmesiden på (pga. problemer med rettigheder og installation af de programmer jeg gerne ville bruge), har jeg heller ikke fået sat mig ind i, hvad der skal rettes hvor.

En funktionalitet, der kunne være brugbar på "source code"-siden, var en søgefunktion. Hvad gør man hvis man vil finde et helt bestemt modul? Man blader sig igennem strukturen og hvis man har tilstrækkeligt med overblik, finder man det også. Men det ville lette ens arbejde, hvis man blot kunne søge på modulet. En sådan søgefunktion, kan downloades og installeres fra fx "ht://Dig" [9]. Deres søgemodul kan hurtigt gennemsøge alle tekster og html-filer på serveren. Det er gratis at hente, virker på de fleste UNIX-systemer og kræver en C og C++ oversætter.

4.2.5 Design Documents

Design dokumenterne, eller rapporterne, er generelle artikler/rapporter om Copenhagen STL og disse er generelt skrevet af studerende som også er udviklere af moduler. Bl.a. vil denne rapport om webdesign af Copenhagen STL også komme til at ligge under dette punkt.

Igen skal der være så lidt arbejde som muligt, i at vedligeholde denne side. Der skal derfor her være en funktion, som rekursivt kan skanne mappen, hvorunder alle rapporterne ligger, og udskrive dem ud på siden. Rapporterne, der ligger i "Report"-mappen, på Copenhagen STLs server, er i øjeblikket på postscript-format (.ps), men skal i sidste ende gerne være i .pdf-format (i følge afsnit 3.4). Smartest ville det være, hvis rapporterne kunne hentes i begge formater, så den besøgende selv kunne vælge. Hvilket format der er smartest, afhænger jo af hvilket operativsystem den besøgende har, og hvilke programmer hun/han har installeret på sin maskine.

Rapporterne kommer til at hedde metodiske navne (afsnit 3.4), som vil gøre det nemt at overskue hvornår rapporten er skrevet (i hvert fald i hvilken rækkefølge). Desuden skal hver rapport skannes for forfatternavn og titel, så det vil være nemt at overskue, hvem der har skrevet hvilken rapport og hvad rapporten indeholder.

Da hver rapport indeholder både titel og forfatter, kan disse fiskes frem fra de til postscript svarende .tex-filer, ved at matche på "regular expression" [10]. Det kræver dog rimelig meget entydighed af rapporterne. Forfatteren skrives ud, ved at hente indholdet af `author{...}`-feltet, og derfra filtreres alt andet end navnene ud. Dette kan dog være lidt problematisk, for hvordan genkendes udelukkende navne, hvis der i authorfeltet også står e-mail adresser og/eller andet? Evt. kunne alt undtagen `and-tags` skrives ud. Andre problemer, som indlejrede tags, kan også opstå. Fx er det ikke nok, at skrive alt det der kommer efter "`author{`" og før "`}`" ud, hvis der er indlejret et tag, som også slutter med et "`}`"

Titlen skrives ud ved at fiske indholdet af `title{}` ud. Dette kan gøres på samme som med forfatternavnene, men her vil sandsynligvis ikke være nær så mange problemer, da titlen (af egen erfaring) i de fleste tilfælde kun indeholder selve titlen på rapporten.

4.2.6 Contributors

Under "Contributors" vil man kunne se en liste af alle bidragyderne til udvikling af Copenhagen STL, samt se hvilke moduler de er ved at udvikle eller har udviklet.

Der findes i alt tre typer moduler - dem der endnu ikke er taget af nogen

udvikler, dem der er under udvikling, men endnu ikke godkendt og dem der allerede er udviklet og godkendt. De moduler der endnu ikke er taget, bliver behandlet i næste underafsnit (4.2.7). De to andre typer, er begge nogle der skal skrives ud under “Contributors” sammen med de udviklere der er i gang med at udvikle eller har udviklet disse moduler, og gerne med en angivelse af, hvorvidt moduler er godkendt eller ej.

Listen af både bidragydere og moduler skal udskrives dynamisk. Bedst ville det være hvis mappen med source-koden (“cphstl/”) blev skannet rekursivt, og de moduler der var lavet, ville blive skrevet ud. .cpp- og .h-filerne skulle hedde det samme som modulet, for at dette skulle kunne lade sig gøre, og der skulle ikke ligge andre .cpp- og .h-filer end selve modulerne. Da hver modul-source-kode indeholder (eller burde indeholde) dens forfatter(e), skulle selve source-filen også skannes (vha. regular expressions), og udviklerne udskrives.

En mindre besværlig, men tilgængelig også mindre dynamisk, måde at gøre det på, er at tilføje både udviklerne og modulerne efterhånden som der er brug for det, vha. et administrationsmodul. Dette ville kræve, at alle selv holdt styr på at både de selv og deres moduler var tilføjet.

Det har været oppe og vende, at hver udvikler evt. skulle have en egen personlige side, med korte oplysninger/fakta om sig selv, samt evt. et billede. Alle disse oplysninger kunne fx lægges i persontabellen (se afsnit 4.4.2), således at den personlige side kunne genereres automatisk for hver person med indtastede oplysninger.

4.2.7 New Contributors

Der skal været et punkt, hvor der søges efter nye udviklere, og “reklameres” for de moduler der endnu ikke er taget. De moduler der ikke er taget, burde også gerne udskrives dynamisk, men det vil ikke kunne lade sig gøre, at gøre det så dynamisk som foreslået i forrige underafsnit (4.2.6), da de moduler der endnu ikke er taget, ikke ligger nogen steder, som de kan skannes fra.

Hvis alle de moduler, der skal laves, er tilføjet gennem administrationsmodul i en database, kan modulerne nemt skrives ud ud fra denne.

4.2.8 Links og Literature

Disse punkter skal ikke ændres i forhold til det oprindelige. Under “Links” er en liste af hjælpsomme links til andre steder på nettet, hvor man kan finde informationer, som er relevante for udvikling af Copenhagen STL. De forskellige link er inddelt i grupper, hvilket gør det let at overskue udvalget.

“Literature” skal også bare indeholde den litteraturliste, som findes på siden i den nuværende situation.

4.2.9 Bug Reports

Der skal være mulighed for at reportere fejl til Copenhagen STLs bidragydere og/eller ansvarlige. Dette kan klares vha. et færdigkodet stykke software, a.la. RequestTracker eller Bugzilla. Læs mere om dette i afsnit 4.5.5.

4.2.10 Mailing List og Contact Us

Mailinglisten er kun til intern brug på DIKU. Her findes et link, hvor man kan få mere information om listen (linket virker dog kun fra DIKUs lokal netværk).

Under "Contact Us", skal man have mulighed for at komme i kontakt med en ansvarlig for Copenhagen STLs projekt.

4.3 Administration

I administrationsmodulet skal man nemt kunne tilføje, opdatere og fjerne elementer til/fra databasen. Administrationsmodulet skal kun være tilgængelig for udviklerne af Copenhagen STL. Modulet vil være beskyttet med password (det er det dog ikke i parksis lige nu, da det kun er en prototype, som alle har adgang til at teste). Indtil videre har jeg valgt kun at have en bruger, nemlig "administrator" til at kunne logge på sitet, og brugeren har rettigheder til alt på dette site. Senere, hvis dette blev nødvendigt, kunne man evt. både have en administrator, som havde adgang til alt, og brugere, som fx kun havde adgang til at rette ved sine egne oplysninger og ligende. Så langt er jeg dog slet ikke nået i dette projekt.

4.3.1 News

Fra administrationsside skal nyheder kunne tilføjes, rettes, slettes og arkiveres samt de-arkiveres.

Der må gerne benyttes HTML i nyhedsfeltet, og der skal ikke være nogen funktionalitet, der omdanner alt hvad der starter med "http://" til et link, eller indsætte
-tags ved linjeskift osv. - dette skal gøres manuelt vha. HTML, når man tilføjer en nyhed.

4.3.2 News via. mail

For at kunne tilføje en nyhed ved at sende en mail, skal der sættes en mailadresse op, som nyheden sendes til. Dette kunne fx være `news@cphts1.dk` eller lignende. Når man sender til denne adresse, skal mailserveren kunne sende mailen videre til et program, som vil sørge for at lægge nyheden ind i "news"-databasen. Hertil kunne fx tilføjes et "submitted by mail by ...", så man kan se hvem der har afsendt mailen. For at gøre dette, skal mailens "from"-sektion kopieres, og dette kan lægges i et evt. ekstra felt i news-tabellen i databasen, a.la. "submitted_by".

Noget som skal overvejes med denne form for nyhedsadministration, er sikkerheden. Det er ikke alle, som skal have adgang til at sætte en nyhed op bare ved at sende en mail til den rigtige mailadresse. Dette problem kan løses på flere måder:

1. Mailserveren tager kun imod mails fra bestemte e-mail adresser.
2. En nyheds-mail skal kunne godkendes, før den bliver lagt op nyhedssiden.

Hvis man benytter sig af første mulighed, kan det være relativt nemt at “snyde serveren”, ved at sende en mail hvor man vha. sit mailprogram angiver sin afsenderadresse, som en anden adresse end den egentlig er.

Mulighed to er relativt sikker. Den er dog mere besværlig, da man ikke altid kan være sikker på, at den der kan godkende nyheden, er til stede til at gøre det.

Da metoden med at godkende mailen er mere sikker end at serveren kun accepterer bestemte e-mail adresser, er det denne jeg gerne vil implementere. Da jeg dog ikke har adgang til Copenhagen STLs mail-server (og ikke har tid til at få det), vil jeg nøjes med at fortælle hvordan dette skal gøres.

Måden det ville fungere på, er at mailserveren, når den modtager en mail på nyheds-adressen, giver mailen videre (via. sendmail) til et program, der forwarder nyhedsmailen til den eller de personer som skal godkende mailen, evt. med et id. Nyheden kan allerede på dette tidspunkt lægges ind i nyhedstabellen (afsnit 4.4.1) af det program, som har modtaget mailen, og som kan indsætte i tabellen. Dog skal der til dette bruges et felt der hedder “approved” eller lignende, som indeholder et NULL hvis nyheden endnu ikke er godkendt og et 1-tal hvis nyheden er godkendt.

Modtageren skal da svare på denne mail, hvis han/hun gerne vil godkende den, og sørge for at id’et findes i emne-feltet af mailen (dvs. bare lade subjektet være som det er). Når mail-serveren får mailen med det id som hører til nyheden, sender den mailen videre til programmet, som sørger for at ændre “approved” i den record i nyheds-tabellen, som indeholder det id, som findes i subjektet, til 1.

Problemet med at en af de personer der er sat til at godkende mailen, er på ferie eller lignende, og derfor ikke lige til stede til at godkende en måske mere eller mindre vigtig nyhed, kan også løses. Fx. kan man i administrationsmodulet have mulighed for at ændre mail-adressen, som den mail der skal godkendes vil blive sendt til.

Et andet sikkerhedsmæssigt problem med at sende en mail til godkendelse, er at hvis nogen virkelig var opsat på at knække systemet og lægge en nyhed op mod alle andres vilje, så kan dette stadigvæk lade sig gøre med lidt snedighed og lidt held. Det eneste “gerningsmanden” skulle for at dette skulle kunne lade sig gøre, var at gætte id’et af den nyhedsmail han/hun lige har sendt afsted. For at komme udenom dette problem (hvis det altså er så stort et problem..

det er trods alt ikke nationalbankens sikkerhed vi taler om her!), kan den person, som skal godkende nyheden, bruge kryptering til at lave en "signature", så mailserveren ved at mailen er sendt fra den rigtige person, og ikke en tilfældig forbipasserende. Dette er jeg dog ikke sat så meget ind i, så jeg ville ikke komme nærmere på hvordan det bliver gjort.

4.3.3 Modules

Man skal kunne tilføje nye moduler i modul-tabellen, og disse kan både rettes og slettes igen.

4.3.4 Contributors

Man skal kunne tilføje nye brugere, slette dem igen og eventuelt rette. Man skal desuden kunne tilknytte moduler til udviklere. Kun moduler, der allerede eksisterer i modul-tabellen, kan tilføjes til udviklere. Disse moduler kan fjernes fra en udvikler igen, hvis dette ønskes.

4.3.5 Links

På administrationssiden, kan link tilføjes, kategoriseres, opdateres og slettes. Hvert link hører ind under netop en kategori. Kategorier kan tilføjes og slettes igen, hvis ingen links er tilknyttet kategorien.

4.4 Database

I denne sektion analyseres database-designet. Udskriften af den fil som generer tabellerne kan ses i bilag D.26.

4.4.1 News

En nyhed består af et id, en dato, selve teksten (nyheden) og et felt som angiver, om nyheden er arkiveret eller ej.

Id'et (*nid*), er et heltal, som bliver genereret automatisk, og bliver brugt til at gøre hver nyhed unik, samtidig med at det er nemmere at holde styr på rækkefølgen, som nyhedene bliver skrevet i.

Datoen er en streng, som indeholder en script-genereret dags dato for den dag nyheden bliver lagt i databasen på. Jeg har overvejet, at benytte typen "DATE" til formålet, men da jeg primært skal bruge datoen til at skrive ud på siden, og ikke til at sammenligne tidspunkter eller lignende, er det mere passende at have en streng, hvor man selv kan vælge på hvilken form datoen skal skrives.

Jeg har valgt at lade datoen være på formen "dd. måned, åååå", hvor "dd" angiver dagen med to cifre, "måned" angiver måneden på skriftform og åååå angiver året med fire cifre. Hvordan man angiver datoen, er udelukkende en

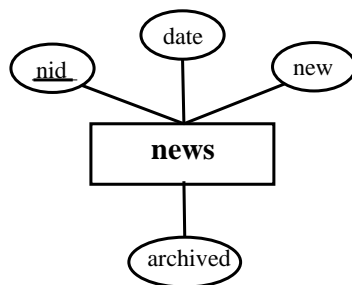


Figure 1: ER-diagram for tabellen “News”.

smagssag, og kan ændres i scriptet (hvis man ændrer det, skal man dog være opmærksom på, at de gamle nyheder, som har været indtastet forinden ændringen, stadig vil være på den gamle form).

Selve nyheden er på tekst-/htmlform og jeg har angivet typen TEXT, til at angive nyheden.

Derudover har tabellen et felt, der angiver om nyheden er aktiv eller arkiveret. “archived” er et heltal, men kun 0 og 1 bliver brugt. 0 angiver, at nyheden ikke er arkiveret, 1 angiver at den er det.

Der skal muligvis også være et “submitted_by”-felt, som angiver navnet på den person, som har lagt nyheden i databasen, samt et “approved”-felt, hvis news via. mail (afsnit 4.3.2) skal implementeres.

ER-diagrammet for nyheds-tabellen, kan ses på figur 1.

4.4.2 Contributor

En “contributor”, eller “person”, som jeg har valgt at kalde tabellen, består af en id, et navn, og et URL.

Id’et (pid) angiver personen unikt, og er genereret automatisk (vha. AUTO_INCREMENT). Personens navn er både fornavn og efternavn, eller hvordan man nu ønsker at angive bidragerne. Muligvis kunne man også dele navnet op i fornavn og efternavn hvis det fx skulle bruges til at sortere efter efternavn, eller hvis man skulle søge på kun det ene eller det andet. Da dette dog ikke er så vigtig en funktion på Copenhagen STLs side, har jeg valgt at lade bidragerne blive navngivet i et felt, som både skal indeholde fornavn, efternavn, og eventuelle kalde- og øgenavne.

Personens URL, er et link til en eventuel personlig hjemmeside. Dette er det link, som dukker op på personens navn. Hvis dette felt er tomt, eller kun inde-

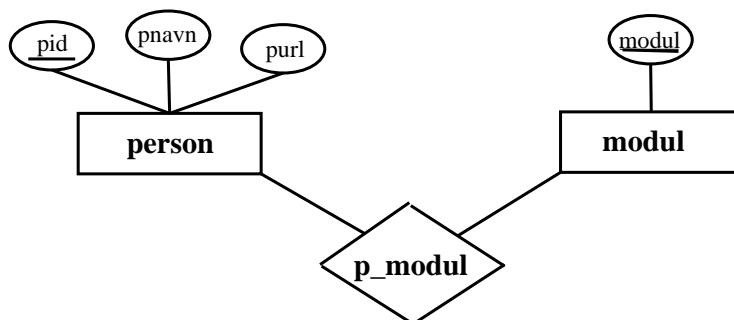


Figure 2: ER-diagram for tabellen “person” og “modul” samt relationen “p_modul”.

holder et “http://”, vil der ikke blive genereret noget link.

ER-diagrammet for person-tabellen ses på figur 2.

4.4.3 Modul og person/modul-relationen

Modul-entiteten består udelukkende af en modul-attribut. Jeg har overvejet at give modulet et id, men da moduler er unikt defineret ved sit eget navn, virker det ikke nødvendigt.

En modul kan tilknyttes flere udviklere, og en udvikler kan have (og har for det meste) flere moduler.

ER-diagrammet for person-modul-relationen ses på figur 2.

4.4.4 Links

Selvom de links der er på siden, egentlig ikke er så vigtige, har jeg valgt at lægge dem ind i en database, så de bliver lettere at administrere. I administrationsmodulet skal man altså kunne tilføje et link, og denne skal høre under en kategori (linkene er på nuværende tidspunkt også delt op i kategorier). Man kan enten tilføje et link under en af de nuværende kategorier, eller man kan tilføje en ny kategori først. Et link kan kun tilføjes under en kategori, som i forvejen eksisterer i “link_cat”-tabellen. Man kan flytte links til andre kategorier, ved at opdatere dem (bruge “update”-funktionen). Her kan man nemlig både rette i titel, URL samt tilhørende kategori.

Links kan både rettes og slettes. En kategori kan altid rettes, men kan kun slettes, hvis der ikke er nogen links der hører ind under denne.

Figur 3 viser ER-diagrammet for links og kategori.

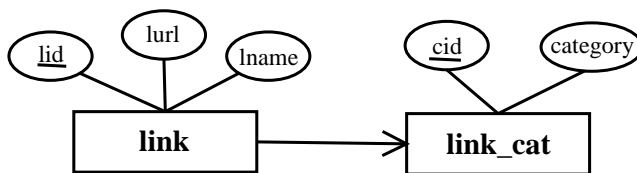


Figure 3: ER-diagram for tabellerne "links" og "link_cat".

4.5 Redskaber

I dette afsnit vil jeg beskrive de overvejelser, som jeg har foretaget mht. tekniske redskaber til design af Copenhagen STLs hjemmeside.

4.5.1 Scriptsprørog

Foruden HTML, er scripts uundværlige for en dynamisk hjemmeside. Det er vha. af scripts, at informationer kan hentes fra en database, og/eller behandles forskelligt, afhængigt af forskellige forhold og betingelser.

Hvilken script man vil benytte sig af, afhænger naturligvis af hvad man skal benytte det til, men er ofte lige så meget et smagsspørgsmål i sidste ende.

CGI-scripts: CGI står for Common Gateway Interface og er et interface mellem browseren og webserveren. CGI gør det altså muligt for browseren at kommunikere med serveren [11]. DIKUs servere tillader, så vidt jeg ved, ikke CGI scripts, men jeg er i tvivl om hvorvidt Copenhagen STLs hjemmesider kommer til at ligge på DIKUs servere eller ej.

Perl: Perl er et gratis sprog, som er velegnet til CGI. Det er designet med henblik på tekst-manipulation, hvilket også er det der oftest er brug for i web-scripting. Perl er dog mere end et web-scripting sprog, og er ikke specielt specialiseret til formålet.

PHP: PHP er specielt designet til web-scripting, hvilket betyder, at de funktionaliteter, som man ofte får brug for til webdesign af en dynamisk side, er blevet optimeret og kan ofte skrives med færre linjers kode, end ved brug af andre scripts [12]. PHP er gratis og minder i syntaksen om C++, hvilket gør at det er nemt at lære, hvis man er vant til at bruge C eller C++.

Til design af Copenhagen STL har jeg valgt at benytte mig af PHP. For det første fordi det er specielt designet til web-scripting og for det andet fordi jeg godt kan lide at det minder om C++, hvilket betyder at jeg ikke er helt på bar bund, når jeg skal til at lære sproget. Dette betyder ikke at jeg synes at PHP i sig selv er bedre en Perl. Mit valg er primært baseret på en personlig præference

4.5.2 Database

Da jeg først havde besluttet mig for at bruge PHP, virkede det for mig åbentlyst at bruge MySQL [13]. MySQL er et “let og hurtig” Database Management System (DBMS), og jeg ville ikke bruge noget, som kunne være for belastende for Copenhagen STLs server.

Jeg har først senere erfaret, at PostgreSQL [14] muligvis ville have været mindst lige så godt som MySQL. Sammenligner man disse to [15], tager det 2-3 gange længere tid for PostgreSQL, at generere en side i forhold til MySQL. Tilgængæld kan PostgreSQL klare ca. 3 gange så mange “connections” som MySQL.

Jeg har været godt tilfreds med MySQL og har ikke haft problemer med funktionaliteter, som den ikke kan understøtte. Da Copenhagen STLs side desuden ikke er en “mainstream” populær hjemmeside, som vil blive overbelastet af brugere, vil jeg mene at MySQL databasen er god nok til formålet.

4.5.3 Doxygen

Doxygen [7] bruges til at generere en omfattende dokumentation og struktur-analyse af C++programmer. Copenhagen STL bruger Doxygen dette formål i det afsnit som i nuværende situation hedder “Window to our CVS Repository”. Doxygen er et gratis program, som kan downloades og installeres på de fleste platforme. Da Copenhagen STL allerede er godt i gang med at bruge programmet, vil jeg ikke gå nærmere i detaljer om hvordan det virker, men bare konstatere, at det virker udmærket til formålet.

4.5.4 CVS

Det var egentlig meningen, at CVS skulle bruges til at holde styr på udviklingen af hjemmesiden. Det var sådan set ikke et krav, men det var noget der ville have været rart at have. Grunden til, at jeg ikke har brugt det, er at jeg brugte det meste af min projekt-periode til at være forvirret over hvordan CVS virker rent praktisk. Og nu, hvor jeg endelig er ved at få styr på det, er det for sent at benytte mig af dette redskab.

Copenhagen STL bruger CVS til at holde styr på udviklingen af deres STL. Men CVS bliver ikke brugt til webben, dvs. det bliver ikke brugt til at vise folk udefra, hvordan udviklingen foreløber, hvem der har rettet hvad og hvornår. Jeg blev nok en del forvirret over, at den Doxygen-genererede side, bliver kaldt for “Window to our CVS Repository”, når det nu slet ikke CVS-indholdet der bliver skrevet ud der. Jeg fandt aldrig helt ud af om Copenhagen STL gerne ville bruge CVS web [16] til at vise udviklingen eller ej, men som jeg også fik at vide til et af interviewene (afsnit 3.3.6), så ved Copenhagen STL endnu ikke helt selv, hvad der skal vises og hvad der skal være hemmeligt.

4.5.5 Bug Tracker

Der er behov for at have et program der kan bruges til at reportere fejl tilbage til Copenhagen STL. Der er før nævnt to forskellige programmer, som kan bruges til dette formål.

RequestTracker [3] tillader en bruger at sende spørgsmål og problemer vha. mail. Brugeren sender en mail, som bliver forwardet med et id, til de ansvarlige i køen (disse kunne fx være udviklerne af Copenhagen STL). En af disse kan vælge at tage sig af problemet, og derved få rettigheder til at besvare mailen.

Bugzilla [2] virker ved, at brugerne kan sende en bug gennem webben, hvorefter buggen lægges i en dertil designet database og sender den videre til den udvikler, som er ansvarlig for det komponent, hvor den reporterede bug findes. Bugzilla kan også bruges af udviklerne til at holde styr på de bugs der skal rettes samt prioritere, skedulere og planlægge.

Det brugerne af Copenhagen STL har brug for, er et system, som gør det muligt for en bruger af Copenhagen STLs produkter, at reportere bugs til den eller de bestemte personer, der er ansvarlige for det modul der fundet bug i (eller skal rettes spørgsmål til). Derfor virker Bugzilla som et oplagt valg af et bug tracking system. Bugzilla kan downloades gratis [17] - men kræver dog at Perl er installeret på serveren.

En sidste mulighed er også at bruge det gode gamle nntp i browseren. Dette vil give mulighed for at starte en tråd om et problem der skal løses, og via webben kan dette besvares af de ansvarlige. Jeg er dog gået væk fra denne mulighed, da Bugzilla virker som det bedste redskab til at dække Copenhagen STLs behov, og er gratis at hente.

4.6 Præsentation

Selvom der i interviewet med Lars Yde (afsnit 3.3.5), er blevet argumenteret primært for et design med topbjælke, vil dette i længden ikke være fleksibelt nok til Copenhagen STLs side, som det også er blevet argumenteret for i interviewet med Jyrki Katajainen (afsnit 3.3.6). Navigeringen sker vha. sidebjælken.

Der skulle også gerne være et eller andet til at angive hvor i menuen man befinder sig. Her har jeg en lavet en lille gif i for af en pil, der i menuen peger på det punkt, som man i øjeblikket befinder sig i.

Screenshots af det færdige design, kan ses i bilag B.

4.7 Delkonklusion af designanalysen

Det har været svært at gå i dybden og i bredden samtidigt, og for at få dækket alle aspekter af udviklingen af Copenhagen STLs hjemmeside, valgte jeg at gå mere i bredden end i dybden. Også derfor er nogle punkter i dette afsnit blevet gennemarbejdet grundigere end andre. Nyhedssiden har fået en del opmærksomhed i forhold til andre emner, mens "Source Code" og alt hvad der er Doxygen-relateret godt kunne trænge til en del mere analyse.

Jeg må konstatere, at jeg måske skulle have været bedre til at afgrænse, hvilke emner jeg ville lægge mest vægt på i dette projekt.

I næste afsnit beskrives implementationen af designet.

5 Implementation

Det meste af “programmet” er kommenteret, det virker derfor ikke nødvendig med lange forklaringer i dette afsnit.

Alle Copenhagens STLs sider består af en index-header og en index-footer. Index-headeren (`index-header.html`) indeholder hele menuen, samt sponsorlinks, således, at der kun skal rettes et sted, hver gang der skal tilføjes eller ændres noget i menuen. Det samme gælder for index-footeren (`index-footer.html`), som indeholder en linje med Copenhagen STLs copyright-erklæring. Index-headeren og -footeren indeholder også start- og slutstrukturen af den tabel, hvori indholdsteksterne ligger. Jeg har også overvejet at åbne og lukke for forbindelsen til MySQL i hhv. headeren og footeren, men da det ikke er alle sider, der benytter sig af denne forbindelse, føler jeg at det ville være spild af ressourcer. Derfor åbnes forbindelsen udelukkende når der er brug for det i de respektive filer.

I administrationssiden findes funktioner til at manipulere med data fra databasen. Disse funktioner er beskrevet i afsnit 4.3 og kommenteret i programmet. Programudskriften kan ses i bilag D.

6 Afprøvning

I dette afsnit testes korrekthed af HTML-koden samt korrektheden af de funktioner der er brugt, især i administrationsmodulet. Desuden testes der for hvor hurtigt siderne loader ved forskellige båndbredder.

6.1 Korrekthed: HTML

For at sikre at den HTML-kode, der bruges til at generere hjemmesiden, er i overensstemmelse med W3Cs [18] retningslinjer for korrekt HTML, kan W3s eget program til at efterprøve gyldighed af HTML, benyttes.

Jeg har testet samtlige af de sider der bliver genereret af min HTML-kode, og resultatet af afprøvningen kan ses i bilag C.1

Som det ses af bilag C.1, er der en enkelt "fejl", som går igen på alle ikke-admin siderne. Dette skyldes at jeg i headeren, som bliver inkluderet i alle filerne, bruger en HEIGHT-attribut i en tabel, til at angive at denne skal være 100% høj (fylde skærmen i længden). Jeg er udmærket klar over at dette ikke er helt korrekt, men jeg kan ikke finde en bedre måde at placere copyright-teksten nederst på siden (uden brug af frames) uafhængigt af browserens størrelse, og denne metode virker.

På administrations-siderne, er der også en enkelt fejl der går igen. Denne fejl er rent faktisk ikke en fejl i min HTML, men skyldes følgende forklaring, som er fundet på W3s sider:

"A URL for a CGI program that uses '&' as a separator, such as "http://host/prog?x=1&y=2". This is a common problem: the inventors of CGI didn't think things through very carefully when they decided to use the '&' character as a separator between CGI arguments, because '&' has special status in HTML."

Nu er det godt nok ikke CGI-script jeg har benyttet mig af, men jeg går ud fra at det tilsvarende gælder for andre scripts.

6.2 Korrekthed: scripts

I dette afsnit testes funktionaliteten af de scripts, der primært bruges i administrationsmodulet, til at manipulere med indholdet af databasen.

I bilag C.2 ses resultatet af tests af de funktioner der benyttes i administrationsmodulet. Det ses at de benyttede scripts virker efter hensigten.

6.3 Hastighed: load tid

Jeg har brugt NetMechanics Load Time Check [19] til test af Load Tid for forskellige båndbredder. NetMechanics Load Time Check siges at virke ved at

teste på antal og størrelse af billeder, filens størrelse, brug af højde- og breddeattributter i billeder og tabeller og antallet af servere, der skal kontaktes, men jeg har ikke noget egentlig dokumentation for helt præcist hvordan det virker. Da jeg dog ikke kan nå at udvikle eget test-program selv, må jeg bruge dette som en vejledende hjælp. Jeg har sammenlignet med forskellige tests, fundet på nettet, og load tiderne varierer en smule. Fx loader siderne generelt lidt hurtigere i følge Web Site Garage [20], men som sagt, alle tiderne er blot vejledende.

Generelt vil siderne sandsynligvis loade en del hurtigere end angivet, da Copenhagen STLs "logo", samt den lille pil, der angiver ens nuværende position i menuen, som går igen på alle siderne og påvirker load tiden i testen, vil ligge i cachen efter at en bruger har været inde på den første af siderne, og vil derfor ikke tage nogen nævneværdig tid at loade. Til sammenligning kan det nævnes, at selve tilføjelsen af den lille gif, som er 6x6 pixel stor, har i følge denne test forværret load tiden med op til 40 sekunder i værste tilfælde!

En udskrift af test for samtlige af de sider, som designet af Copenhagen STL indbefatter, kan ses i bilag C.3. I følge denne, er den langsomste load tid ved en båndbredde på 14.4K, på under 7 sekunder. Dette synes jeg er acceptabelt, taget i betragtning af at det er den værste mulige load tid og kun ved første besøg af den første side i Copenhagen STL.

7 Konklusion

Behovsanalysen er gået fint, men burde have været mere opdateret i løbet af hele projekt-perioden.

Jeg har brugt behovsanalysen som grundlag til designanalysen, og denne er gået OK, men der har været for mange aspekter at tage fat på, hvorfor jeg ikke har fået lavet en komplet dækkende designanalyse.

Jeg har fået implementeret en del af designet.

Webdesignet er på ingen måde et færdigt design, men snarere en prototype til det. Det er et grundlag for en revurderet version af behovsanalysen, som er et grundlag for udvidede designovervejelser og som igen er grundlag for et udvidet design.

Litteraturliste

- [SYS] Hasse Clausen, *“Informationsteknologiens menneskelige grundlag”*, 2000, Teknisk Forlag.
- [1] Copenhagen STLs nuværende hjemmeside:
<http://www.cphstl.dk>
- [2] Bugzilla:
<http://www.mozilla.org/bugs/>
- [3] Request Tracker:
<http://www.fsck.com/projects/rt/>
- [4] Det færdige design, på min lokale server:
<http://abel.erichsen.com/cphstl/ny/index.php>
- [5] Administrationsmodul til design, på min lokale server:
<http://abel.erichsen.com/cphstl/ny/admin/index.php>
- [6] MU web design: *“Interview Notes”*:
http://studsys.mscs.mu.edu/~{}georgec/Classes/Web.1996/Notes/interview_notes.html
- [7] Doxygen:
<http://www.stack.nl/~{}dimitri/doxygen/>
- [8] PHPs officielle hjemmeside:
<http://www.php.net>
- [9] ht://Dig: *“Internet search engine software”*:
<http://htdig.org>
- [10] PHPBuilder.com: *“Learning to Use Regular Expressions by Example”*:
<http://www.phpbuilder.com/columns/dario19990616.php3>
- [11] Knowledge Base: *“What is CGI?”*:
<http://developer.irt.org/script/5600.htm>
- [12] About.com: *PHP vs. Perl/CGI*:
<http://perl.about.com/compute/perl/library/weekly/aa040500a.htm>
- [13] MySQLs officielle hjemmeside:
<http://www.mysql.com>
- [14] PostgreSQLs officielle hjemmeside:
<http://www.se.postgresql.org/>
- [15] PHPBuilder.com: *“MySQL and PostgreSQL Compared”*:
<http://www.phpbuilder.com/columns/tim20000705.php3>

- [16] Et eksempel på CVS Repository på webben:
<http://cvsweb.horde.org/cvs.php>
- [17] Download Bugzilla:
<http://www.mozilla.org/projects/bugzilla/>
- [18] World Wide Web Consortium:
<http://www.w3.org>
- [19] NetMechanic: "*Load Time Check*":
http://www.aota.net/HTML_Validation_Tools/loadcheck.php3
- [20] Web Site Garage: "*Improve your site*":
<http://websitegarage.com>

A Bilag - De tre designeksempler



Figure 4: Eksempel 1

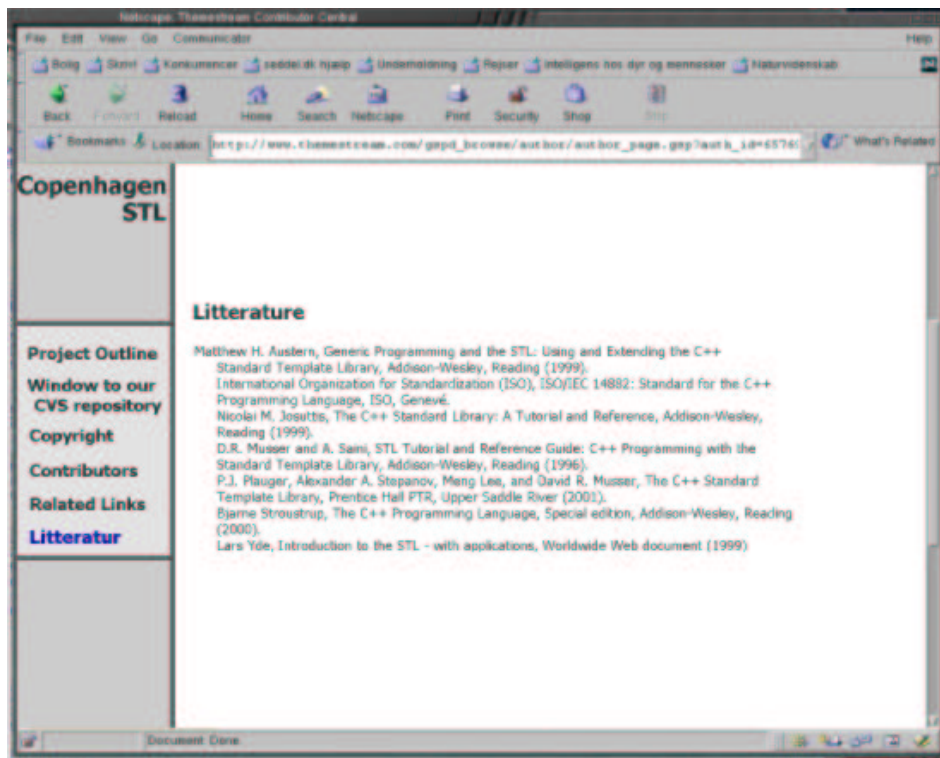


Figure 5: Eksempel 2

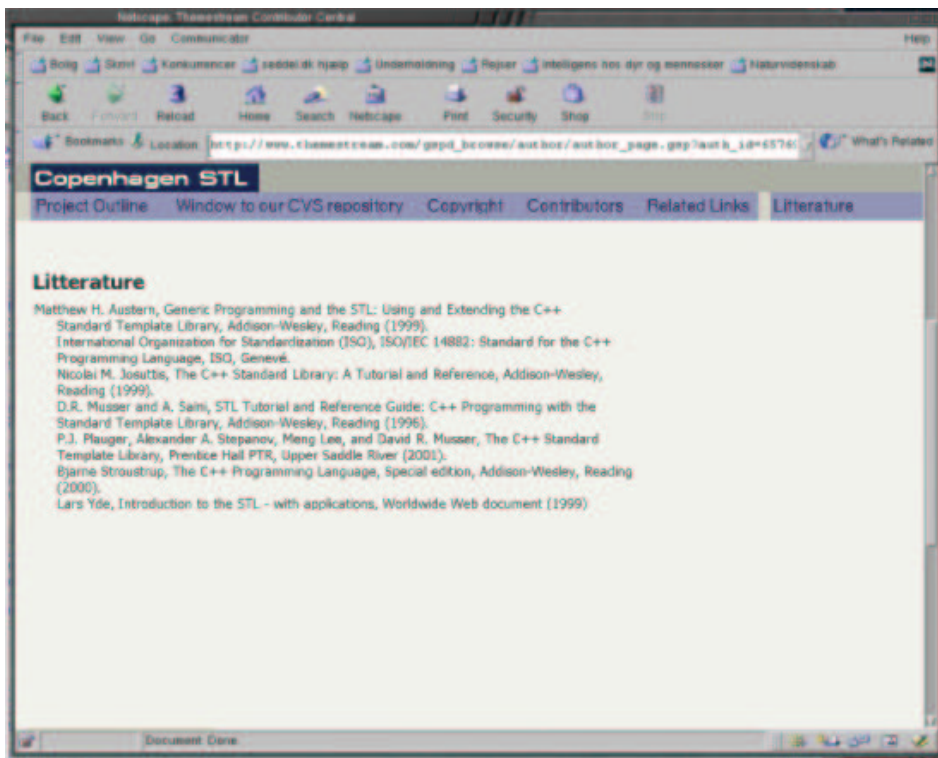


Figure 6: Eksempel 3

B Bilag - Det færdige Design

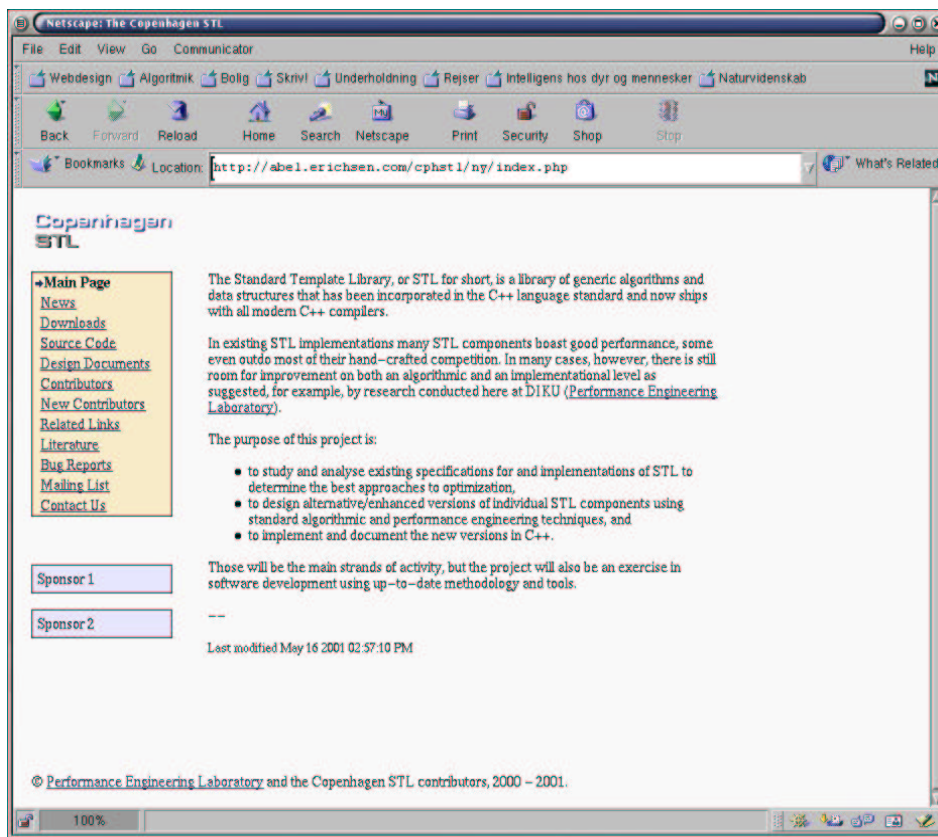


Figure 7: Design - screenshot af forsiden

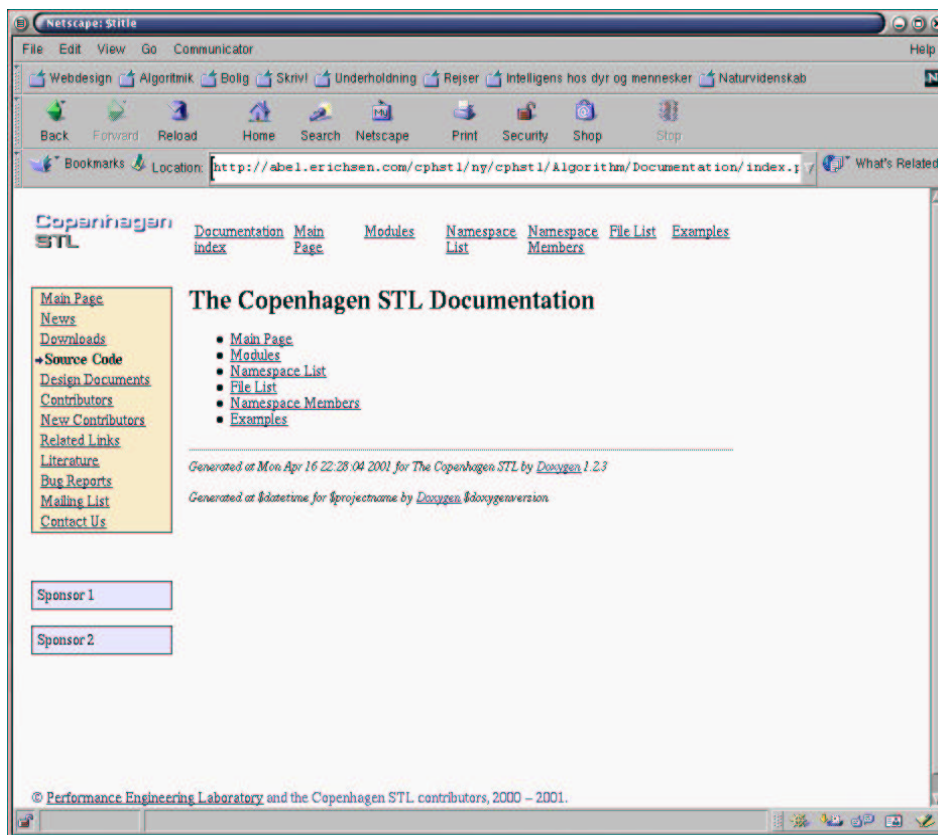


Figure 8: Design - Screenshot af hvordan indholdet af den Doxygen-genererede side skulle se ud

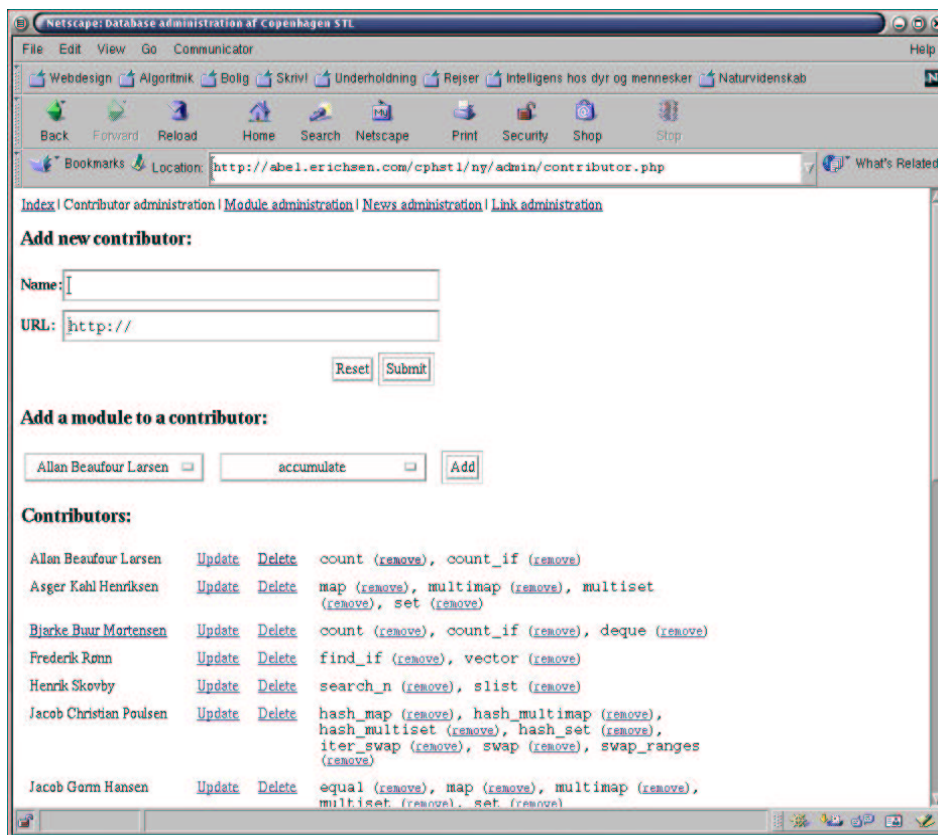


Figure 9: Administratormodul - screenshot af administration af bidragydere

C Bilag - Afprøvning

C.1 Test af korrekthed: HTML

Input i http://validator.w3.org/	Resultat
http://abel.erichsen.com/cphstl/ny/index.php	Fejl 1*
http://abel.erichsen.com/cphstl/ny/news.php	Fejl 1*
http://abel.erichsen.com/cphstl/ny/downloads.php	Fejl 1*
http://abel.erichsen.com/cphstl/ny/source.php	Fejl 1*
http://abel.erichsen.com/cphstl/ny/docs.php	Fejl 1*
http://abel.erichsen.com/cphstl/ny/contributors.php	Fejl 1*
http://abel.erichsen.com/cphstl/ny/newcontrib.php	Fejl 1*
http://abel.erichsen.com/cphstl/ny/links.php	Fejl 1*
http://abel.erichsen.com/cphstl/ny/literature.php	Fejl 1*
http://abel.erichsen.com/cphstl/ny/bugs.php	Fejl 1*
http://abel.erichsen.com/cphstl/ny/ mailing.php	Fejl 1*
http://abel.erichsen.com/cphstl/ny/contact.php	Fejl 1*
http://abel.erichsen.com/cphstl/ny/admin/index.php	Ingen fejl
http://abel.erichsen.com/cphstl/ny/admin/contributor.php	Fejl 2**
http://abel.erichsen.com/cphstl/ny/admin/module.php	Fejl 2**
http://abel.erichsen.com/cphstl/ny/admin/news.php	Fejl 2**
http://abel.erichsen.com/cphstl/ny/admin/links.php	Fejl 2**

* Fejl 1:

Line 10, column 74:

```
... ELLPADDING="5" CELLSPACING="5" WIDTH="700" HEIGHT="100%">
```

Error: there is no attribute "HEIGHT" for this element (in this HTML version)

** Fejl 2 a.l.a. denne her: :

Line 64, column 228

```
... N="left"><A HREF="contributor.php?action=delete_p&pid=10">Delete</A></...
...
```

Error: unknown entity "pid"

C.2 Test af korrekthed: scripts

Fil under admin/	Funktionalitet	Virker?
contributor.php	Tilføj ny udvikler	Ja
contributor.php	Tilføj modul til udvikler	Ja
contributor.php	Opdater udvikler	Ja
contributor.php	Slet udvikler uden tilknyttede moduler	Ja
contributor.php	Slet udvikler med tilknyttede moduler	Næsten*
contributor.php	Fjern modul fra udvikler	Ja
module.php	Tilføj modul	Ja
module.php	Opdater modul	Ja
module.php	Slet modul	Næsten*
news.php	Tilføj nyhed	Ja
news.php	Opdater nyhed	Ja
news.php	Slet nyhed	Ja
news.php	Arkiver/"de-arkiver" nyhed	Ja
links.php	Tilføj link	Ja
links.php	Opdater link	Ja
links.php	Slet link	Ja
links.php	Tilføj kategori	Ja
links.php	Opdater kategori	Ja
links.php	Slet tom kategori	Ja

* Af en eller anden grund, kan jeg ikke få ON DELETE CASCADE til at virke, og derfor slettes rækken i relationen mellem person og modul ikke, hvis jeg fjerner enten det pågældende modul eller den pågældende personen fra databasen.

C.3 Test af hastighed - load tid

Load tid målt i sekunder					
Fil	14.4K	28.8K	56K	ISDN 128K	T1 1.44Mbps
index.php	5.95	3.97	3.00	2.44	2.04
news.php	6.70	4.35	3.20	2.53	2.05
downloads.php	5.21	3.60	2.82	2.36	2.03
source.php	6.35	4.17	3.11	2.49	2.04
docs.php	5.47	3.73	2.88	2.39	2.03
contributors.php	6.30	4.15	3.09	2.48	2.04
newcontrib.php	5.79	3.89	2.96	2.43	2.04
links.php	6.11	4.06	3.05	2.45	2.04
literature.php	5.82	3.91	2.97	2.43	2.04
bugs.php	5.19	3.59	2.81	2.36	2.03
mailing.php	5.37	3.68	2.86	2.38	2.03
contact.php	5.37	3.68	2.86	2.38	2.03


```
</TABLE>  
  
</TD>  
<TD WIDTH="540" HEIGHT="85%" VALIGN="top">  
  
<!-- index-header end -->
```

D.2 index.php

```
<!-- index.php -->

<?php
include("index-header.html");
?>

<P>
The Standard Template Library, or STL for short, is a library of
generic algorithms and data structures that has been
incorporated in the C++ language standard and now
ships with all modern C++ compilers.
</P>

<P>
In existing STL implementations many STL components boast good performance, some even outdo most of their
hand-crafted competition. In many cases, however, there is still room for improvement on both an algorithmic
and an implementational level as suggested, for example, by research conducted here at DIKU
(<a href="http://www.diku.dk/research-groups/performance-engineering">Performance Engineering Laboratory</a>).
</P>

<P>
The purpose of this project is:
</P>
<UL>
<LI>to study and analyse existing specifications for and implementations of STL to determine the
best approaches to optimization,
</LI>
<LI>to design alternative/enhanced versions of individual STL components using standard algorithmic
and performance engineering techniques, and
</LI>
<LI>
to implement and document the new versions in C++.
</LI>
</UL>
<P>
Those will be the main strands of activity, but the project will also be an
exercise in software development using up-to-date methodology and tools.
</P>

<P>--</P>
<?php
$filemod = filemtime(__FILE__);
$filemtime = date("F j Y h:i:s A", $filemod);
echo ("<FONT SIZE=\2\>Last modified $filemtime</FONT>");
?>

<?php
include("index-footer.html");
?>
```

D.3 index-footer.php

```
<!-- index-footer.html -->

    </TD>
  </TR>
<TR>
  <TD WIDTH="700" HEIGHT="20" VALIGN="bottom" COLSPAN="2">
    &copy; <A
      HREF="http://www.diku.dk/research-groups/performance-engineering">Performance
      Engineering Laboratory</A> and the Copenhagen STL
      contributors, 2000 - 2001.
    </TD>
</TR>
</TABLE>

</BODY>
</HTML>
```

D.4 news.php

```
<!-- news-php -->

<?php
include("index-header.html");
?>

<?php
$db = mysql_connect("localhost", "lenka", "sedelikaki")
or die("Could not connect");

mysql_select_db(cphst1, $db);

?>

<H3>News</H3>

<TABLE CELLPADDING="2" CELLSPACING="2" BORDER="0" WIDTH="500">
<?php
    $max_days = "14";
    $max_number = "10";
    $news = mysql_query("SELECT nid, date, new, archived FROM news ORDER BY nid DESC")
or die("Bad query: ".mysql_error());

    while ($new_arr = mysql_fetch_array($news)) {
        extract($new_arr);
        if ($archived == 0 && $max_number != 1) {
            $max_number = $max_number-1;
            echo "<TR><TD VALIGN="top" WIDTH="110" NOWRAP>
<B>$date:</B></TD><TD VALIGN="top" WIDTH="390"> $new</TD></TR>";
        }
    }
?>
</TABLE>

<DIV ALIGN="right">
<A HREF="archived_news.php">Archived news</A>
</DIV>

<?php
mysql_close ($db);
?>

<P>--</P>
<?php
    $filemod = filetime(__FILE__);
    $filemtime = date("F j Y h:i:s A", $filemod);
    echo ("<FONT SIZE="2">Last modified $filemtime</FONT>");
?>

<?php
include("index-footer.html");
?>
```

D.5 archived_news.php

```
<!-- archived_news.php -->

<?php
include("index-header.html");
?>

<?php
$db = mysql_connect("localhost", "lenka", "sedelikaki")
or die("Could not connect");

mysql_select_db(cphst1, $db);

?>

<P>
<H3>Archived News</H3>
<P>

<TABLE CELLPADDING="2" CELLSPACING="2" BORDER="0" WIDTH="500">
  <?php
    $news = mysql_query("SELECT nid, date, new, archived FROM news ORDER BY nid DESC")
or die("Bad query: ".mysql_error());

    while ($new_arr = mysql_fetch_array($news)) {
      extract($new_arr);
      if ($archived == 1) {
        echo "<TR><TD VALIGN=\"top\" WIDTH=\"110\" NOWRAP>
          <B>$date:</B></TD><TD VALIGN=\"top\" WIDTH=\"390\"> $new</TD></TR>";
      }
    }
  ?>
</TABLE>
</P>

<P>
  <DIV ALIGN="right">
    <A HREF="news.php">Back to news</A>
  </DIV>
</P>

<?php
mysql_close ($db);
?>

<?php
include("index-footer.html");
?>
```

D.6 *downloads.php*

```
<!-- downloads.php -->

<?php
include("index-header.html");
?>

    <H3>Downloads</H3>
    Download latest versions.

    <P>--</P>
    <?php
        $filemod = filetime(__FILE__);
        $filemtime = date("F j Y h:i:s A", $filemod);
        echo ("<FONT SIZE=*2*>Last modified $filemtime</FONT*>");
    ?>

<?php
include("index-footer.html");
?>
```

D.7 source.php

```
<!-- source.php -->

<?php
include("index-header.html");
?>

<H3>Documentation and source code index page</H3>
<I>(Dette afsnit er endnu ikke indopereret i dette design, men under "algorithm" er der et eksempel på
hvordan det skulle se ud)</I>
<H4>ISO/ANSO C++ compliant section</H4>
<UL>
  <!-- burde udskrives automatisk ved at skanne alle dokumentationsmapper for index.php filer-->
  <LI><FONT SIZE="3"><A href="cphst1/Algorithm/Documentation/index.php">algorithm</A></FONT></LI>
  <LI><FONT SIZE="3"><A href="cphst1/Vector/Documentation/index.html">vector</A></FONT></LI>
  <LI><FONT SIZE="3"><!--<A href="cphst1/Bitset/Documentation/index.html">bitset</A>--></FONT></LI>
  <LI><FONT SIZE="3"><A href="cphst1/Iterator/Documentation/index.html">iterator</A></FONT></LI>
  <LI><FONT SIZE="3"><A href="cphst1/Deque/Documentation/index.html">deque</A></FONT></LI>
  <LI><FONT SIZE="3"><A href="cphst1/Functional/Documentation/index.html">functional</A></FONT></LI>
  <LI><FONT SIZE="3"><!--<A href="cphst1/Map/Documentation/index.html">map</A>--></FONT></LI>
  <LI><FONT SIZE="3"><A href="cphst1/Memory/Documentation/index.html">memory</A></FONT></LI>
  <LI><FONT SIZE="3"><!--<A href="cphst1/Numeric/Documentation/index.html">numeric</A>--></FONT></LI>
  <LI><FONT SIZE="3"><!--<A href="cphst1/Queue/Documentation/index.html">queue</A>--></FONT></LI>
  <LI><FONT SIZE="3"><!--<A href="cphst1/Set/Documentation/index.html">set</A>--></FONT></LI>
  <LI><FONT SIZE="3"><!--<A href="cphst1/Stack/Documentation/index.html">stack</A>--></FONT></LI>
</FONT>
</UL>
<H4>Copenhagen STL specific extensions</H4>
<UL>
  <LI><FONT SIZE="3"><!--<A href="cphst1/List/Documentation/index.html">list</A>--></FONT></LI>
  <LI><FONT SIZE="3"><A href="cphst1/HashMap/Documentation/index.html">hashmap</A></FONT></LI>
  <LI><FONT SIZE="3"><!--<A href="cphst1/HashSet/Documentation/index.html">hashset</A>--></FONT></LI>
  <LI><FONT SIZE="3"><A href="cphst1/Optimization/Documentation/index.html">optimization</A></FONT></LI>
</UL>

<P--</P>
<?php
$filemod = filetime(__FILE__);
$filemtime = date("F j Y h:i:s A", $filemod);
echo ("<FONT SIZE="2">Last modified $filemtime</FONT>");
?>

<?php
include("index-footer.html");
?>
```

D.8 docs.php

```
<!-- docs.php -->

<?php
include("index-header.html");
?>

<H3>Design Documents</H3>
General documents about Copenhagen STL.

<?php
// (rapporterne har navngivet efter hvornår de er skrevet)

// scan Report dir and return all *.ps files
// this function is from http://www.php.net
// TO-DO: *.pdf files need to be returned as well
function list_ps ($dirname, $recursive = 1) {

    // try to figure out what delimiter is that's already in use...
    $delim = (strstr($dirname, "/") ? "/" : "\\");

    if($dirname[strlen($dirname)-1]==$delim)
        $dirname.=$delim;

    $handle = opendir($dirname);

    while (($file = readdir($handle)) {

        if ($file=='.' || $file=='..') {
            continue;
        }

        if(is_dir($dirname.$file) && $recursive) {
            $x = list_ps($dirname.$file.$delim);
            $result_array = array_merge($result_array, $x);
        } else {
            if (ereg ("\ps$", $file)) {
                $result_array[]=$dirname.$file;
            }
        }
    }

    closedir($handle);

    if (sizeof($result_array)) {
        natsort($result_array);
    }

    return $result_array;
}

$ps_array = list_ps ("cphstl/Report/", 1);
```

```
echo "<TABLE CELLPADDING=\"3\" CELLSPACING=\"3\" BORDER=\"0\" WIDTH=\"500\">";
for ($i = 0; $i < sizeof($ps_array); $i++) {
    $ps_file = basename($ps_array[$i]);

    echo "<TR><TD BGCOLOR=\"#EFEFFF\" VALIGN=\"top\" WIDTH=\"100\">";
    echo "<A HREF=\"$ps_array[$i]\">$ps_file</A>";
    echo "</TD><TD BGCOLOR=\"#EFEFFF\" VALIGN=\"top\" WIDTH=\"400\">";
    echo "Title, author";
    echo "</TD></TR>";
}

echo "</TABLE>";

?>

<P>--</P>
<?php
    $filemod = filetime(__FILE__);
    $filemtime = date("F j Y h:i:s A", $filemod);
    echo ("<FONT SIZE=\"2\">Last modified $filemtime</FONT>");
?>

<?php
    include("index-footer.html");
?>
```

D.9 contributors.php

```
<!-- contributors.php -->

<?php
include("index-header.html");
?>

<?php
$db = mysql_connect("localhost", "lenka", "sedelikaki")
    or die("Could not connect");

mysql_select_db(cphstl, $db);

?>

<H3>Contributors</H3>

<UL>
<?php
$people = mysql_query("SELECT pid, pname, purl FROM person ORDER BY pname")
    or die("Bad query: ".mysql_error());
while ($person_arr = mysql_fetch_array($people)) {
    extract($person_arr);
    if ($purl AND $purl!="http://") {
        echo "<LI><A HREF=\"\$purl\"> \$pname</A>";
    } else {
        echo "<LI> \$pname";
    }

    echo ": &nbsp;<TT>";
    $modul = "";
    $modul_str = "";
    $has_modul = mysql_query("SELECT modul FROM p_modul WHERE pid='\$pid'")
        or die("Bad query: ".mysql_error());
    if (mysql_affected_rows() != 0) {
        if (mysql_affected_rows() == 1) {
            extract(mysql_fetch_array($has_modul));
            echo " \$modul";
        } else {
            while ($modul_arr = mysql_fetch_array($has_modul)) {
                extract($modul_arr);
                $modul_str = $modul_str . " . " . $modul;
            }
        }
    }
    echo substr ($modul_str, 2);
    echo "</TT><BR>";
}

// scan cphstl dirs and write all *.cpp files into an array
// function list_mod ($dirname, $recursive = 1) {
//
```

```
// try to figure out what delimiter is that's already in use...
// $delim = (strstr($dirname, "/") ? "/" : "\\");
//
// if($dirname[strlen($dirname)-1]==$delim)
// $dirname.=$delim;
//
// $handle = opendir($dirname);
//
// while (($file = readdir($handle))) {
//     if ($file=='.'||$file=='..') {
//         continue;
//     }
//
//     if(is_dir($dirname.$file) && $recursive) {
//         $x = list_mod($dirname.$file.$delim);
//         $result_array = array_merge($result_array, $x);
//     } else {
//         if (ereg ("\.cpp$", $file)) {
//             $result_array[]=$dirname.$file;
//         }
//     }
// }
//
// closedir($handle);
//
// if (sizeof($result_array)) {
//     natsort($result_array);
// }
//
// return $result_array;
//
// $mod_array = list_mod ("cphstl/", 1);
//
// // print all modules
// // ATM this functions prints every .cpp-file in the cphstl-dir
// for ($i = 0; $i < sizeof($mod_array); $i++) {
//     $cpp_file = basename($mod_array[$i]);
//     $module = substr ($cpp_file, 0, -4);
//
//     echo "$module, ";
// }
//
// ?>
// </UL>
//
// <?php
//     mysql_close ($db);
```

```
?>

<P>--</P>
<?php
    $filemod = filetype(__FILE__);
    $filemtime = date("F j Y h:i:s A", $filemod);
    echo ("<FONT SIZE=2>Last modified $filemtime</FONT>");
?>

<?php
    include("index-footer.html");
?>
```

D.10 newcontrib.php

```
<!-- newcontrib.psp -->

<?php
include("index-header.html");
?>

    <?php
        $db = mysql_connect("localhost", "lenka", "sedelikaki")
            or die("Could not connect");

        mysql_select_db(cphst1, $db);

    ?>

    <H3>New Contributors</H3>
    <P>
    <B>Components not yet taken</B>
    </P>

    <?php
        $db = mysql_connect("localhost", "lenka", "sedelikaki")
            or die("Could not connect");

        mysql_select_db(cphst1, $db);

        // print not yet taken modules from database
        $free_modules = mysql_query("SELECT modul.modul FROM modul
            LEFT JOIN p_modul ON modul.modul=p_modul.modul WHERE p_modul.modul IS NULL")
            or die("Bad query: ".mysql_error());

        $modul_str = " ";
        while ($module_arr = mysql_fetch_array($free_modules)) {
            extract($module_arr);
            $modul_str = $modul_str . " " . $modul;
        }
        echo substr (" $modul_str", " ", 2);

    ?>

    <?php
        mysql_close ($db);

    ?>

    <P>--</P>
    <?php
        $filemod = filetype(__FILE__);
        $filemtime = date("F j Y h:i:s A", $filemod);
        echo ("<FONT SIZE=\2\>Last modified $filemtime</FONT>");

    ?>

<?php
include("index-footer.html");
?>
```

D.11 links.php

```
<!-- links.php -->

<?php
include("index-header.html");
?>

    <?php
        $db = mysql_connect("localhost", "lenka", "sedelikaki")
            or die("Could not connect");

        mysql_select_db(cphst1, $db);

    ?>
    <H3>Related Links</H3>
    <DL>
    <?php
        $categories = mysql_query("SELECT cid, category FROM link_cat")
            or die("Bad query: ".mysql_error());

        while ($cat_arr = mysql_fetch_array($categories)) {
            extract($cat_arr, EXTR_PREFIX_ALL, "cat");
            echo "<DT><B>$cat_category</B></DT>";

            $links = mysql_query("SELECT lid, lname, lur1, cid FROM links WHERE cid ='$cat_cid'")
                or die("Bad query: ".mysql_error());

            while ($link_arr = mysql_fetch_array($links)) {
                extract($link_arr, EXTR_PREFIX_ALL, "link");
                echo "<DD><A HREF=\"$link_lur1\">$link_lname</A></DD>";

            }
        }
    ?>
    </DL>

    <P>--</P>
    <?php
        $filemod = filemtime(__FILE__);
        $filemtime = date("F j Y h:i:s A", $filemod);
        echo ("<FONT SIZE=2>Last modified $filemtime</FONT>");
    ?>

<?php
include("index-footer.html");
?>
```

D.12 literature.php

```
<!-- literature.php -->

<?php
include("index-header.html");
?>

<H3>Literature</H3>
<UL>
<LI>Matthew H. Austern, <I>Generic Programming and the
STL: Using and Extending the C++ Standard Template Library,</I>
Addison-Wesley, Reading (1999). </LI>

<LI>International Organization for Standardization
(ISO), <I>ISO/IEC 14882: Standard for the C++ Programming
Language,</I> ISO, Geneva. </LI>

<LI>Nicolai M. Josuttis, <I>The C++ Standard Library:
A Tutorial and Reference,</I> Addison-Wesley, Reading (1999). </LI>

<LI>D.R. Musser and A. Saini, <I>STL Tutorial and
Reference Guide: C++ Programming with the Standard Template
Library,</I> Addison-Wesley, Reading (1996). </LI>

<LI>P.J. Plauger, Alexander A. Stepanov, Meng Lee, and
David R. Musser, <I>The C++ Standard Template Library,</I> Prentice
Hall PTR, Upper Saddle River (2001). </LI>
<LI>Bjarne Stroustrup, <I>The C++ Programming
Language,</I> Special edition, Addison-Wesley, Reading (2000). </LI>

<LI>Lars Yde, <I><A
HREF="http://home.worldonline.dk/~lars_yde/stl/Index.html">Introduction
to the STL - with applications</A></I>, Worldwide Web document
(1999)</LI>
</UL>

<P>--</P>
<?php
$filemod = filemtime(__FILE__);
$filemtime = date("F j Y h:i:s A", $filemod);
echo ("<FONT SIZE=2>Last modified $filemtime</FONT>");
?>

<?php
include("index-footer.html");
?>
```

D.13 bugs.php

```
<!-- bugs.php -->

<?php
include("index-header.html");
?>

    <H3>Bug Reports</H3>

    <P>--</P>
    <?php
        $filemod = filemtime(__FILE__);
        $filemtime = date("F j Y h:i:s A", $filemod);
        echo ("<FONT SIZE=*2*>Last modified $filemtime</FONT>");
    ?>

<?php
include("index-footer.html");
?>
```

D.14 mailing.php

```
<!-- mailing.php -->

<?php
include("index-header.html");
?>

    <H3>Malinglist - for intern use only</H3>
    <tt>cphstl@diku.dk</tt> is the discussion forum for all cphstl developers. You have
    full access to this mailing list from inside DIKUs local area network.
    For more details, see

    <P>
    <a href="http://mail.diku.dk/mailman/listinfo/cphstl">http://mail.diku.dk/mailman/listinfo/cphstl</a>
    </P>

    <P>--</P>
    <?php
        $filemod = filetype(__FILE__);
        $filemtime = date("F j Y h:i:s A", $filemod);
        echo ("<FONT SIZE=*2*^>Last modified $filemtime</FONT*>");
    ?>

<?php
include("index-footer.html");
?>
```

D.15 *contact.php*

```
<!-- contact.php -->

<?php
include("index-header.html");
?>

<H3>Contact Us</H3>

If you have ideas how to improve existing STL implementations, want to be a contributor in
this project, want to make a weekend project on one of the modules, or want to write a written
project on some of the modules, please, send a message to
<A HREF="mailto:jyrki-stl@diku.dk">jyrki-stl@diku.dk</A>.

<P>--</P>
<?php
    $filemod = filetype(__FILE__);
    $filemtime = date("F j Y h:i:s A", $filemod);
    echo ("<FONT SIZE=*2*^>Last modified $filemtime</FONT*");
?>

<?php
include("index-footer.html");
?>
```

D.16 admin/index.php

```
<!-- admin/index.php -->
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
  <TITLE>
    Database administration af Copenhagen STL
  </TITLE>
</HEAD>

<BODY BGCOLOR="#FFFFFF" LINK="#333399" VLINK="#000066" ALINK="#919191">
<P><FONT SIZE="3">Index | <A HREF="contributor.php">Contributor administration</A> |
<A HREF="module.php">Module administration</A> | <A HREF="news.php">News administration</A> |
<A HREF="links.php">Link administration</A></FONT></P>
<P>&nbsp;</P>

<H3>What's this?</H3>
<B>Contributor administration:</B> <BR>
<UL>
  <LI>Add new contributors, update and delete existing contributors.</LI>
  <LI>Add modules to contributors, remove modules from contributor.</LI>
</UL>

<B>Module administration</B>:<BR>
<UL>
  <LI>Add new modules. Update and delete existing modules.</LI>
</UL>

<B>News administration</B>: <BR>
<UL>
  <LI>Add, update and delete news, file news in archive</LI>
  <LI><I>Under construction: Change "max number of news" variable, change when news become obsolete.</I></LI>
</UL>

<B>Link administration</B>: <BR>
<UL>
  <LI>Add a new link, delete and update links.</LI>
  <LI>Add a new category. Update categories. Delete empty categories.</LI>
</UL>

<I><B>Under construction:</B></I><BR>
<UL>
  <LI><I>Literature administration.</I></LI>
  <LI><I>Delete alerts ("are you sure you want to delete blah?"),</I></LI>
  <LI><I>password protection for this site</I></LI>
</UL>

<P>&nbsp;</P>

<P><A HREF=" ../index.php">Back to Copenhagen STLs home page</A></P>
</BODY>
</HTML>
```

D.17 admin/contributor.php

```
<!-- admin/contributor.php -->
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
  <TITLE>
    Database administration af Copenhagen STL
  </TITLE>
</HEAD>

<BODY BGCOLOR="#FFFFFF" LINK="#333399" VLINK="#000066" ALINK="#919191">

  <P><FONT SIZE="3"><A HREF="index.php">Index</A> | Contributor administration |
  <A HREF="module.php">Module administration</A> | <A HREF="news.php">News administration</A> |
  <A HREF="links.php">Link administration</A></FONT></P>

  <?php // connect to database
    $db = mysql_connect("localhost", "lenka", "sedelikaki")
      or die("Could not connect");

    // use cphstl database
    mysql_select_db(cphstl, $db);

    // submit, delete and update contributors
    if ($submit_person) {
      $insert_res = mysql_query("INSERT INTO person (pid, pname, purl) VALUES ('NULL', '$pname', '$purl')")
        or die("Bad query: ".mysql_error());
      echo "New contributor \"\$pname\" inserted.";
    }

    if ($action==delete_p) {
      $select_res = mysql_query("SELECT pname FROM person WHERE pid = $pid");
      extract(mysql_fetch_array($select_res));
      $delete_res = mysql_query("DELETE FROM person WHERE pid = $pid")
        or die("Bad Query: ".mysql_error());
      echo "Contributor $pname deleted.";
    }

    if ($update_person) {
      $update_res = mysql_query("UPDATE person SET pname='$pname', purl='$purl' WHERE pid='$pid'")
        or die("Bad query: ".mysql_error());
      echo "Contributor \"\$pname\" updated.";
    }

    // add and remove modules from contributors
    if ($add_p_mod) {
      $pid_res = mysql_query("SELECT pid FROM person WHERE pname = '$pname'")
        or die("Bad query: ".mysql_error());
      extract(mysql_fetch_array($pid_res));

      $insert_res = mysql_query("INSERT INTO p_modul (pid, modul) VALUES ('$pid', '$modul'")
        or die("Bad query: ".mysql_error());
    }
  </?php>
</BODY>
</HTML>
```

```

        echo "Module \"\$modul\" added to \"\$pname\".";
    }

    if ($action==remove_modul) {
        $select_res = mysql_query("SELECT pname FROM person WHERE pid = $pid");
        extract(mysql_fetch_array($select_res));
        $delete_res = mysql_query("DELETE FROM p_modul WHERE pid = $pid AND modul='\$modul'")
            or die("Bad query: ".mysql_error());
        echo "Module \"\$modul\" removed from contributor \"\$pname\"";
    }

?>

<H3>Add new contributor:</H3>
<FORM METHOD="post" ACTION="<?php echo $PHP_SELF;?>">
  <INPUT TYPE="hidden" NAME="submitted" VALUE="1">
  <TABLE CELLPADDING="0" CELLSPACING="0" BORDER="0" WIDTH="400">
    <TR>
      <TD WIDTH="50">
        <B>Name:</B>
      </TD>
      <TD WIDTH="350">
        <INPUT TYPE="text" NAME="pname" VALUE="" SIZE="40">
      </TD>
    </TR>
    <TR>
      <TD WIDTH="50">
        <B>URL:</B>
      </TD>
      <TD WIDTH="350">
        <INPUT TYPE="text" NAME="purl" VALUE="http://" SIZE="40">
      </TD>
    </TR>
    <TR>
      <TD WIDTH="50">
        &nbsp;
      </TD>
      <TD WIDTH="350" ALIGN="right">
        <INPUT TYPE="reset">
        <INPUT TYPE="submit" NAME="submit_person" VALUE="Submit">
      </TD>
    </TR>
  </TABLE>
</FORM>

<H3>Add a module to a contributor:</H3>
<FORM METHOD="post" ACTION="<?php echo $PHP_SELF;?>">
  <SELECT NAME="pname">
  <?php
    $people = mysql_query("SELECT pid, pname FROM person ORDER BY pname")
      or die ("Bad query: ".mysql_error());
    while ($person_arr = mysql_fetch_array($people)) {

```

```

        extract($person_arr);
        echo "<OPTION>$pname</OPTION>";
    }
    ?>
</SELECT>

<SELECT NAME="modul">
<?php
    $all_modules = mysql_query("SELECT modul FROM modul")
        or die ("Bad query: ".mysql_error());
    while ($module_arr = mysql_fetch_array($all_modules)) {
        extract($module_arr);
        echo "<OPTION>$modul</OPTION>";
    }
    ?>
</SELECT>
<?php
    ?>
<INPUT TYPE="hidden" VALUE="<? echo $modul ?>">
<INPUT TYPE="hidden" VALUE="<? echo $pid ?>">
<INPUT TYPE="submit" NAME="add_p_mod" VALUE="Add">
</FORM>

<!-- Print all contributors -->
<H3>Contributors:</H3>
<TABLE CELLSPACING="3" CELLS="3" BORDER="0">
<?php
    // select all contributors from table
    $people = mysql_query("SELECT pid, pname, purl FROM person ORDER BY pname")
        or die("Bad query: ".mysql_error());

    // fetch one row at a time and put in an array - extract array into variables
    while ($person_arr = mysql_fetch_array($people)) {
        extract($person_arr);

        // check if URL is empty - if not, make a link to URL
        if ($purl AND $purl!="http://") {
            echo "<TR><TD VALIGN="top" WIDTH="200"><A HREF="spurl">$pname</A></TD>";
        } else {
            echo "<TR><TD VALIGN="top" WIDTH="200">$pname</TD>";
        }

        echo "<TD VALIGN="top" WIDTH="50"><A HREF="update_person.php?pid=$pid">Update</A></TD>";
        echo "<TD VALIGN="top" WIDTH="50" ALIGN="left">
            <A HREF="contributor.php?action=delete_p&pid=$pid">Delete</A></TD>";

        echo "<TD WIDTH="400"><TT> ";

        // write all modules (if any) made by this contributor into a string
        $modul = "";

```

```
$modul_str = "";
$has_modul = mysql_query("SELECT modul FROM p_modul WHERE pid='$pid'")
    or die("Bad query: ".mysql_error());
if (mysql_affected_rows() != 0) {
    while ($modul_arr = mysql_fetch_array($has_modul)) {
        extract($modul_arr);
        $modul_str = $modul_str . " , " . $modul . "<FONT SIZE=\"2\">
            (<A HREF=\"contributor.php?action=remove_modul&pid=$pid&modul=$modul\">
                remove</A></FONT>";
    }
} else {
    echo "&nbsp;";
}
// print the string without the last ", "
echo substr ($modul_str, 2);

echo "</TT></TD></TR>";

}
?>
</TABLE>

<?php // close connection
mysql_close ($db);
?>

</BODY>
</HTML>
```

D.18 admin/update_person.php

```
<!-- admin/update_person.php -->
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
  <TITLE>
    Database administration af Copenhagen STL
  </TITLE>
</HEAD>

<BODY BGCOLOR="#FFFFFF" LINK="#333399" VLINK="#000066" ALINK="#919191">

  <?php
    $db = mysql_connect("localhost", "lenka", "sedelikaki")
        or die("Could not connect");

    mysql_select_db(cphstl, $db);

    $people = mysql_query("SELECT pid, pname, purl FROM person WHERE pid = $pid")
        or die("Bad query: ".mysql_error());
    extract(mysql_fetch_array($people));
  ?>

  <H3>Update contributor <?php echo "`$pname`";></H3>
  <FORM METHOD="post" ACTION="contributor.php">
    <TABLE CELLSPACING="0" CELLPADDING="0" BORDER="0" WIDTH="400">
      <TR>
        <TD WIDTH="50">
          <B>Name:</B>
        </TD>
        <TD WIDTH="350">
          <INPUT TYPE="text" NAME="pname" VALUE="<?echo $pname?>" SIZE="40">
        </TD>
      </TR>
      <TR>
        <TD WIDTH="50">
          <B>URL:</B>
        </TD>
        <TD WIDTH="350">
          <INPUT TYPE="text" NAME="purl" VALUE="<?echo $purl?>" SIZE="40">
        </TD>
      </TR>
      <TR>
        <TD WIDTH="50">
          &nbsp;
        </TD>
        <TD WIDTH="350" ALIGN="right">
          <INPUT TYPE="reset">
          <INPUT TYPE="hidden" NAME="pid" VALUE="<?echo $pid?>">
          <INPUT TYPE="submit" NAME="update_person" VALUE="Update">
        </TD>
      </TR>
    </TABLE>
  </FORM>
</BODY>
</HTML>
```

D.18 admin/update_person.php D BILAG - "PROGRAM"-UDSKRIFTER

```
</TABLE>
</FORM>
<?php
mysql_close ($db);
?>
</BODY>
</HTML>
```

D.19 admin/module.php

```

<!-- admin/module.php -->
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
  <TITLE>
    Database administration af Copenhagen STL
  </TITLE>
</HEAD>

<BODY BGCOLOR="#FFFFFF" LINK="#333399" VLINK="#000066" ALINK="#919191">

  <P>
    <FONT SIZE="3"><A HREF="index.php">Index</A> | <A HREF="contributor.php">Contributor administration</A> |
    Module administration | <A HREF="news.php">News administration</A> |
    <A HREF="links.php">Links administration</A></FONT>
  </P>

  <?php // connect to database
    $db = mysql_connect("localhost", "lenka", "sedelikaki")
        or die("Could not connect");

    // select cphstl database
    mysql_select_db(cphstl, $db);

    // submit, delete and update modules
    if ($submit_modul) {
        $insert_res = mysql_query("INSERT INTO modul (modul) VALUES ('$modul')")
            or die("Bad query: ".mysql_error());
        echo "Module \"$modul\" added.";
    }

    if ($action==delete_m) {
        $delete_res = mysql_query("DELETE FROM modul WHERE modul = '$modul'")
            or die("Bad Query: ".mysql_error());
        echo "Module $modul deleted.";
    }

    if ($update_modul) {
        $update_res = mysql_query("UPDATE modul SET modul='$module' WHERE modul='$modul'")
            or die("Bad query: ".mysql_error());
        echo "Module \"$modul\" updated to \"$module\".";
    }

  ?>

  <H3>Add new module:</H3>
  <FORM METHOD="post" ACTION="<?php echo $PHP_SELF;?>">
    <INPUT TYPE="hidden" NAME="submitted" VALUE="1">
    <INPUT TYPE="text" NAME="modul" VALUE="" SIZE="20"><BR>
    <INPUT TYPE="reset">
    <INPUT TYPE="submit" NAME="submit_modul" VALUE="Submit">

```

```
</FORM>

<H3>Modules:</H3>
<TABLE CELLPADDING="3" CELSPACING="5" WIDTH="400">
  <?php
    // select modules from table
    $modules = mysql_query("SELECT modul FROM modul ORDER BY modul")
    or die("Bad query: ".mysql_error());

    // fetch one row at a time and put in an array - extract array into variables
    // print all modules
    while ($modul_arr = mysql_fetch_array($modules)) {
      extract($modul_arr);
      echo "<TR><TD WIDTH=\"100\"><TT>$modul </TT></TD>";
      echo "<TD WIDTH=\"50\"><A HREF=\"update_module.php?modul=$modul\">Update</A></TD>";
      echo "<TD WIDTH=\"50\"><A HREF=\"module.php?action=delete_m&modul=$modul\">Delete</A></TD>";

    }
  ?>
</TABLE>

<?php // close connection
mysql_close ($db);
?>

</BODY>
</HTML>
```

D.20 admin/update_module.php

```
<!-- admin/update_module.php -->
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
  <TITLE>
    Database administration af Copenhagen STL
  </TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF" LINK="#333399" VLINK="#000066" ALINK="#919191">
  <?php
    $db = mysql_connect("localhost", "lenka", "sedelikaki")
      or die("Could not connect");

    mysql_select_db(cphstl, $db);

    $modules = mysql_query("SELECT modul FROM modul WHERE modul = '$modul'")
      or die("Bad query: ".mysql_error());
    extract(mysql_fetch_array($modules));
  ?>
  <H3>Update module <?php echo "'`$modul'`?></H3>
  <FORM METHOD="post" ACTION="module.php">
    <INPUT TYPE="text" NAME="module" VALUE="<?echo $modul?>" SIZE="30">
    <INPUT TYPE="hidden" NAME="modul" VALUE="<?echo $modul?>"><BR>
    <INPUT TYPE="reset">
    <INPUT TYPE="submit" NAME="update_modul" VALUE="Update">
  </FORM>

  <?php
    mysql_close ($db);
  ?>
</BODY>
</HTML>
```

D.21 admin/news.php

```
<!-- admin/news.php -->
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<TITLE>
Database administration af Copenhagen STL
</TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF" LINK="#333399" VLINK="#000066" ALINK="#919191">
<P><FONT SIZE="3"><A HREF="index.php">Index</A> | <A HREF="contributor.php">Contributor administration</A> |
<A HREF="module.php">Module administration</A> | News administration |
<A HREF="links.php">Link administration</A></FONT></P>
<?php // connect to database
$db = mysql_connect("localhost", "lenka", "sedelikaki")
or die("Could not connect");
// select cphstl database
mysql_select_db(cphstl, $db);
// submit, delete and update news
if ($submit_news) {
    $today = date("F j, Y");
    $insert_res = mysql_query("INSERT INTO news (nid, date, new) VALUES ('NULL', '$today', '$new')")
or die("Bad query: ".mysql.error());
    echo "<B>$today:</B> News \"\$new\" added.";
}
if ($action==delete_n) {
    $delete_res = mysql_query("DELETE FROM news WHERE nid = $nid")
or die("Bad Query: ".mysql.error());
    echo "News deleted.";
}
if ($update_news) {
    $update_res = mysql_query("UPDATE news SET new='$new' WHERE nid=$nid")
or die("Bad query: ".mysql.error());
    echo "News updated.";
}
// file news into archive or retrieve from archive
if ($action==archive_n) {
    $archive_res = mysql_query("UPDATE news SET archived=1 WHERE nid=$nid")
or die("Bad Query: ".mysql.error());
    echo "News filed.";
}
```

```

        if ($action==not_archive_n) {
            $archive_res = mysql_query("UPDATE news SET archived=0 WHERE nid='$nid'")
            or die("Bad Query: ".mysql_error());
            echo "News brought back from archive.";
        }

    ?>

<H3>Add news:</H3>
<I>Use of HTML is OK, just remember to dobbeltcheck everything.</I>
<FORM METHOD="post" ACTION="<?php echo $PHP_SELF;?>">
    <INPUT TYPE="hidden" NAME="issubmitted" VALUE="1">
    <TEXTAREA ROWS="10" COLS="70" NAME="new"></TEXTAREA><BR>
    <INPUT TYPE="reset">
    <INPUT TYPE="submit" NAME="submit_news" VALUE="Submit">
</FORM>

<H3>News:</H3>
<TABLE CELLPADDING="3" CELLSPACING="5" WIDTH="700" BORDER="0">
    <?php
        // select all news from table
        $news = mysql_query("SELECT nid, date, new, archived FROM news ORDER BY nid DESC")
        or die("Bad query: ".mysql_error());

        // fetch one row at a time and put in an array - extract array into variables
        while ($new_arr = mysql_fetch_array($news)) {
            extract($new_arr);
            // print all news that are not archived
            if ($archived == 0) {
                echo "<TR><TD VALIGN="\"top\" WIDTH="\"100\" NOWRAP><B>$date </B></TD>";
                echo "<TD VALIGN="\"top\" WIDTH="\"420\">$new</TD>";
                echo "<TD VALIGN="\"top\" WIDTH="\"50\">";
                echo "<A HREF="\"update_news.php?nid=$nid\">Update</A></TD>";
                echo "<TD VALIGN="\"top\" WIDTH="\"50\">";
                echo "<A HREF="\"news.php?action=delete_n&nid=$nid\">Delete</A></TD>";
                echo "<TD VALIGN="\"top\" WIDTH="\"80\">";
                echo "<A HREF="\"news.php?action=archive_n&nid=$nid\">Archive</A></TD></TR>";
            }
        }
    ?>
</TABLE>

<H3>Archived news:</H3>
<TABLE CELLPADDING="3" CELLSPACING="5" WIDTH="700" BORDER="0">
    <?php
        // select all news from table
        $news = mysql_query("SELECT nid, date, new, archived FROM news ORDER BY nid DESC")
        or die("Bad query: ".mysql_error());

        // fetch one row at a time and put in an array - extract array into variables
        while ($new_arr = mysql_fetch_array($news)) {

```

```
extract($new_arr);
// print all news that are archived
if ($sarchived == 1) {
    echo "<TR><TD VALIGN=\"top\" WIDTH=\"100\"><B>$date </B></TD>";
    echo "<TD VALIGN=\"top\" WIDTH=\"420\">$new</TD>";
    echo "<TD VALIGN=\"top\" WIDTH=\"50\">";
        <A HREF=\"update_news.php?nid=$nid\">Update</A></TD>";
    echo "<TD VALIGN=\"top\" WIDTH=\"50\">";
        <A HREF=\"news.php?action=delete_n&nid=$nid\">Delete</A></TD>";
    echo "<TD VALIGN=\"top\" WIDTH=\"80\">";
        <A HREF=\"news.php?action=not_archive_n&nid=$nid\">
        Retrieve from archive</A></TD></TR>";
    }
}
?>
</TABLE>

<?php // close connection
mysql_close ($db);
?>

</BODY>
</HTML>
```

D.22 admin/update_news.php

```
<!-- admin/update_news.php -->
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
  <TITLE>
    Database administration af Copenhagen STL
  </TITLE>
</HEAD>

<BODY BGCOLOR="#FFFFFF" LINK="#333339" VLINK="#000066" ALINK="#919191">
  <?php echo $error ?>

  <?php
    $db = mysql_connect("localhost", "lenka", "sedelikaki")
      or die("Could not connect");

    mysql_select_db(cphstl, $db);

    $news = mysql_query("SELECT nid, date, new FROM news WHERE nid = $nid")
      or die("Bad query: ".mysql_error());
    extract(mysql_fetch_array($news));
  ?>

  <H3>Update news:</H3>
  <I>The use of HTML is fine, just remember to dobbeltcheck everything.</I>
  <FORM METHOD="post" ACTION="news.php">
    <TEXTAREA ROWS="10" COLS="70" NAME="new" MAXLENGHT="40"><?echo $new?></TEXTAREA><BR>
    <INPUT TYPE="reset">
    <INPUT TYPE="hidden" NAME="nid" VALUE="<?echo $nid?>">
    <INPUT TYPE="submit" NAME="update_news" VALUE="Update">
  </FORM>

  <?php
    mysql_close ($db);
  ?>

</BODY>
</HTML>
```

D.23 admin/links.php

```

<!-- admin/links.php -->
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
  <TITLE>
    Database administration af Copenhagen STL
  </TITLE>
</HEAD>

<BODY BGCOLOR="#FFFFFF" LINK="#333399" VLINK="#000066" ALINK="#919191">

  <P><FONT SIZE="3"><A HREF="index.php">Index</A> | <A HREF="contributor.php">Contributor administration</A> |
  <A HREF="module.php">Module administration</A> | <A HREF="news.php">News administration</A> |
  Links administration</FONT></P>

  <?php // connect to database
    $db = mysql_connect("localhost", "lenka", "sedelikaki")
        or die("Could not connect");

    // use cphstl database
    mysql_select_db(cphstl, $db);

    // submit, delete or update link or category
    if ($submit_link) {
        $cid_res = mysql_query("SELECT cid FROM link_cat WHERE category = '$category'")
            or die("Bad query: ".mysql_error());
        extract(mysql_fetch_array($cid_res));

        $insert_res = mysql_query("INSERT INTO links (lid, lname, lur1, cid)
            VALUES ('NULL', '$lname', '$lur1', '$cid')")
            or die("Bad query: ".mysql_error());
        echo "New link <A HREF=\"$lur1\">$lname</A> added.";
    }

    if ($submit_category) {
        $insert_res = mysql_query("INSERT INTO link_cat (cid, category) VALUES ('NULL', '$category')")
            or die("Bad query: ".mysql_error());
        echo "New Category \"$category\" added.";
    }

    if ($action==delete_l) {
        $delete_res = mysql_query("DELETE FROM links WHERE lid = $lid")
            or die("Bad Query: ".mysql.error());
        echo "Link deleted.";
    }

    if ($action==delete_cat) {
        $delete_res = mysql_query("DELETE FROM link_cat WHERE cid = $cid")
            or die("Bad Query: ".mysql.error());
        echo "Empty category deleted.";
    }

```

```
}

if ($update_link) {
    $select_res = mysql_query("SELECT cid FROM link_cat WHERE category = '$category'")
    or die("Bad query: ".mysql_error());
    extract(mysql_fetch_array($select_res));

    $update_res = mysql_query("UPDATE links SET lname='$lname', lurl='$lurl', cid='$cid' WHERE lid='$lid'")
    or die("Bad query: ".mysql_error());
    echo "Link <A HREF=\"\$lurl\">{$lname}</A> updated.";
}

if ($update_cat) {
    $update_res = mysql_query("UPDATE link_cat SET category='$new_category' WHERE cid='$cid'")
    or die("Bad query: ".mysql_error());
    echo "Category $new_category updated.";
}

?>

<TABLE CELLSPACING="0" CELLPADDING="0" BORDER="0" WIDTH="700">
<TR>
<TD VALIGN="TOP" WIDTH="400">
<H3>Add a new link:</H3>
<FORM METHOD="post" ACTION="<?php echo $PHP_SELF;?>">
<INPUT TYPE="hidden" NAME="issubmitted" VALUE="1">
<TABLE CELLSPACING="0" CELLPADDING="0" BORDER="0" WIDTH="400">
<TR>
<TD WIDTH="50">
<B>Title:</B>
</TD>
<TD WIDTH="350">
<INPUT TYPE="text" NAME="lname" VALUE="" SIZE="40">
</TD>
</TR>
<TR>
<TD WIDTH="50">
<B>URL:</B>
</TD>
<TD WIDTH="350">
<INPUT TYPE="text" NAME="lurl" VALUE="http://" SIZE="40">
</TD>
</TR>
<TR>
<TD WIDTH="400" COLSPAN="2" ALIGN="left">
<B>Choose Category:</B>
<SELECT NAME="category">
<?php
    $categories = mysql_query("SELECT cid, category FROM link_cat ORDER BY category")
    or die ("Bad query: ".mysql_error());
    while ($cat_arr = mysql_fetch_array($categories)) {
```

```

                                extract($cat_arr);
                                echo "<OPTION>$category</OPTION>";
                                }
                            ?>
                        </SELECT>
                    </TD>
                </TR>
            </TR>
            <TR>
                <TD WIDTH="50">
                    &nbsp;
                </TD>
                <TD WIDTH="350" ALIGN="right">
                    <INPUT TYPE="hidden" VALUE="<? echo $cid ?>">
                    <INPUT TYPE="reset">
                    <INPUT TYPE="submit" NAME="submit_link" VALUE="Submit">
                </TD>
            </TR>
        </TABLE>
    </FORM>
</TD>
<TD WIDTH="50">&nbsp;
</TD>
<TD VALIGN="TOP" WIDTH="250">
    <H3>Add a category:</H3>
    <FORM METHOD="post" ACTION="<?php echo $PHP_SELF?>">
        <INPUT TYPE="text" NAME="category" VALUE="" SIZE="25"><BR>
        <INPUT TYPE="reset">
        <INPUT TYPE="submit" NAME="submit_category" VALUE="Submit">
    </FORM>
</TD>
</TR>
</TABLE>

<H3>Related Links</H3>
<P><I>(Categories can only be deleted when empty)</I></P>

<TABLE CELLPADDING="0" CELLSPACING="0" BORDER="0" WIDTH="600">
    <?php
        // select all categories from table
        $categories = mysql_query("SELECT cid, category FROM link_cat ORDER BY category")
        or die("Bad query: ".mysql_error());

        // fetch one row at a time and put into array - extract into variables with prefix "cat_"
        while ($cat_arr = mysql_fetch_array($categories)) {
            extract($cat_arr, EXTR_PREFIX_ALL, "cat");

            // select all links in this category
            $links = mysql_query("SELECT lid, lname, lur1, cid FROM links WHERE cid ='$cat_cid'")
            or die("Bad query: ".mysql_error());

            echo "<TR><TD><DL><DT><B>$cat_category</B></DT></DL></TD>";
            echo "<TD><B><A HREF=\"update_cat.php?cid=$cat_cid?category=$cat_category\">";

```

```
Update category</A></B>&nbsp; </TD>";

// if there are no links in category, the category can be deleted
if (mysql_num_rows($links)!=0) {
    echo "<TD>&nbsp; </TD></TR>";
} else {
    echo "<TD><B><A HREF=\"links.php?action=delete_cat&cid=$cat_cid\">
        Delete category</A></B></TD></TR>";
}

// print all links in category
while ($link_arr = mysql_fetch_array($links)) {
    extract($link_arr);
    echo "<TR><TD><DL><DD><A HREF=\"$lurl\">$lname</A></DD></DL></TD>";
    echo "<TD><A HREF=\"update_link?lid=$lid\">Update link</A></TD>";
    echo "<TD><A HREF=\"links.php?action=delete_l&lid=$lid\">Delete link</A></TD></TR>";
}

// echo "</DD>";
}
?>
</TABLE>

<?php // close connection
mysql_close ($db);
?>

</BODY>
</HTML>
```

D.24 admin/update_cat.php

```
<!-- admin/update_cat.php -->
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
  <TITLE>
    Database administration af Copenhagen STL
  </TITLE>
</HEAD>

<BODY BGCOLOR="#FFFFFF" LINK="#333399" VLINK="#000066" ALINK="#919191">

  <?php
    $db = mysql_connect("localhost", "lenka", "sedelikaki")
      or die("Could not connect");

    mysql_select_db(cphstl, $db);

    $this_category = mysql_query("SELECT cid, category FROM link_cat WHERE cid = '$cid'")
      or die("Bad query: ".mysql_error());
    extract(mysql_fetch_array($this_category));

  ?>

  <H3>Update Category <?php echo "'".$category."'?></H3>

  <FORM METHOD="post" ACTION="links.php">
    <INPUT TYPE="text" NAME="new_category" VALUE="<?echo $category?>" SIZE="40">
    <INPUT TYPE="reset">
    <INPUT TYPE="hidden" NAME="cid" VALUE="<?echo $cid?>">
    <INPUT TYPE="submit" NAME="update_cat" VALUE="Update">
  </FORM>

  <?php
    mysql_close ($db);
  ?>

</BODY>
</HTML>
```

D.25 admin/update_link.php

```

<!-- admin/update_links.php -->
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
  <TITLE>
    Database administration af Copenhagen STL
  </TITLE>
</HEAD>

<BODY BGCOLOR="#FFFFFF" LINK="#333399" VLINK="#000066" ALINK="#919191">

  <?php
    $db = mysql_connect("localhost", "lenka", "sedelikaki")
      or die("Could not connect");

    mysql_select_db(cphstl, $db);

    $link = mysql_query("SELECT lid, lname, lurl, cid FROM links WHERE lid = $lid")
      or die("Bad query: ".mysql_error());
    extract(mysql_fetch_array($link));
  ?>

  <H3>Update Link <?php echo "`$lurl`:"?</H3>
  <FORM METHOD="post" ACTION="links.php">
    <TABLE CELLPADDING="0" CELLSPACING="0" BORDER="0" WIDTH="400">
      <TR>
        <TD WIDTH="50">
          <B>Title:</B>
        </TD>
        <TD WIDTH="350">
          <INPUT TYPE="text" NAME="lname" VALUE="<?echo $lname?>" SIZE="40">
        </TD>
      </TR>
      <TR>
        <TD WIDTH="50">
          <B>URL:</B>
        </TD>
        <TD WIDTH="350">
          <INPUT TYPE="text" NAME="lurl" VALUE="<?echo $lurl?>" SIZE="40">
        </TD>
      </TR>
      <TR>
        <TD WIDTH="400" COLSPAN="2" ALIGN="left">
          <B>Choose Category:</B>
          <SELECT NAME="category">
            <?php
              $categories = mysql_query("SELECT cid, category FROM link_cat ORDER BY category")
                or die ("Bad query: ".mysql_error());
              while ($cat_arr = mysql_fetch_array($categories)) {
                extract($cat_arr);
                echo "<OPTION>$category</OPTION>";
            }
          </SELECT>
        </TD>
      </TR>
    </TABLE>
  </FORM>

```

```
        }
        ?>
    </SELECT>
</TD>
</TR>
</TR>
<TR>
<TD WIDTH="50">
    &nbsp;
</TD>
<TD WIDTH="350" ALIGN="right">
    <INPUT TYPE="reset">
    <INPUT TYPE="hidden" NAME="<? echo $category?>">
    <INPUT TYPE="hidden" NAME="lid" VALUE="<?echo $lid?>">
    <INPUT TYPE="submit" NAME="update_link" VALUE="Update">
</TD>
</TR>
</TABLE>

</FORM>

<?php
mysql_close ($db);
?>

</BODY>
</HTML>
```

D.26 tables.sql

```
#tables.sql
CREATE DATABASE IF NOT EXISTS cphst1;
use cphst1;

DROP TABLE IF EXISTS person;
CREATE TABLE person (
    pid          INT NOT NULL AUTO_INCREMENT,
    pname        CHAR(100) NOT NULL,
    purl         CHAR(100),
    PRIMARY KEY (pid)
);
DROP TABLE IF EXISTS modul;
CREATE TABLE modul (
    modul        CHAR(40) NOT NULL,
    PRIMARY KEY (modul)
);
DROP TABLE IF EXISTS p_modul;
CREATE TABLE p_modul (
    pid          CHAR(100) NOT NULL,
    modul        CHAR(40) NOT NULL,
    PRIMARY KEY (pid, modul),
    FOREIGN KEY (pid) REFERENCES person
        ON DELETE CASCADE,
    FOREIGN KEY (modul) REFERENCES modul
        ON DELETE CASCADE
);
DROP TABLE IF EXISTS news;
CREATE TABLE news (
    nid          INT NOT NULL AUTO_INCREMENT,
    date         CHAR(15),
    new          TEXT NOT NULL,
    archived     INT,
    PRIMARY KEY (nid)
);
DROP TABLE IF EXISTS link_cat;
CREATE TABLE link_cat (
    cid          INT NOT NULL AUTO_INCREMENT,
    category     CHAR(40) NOT NULL,
    PRIMARY KEY (cid)
);
DROP TABLE IF EXISTS links;
CREATE TABLE links (
    lid          INT NOT NULL AUTO_INCREMENT,
    lname        CHAR(100),
    lur1         CHAR(100),
    cid          INT NOT NULL,
    PRIMARY KEY (lid),
    FOREIGN KEY (cid) REFERENCES category
        ON DELETE NO ACTION
);
```