interacTV

Jon Rønning Vatne

Våren 2007

Forord

Denne rapporten er en del av hovedprosjektet «interacTV», og markerer avslutningen av en 3-årig bachelorutdanning i Medieteknikk ved Høgskolen i Gjøvik. Arbeidet ar blitt gjort i løpet av våren 2007.

Prosjektet er initialisert av studenter, og det bygger på en idé som går ut på å gjøre det mulig å skape filmer hvor den som ser filmen kan være med å bestemme hva som skal skje i filmen. *interacTV* er et nettbasert system som gjør dette mulig.

Hovedprosjektet er en videreutvikling av et prosjekt som ble gjennom ført høstsemesteret 2006 i emnet Fordybning i medieteknologi (IMT3391) på HiG.

Prosjektet har blitt gjennomført av én av iniativtakerne og utviklingsprosessen har derfor vært veldig givende.

Takk til:

- Øyvind Kolås for å gjøre en god veiledningsjobb ved å alltid styre meg i riktig retning.
- Mariann Nikman Freij for hjelp med logo, plakat og figur 3.1 på side 15, og selvfølgelig fordi du er så fin å være sammen med.
- De som har satt et bordtennisbord i så umidelbar nærhet til grupperommet.
- Stian Lund Hansen for hjelp med CSS og for alltid å vise interesse for prosjektet.
- Einar Jørgen Haraldseid, Lars Sigurdson og Bjørn Haugland for et alternativt arbeidsmiljø.

Gjøvik, 24.mai 2007

Jon Rønning Vatne

Innhold

Kapittel 1

Innledning

Forestill deg at du ser en film hvor du kan være med å påvirke hva som skal skje i filmen. Vi har valgt å gjøre dette mulig.

1.1 Problemområde og avgrensning

I dette prosjektet skulle vi lage et nettbasert system for redigering og visning av interaktive filmer.

Med «interaktive filmer» menes filmer hvor mottakeren kan være med å bestemme hendelsesforløpet i filmen ved å ta valg han blir stilt ovenfor i løpet av filmvisningen.

Det skal være enkelt for brukere å laste opp filmklipp og sette disse sammen i en sammenheng, med valg og valgalternativer som utgjør strukturen og rammene for visningen av den interaktive filmen. Denne strukturen, med dens rammer, kan sees på som den interaktive filmens skjelett og «et interaktivt manus».

For at systemet skal være lett å bruke for brukeren, må brukeren enkelt kunne laste opp filmfiler fra sin klientmaskin og kunne få tilgang til disse i systemet. Brukeren skal ikke trenge å konvertere filene sine til et spesielt format. For at filmfilene skal kunne brukes av resten av systemet må de derfor transkodes til et ønsket format på tjenersiden.

Når en bruker velger å publisere sin interaktive film må denne filmen bli gjort tilgjengelig for allmennheten. Ikke minst skal det være mulig for mottakerne å spille av, interaktere med og påvirke filmvisningen, slik brukeren hadde forespeilet seg. For å gjøre dette mulig må det lages en filmavspiller som innehar all den nødvendige funksjonaliteten.

1.1.1 Effekt- og resultatmål

Effektmål

- Tilby en enkel måte å laste opp filmklipp og klargjøre filmvisninger for brukere.
- Gjøre det mulig å gjennomføre og vise interaktive filmer for mottakerne.

Resultatmål

- Lage et nettbasert system som gjør det mulig for registrerte brukere å laste opp filmklipp i forskjellige filformater og filstørrelser, for så sette sammen og lagre disse i en sammenheng, med valg og valgmuligheter som vil legge til rette for visningen av den interaktive filmen.
- Lage en spiller som kan implementeres på en nettside og som kan vise filmklipp og valg for mottakere, i den rekkefølgen som er angitt i visningsstrukturen som er lagret på tjener av en bruker.

1.2 Formål og bakgrunn

Dette hovedprosjektet er en videreutvikling av et prosjekt som ble gjennomført høsten 2006 i emnet Fordypning i medieteknikk (IMT3391). Den gangen bestod prosjektgruppen av fire studenter som hver hadde sitt ansvarsområde.

Prosjektet ble i utgangspunktet påbegynt fordi videoavspilling på nett ved hjelp av Flash ble sett på som et veldig spennende og aktuelt tema. I tillegg ble

interaktive filmvisninger sett på som et lite utbredt men et konsept med stort potensiale, som absolutt burde være mer tilgjengelig for allmennheten.

1.3 Målgruppe

Mottakerne: Er de som vil besøke nettstedet for å se, oppleve og bruke de interaktive filmene, og som tar valg som påvirker historien i filmene.

Brukere: Er de som vil benytte seg av «systemet bak» for å laste opp filmklipp og koble disse sammen i en struktur med valg og valgmuligheter for å skape en filmvisning som kan sees og brukes av *mottakerne*.

1.4 Prosjektorganisering

Studenter: Jon Rønning Vatne (04HBMETEA)

Prosjektveileder: Øyvind Kolås

Oppdragsgiver: Høgskolen i Gjøvik.

Siden dette prosjektet er initialisert på grunn av studentenes egne ønsker, framfor gitt i oppdrag av eksterne oppdragsgivere, vil HiG[1] fungere som oppdragsgiver.

1.5 Rammer

1.5.1 Arbeidsmengde

Hovedprosjektet har, ifølge emnebeskrivelsen [1], en arbeidsmengde tilsvarende 20 studiepoeng. 20 studiepoeng er beregnet til å tilsvare en arbeidsmengde på ca 25 timer i uka for en student, og dette vil altså tilsvare prosjektetets arbeidsmengde.

1.6 Egen bakgrunn og kompetanse

Gruppen består av en student som fullfører sin bachelor-grad i Medieteknikk ved HiG.

Gruppen har en del erfaring med PHP-programmering. Gruppen har også litt erfaring med Flash-utvikling grunnet kunnskapen vi opparbeidet oss høstsemesterets prosjekt.

Bortsett fra dette måtte vi lære oss de andre teknologiene brukt i systemet. Før prosjektet hadde vi for eksempel ingen erfaring eller kunnskap innenfor bruk av Linux, drifting av servere, og språk som BASH og Java.

I tillegg har vi, på grunn av valg tatt rundt arbeidsmetoder, opparbeidet oss erfaring innenfor teknologier som LaTeX, SVN og Doxygen.

[1] Høgskolen i Gjøvik

1.7 Arbeidsmetoder

1.7.1 Utviklingsmodell

I dette prosjektet har det, på grunn av arbeidet utført høsten 2006 i emnet IMT3391, og på grunn av at mange av ideene rundt krav til systemet har kommet fra de som har utført prosjektet, vært lett å spesifisere en del av kravene tidlig. Systemet kunne enkelt deles inn i inkrementer siden det er flere klart definerbare og adskilte prosesser som utgjør systemet.

På grunnlag av dette var det naturlig å velge en inkrementell utviklingsmodell, hvor det som skulle gjøres i de aktuelle inkrementene ble avklart og planlagt i detalj i begynnelsen av hvert inkrement. Denne type utviklingsmodell har vært fordelaktig siden den har gjort det mulig å bruke nye idéer som har kommet fram underveis i prosjektperioden, og å omprioritere oppgaver som skulle utføres fortløpende.

1.7.2 Statusmøter

Hver tirsdag, med noen unntak grunnet annen jobbing, har det blitt gjennomført møter med veileder hvor progresjon siden sist har blitt gjennomgått, og planlegging for videre jobbing fram til neste møte har blitt avklart.

Det har ikke blitt fokusert på å skrive statusrapporter; hovedsaklig fordi arbeid uansett ikke skulle delegeres til flere personer.

1.8 Organisering av rapporten

1.8.1 Målgruppe for rapporten

Målgruppen for denne rapporten vil hovedsaklig være veileder, Øyvind Kolås, og den eksterne sensoren som vil evaluere prosjektet. Derfor vil rapporten være relativt teknisk og det er forventet at leseren har en viss kunnskap til blant annet programutvikling.

Sensor vil bruke prosjektrapporten med vedlegg til å evaluere prosjektets omfang og kvalitet.

1.8.2 Terminologi

For å klargjøre hva som til en hver tid menes i rapporten kommer her en forklaring på noen ord og begreper.

- Brukeren og mottakeren omtales som *han* (fremfor *han/hun* som er mer tungvint både å skrive og lese).
- *Vi* brukes fremfor *jeg*; dette kan muligens i noen tilfeller føre til uklarhet med tanke på at det finnes to grupper: Den som gjennomførte prosjektet i emnet IMT3391 i høstsemesteret '06, og den som gjennomfører hovedprosjekt vårsemesteret '07.
- En *interaktiv film* blir omtalt som *filmen* i enkelte tilfeller.

- Med *filmklipp* menes filer brukeren har lastes opp for at de skal kunne brukes i en interaktiv film.
- En interaktiv film har flere *valg*, disse omtales også som *branches*.
- Et valg har én eller flere *valgalternativ*, disse kan også omtales som *svaralternativer* eller *linker*.
- Et *standardvalg* er en spesiell type alternativ (i et valg).
- *Filmavspilleren* er Flash-programmet som skal vise den interaktive filmen.
- Den interaktive filmens *struktur* blir også omtalt som *graf*, *valg-graf*, *option graphs* og tidligere versjoner blir også omtalt som *valgtrær*.
- *Chunks* er en beskrivelse på en del av en binærfil.

Kapittel 2

Kravspesifikasjon

Selv om, og kanskje nettopp fordi, prosjektet i praksis ikke har hat noen ekstern arbeidsgiver, har det vært viktig å klargjøre hva som forventes av systemet

For tidlig å få oversikt over hva som skulle utvikles i prosjektet, utviklet vi *Use cases*; etter å ha kartlagt på et grovere plan hva som forventes av systemet. Use casene var et godt verktøy for å tidlig kartlegge og konkretisere kravene til systemet, på et mer detaljert nivå.

2.1 Overordnede funksjonelle krav

Vi skal lage et brukersystem som gjør det mulig for registrerte brukere å logge seg sikkert inn og ut, og endre persondata og bilde.

Vi skal lage en løsning som lar brukeren laste opp store filmfiler til serveren. Det må også lages en applikasjon på tjenersiden som tar imot disse filene og registrerer de i systemet.

Opplastede filmfiler må automatisk transkodes av systemet til et format som er kompatibelt med resten av systemet.

Brukerne må ha mulighet til å opprette og redigere strukturer for interaktive filmer, ved å sette sammenhenger mellom sine opplastede filmklipp, med blant annet valg og valgalternativer.

De interaktive filmene skal kunne spilles av for mottakeren. Under avspillingen skal mottakeren ha mulighet til å ta valg han blir stilt ovenfor, og filmvisningen skal automatisk spille videre ut ifra hva mottakeren har valgt. I realiteten vil spilleren spille av mange forskjellige filmklipp, men for mottakeren bør det virke som om han er med å bestemme handlingen i en sammenhengende film.

For at andre lett skal få en oppfatning av hvilke muligheter som ligger i dette systemet, er det også ønskelig å lage en interaktiv film som viser potensialet systemet har.

2.2 Funksjonelle krav

Nettsiden

Registrerte brukere skal ha mulighet til å endre persondata om seg selv.

Registrerte brukere skal ha mulighet til å invitere andre til å bli brukere.

Potensielle mottakere som besøker nettsidene skal få listet opp publiserte interaktive filmer, sortert etter når de sist ble endret, med disse øverst.

Filmspilleren og redigering av filmstruktur

Mottakeren skal ha mulighet til å pause filmvisningen og justere volumet på lyden.

På tidspunkter brukeren har bestemt, skal det vises gitte valg for mottakeren. Spillerens oppførsel avhenger av hvordan mottakeren samhandler med spilleren når et valg vises.

Om mottakeren velger et av alternativene, skal spilleren spille av det filmklippet som er koblet til det aktuelle valget. Eventuelt skal mottakeren sendes videre til en ekstern nettside, og filmvisningen stoppes, alt etter som hva brukeren har valgt at skal skje i det aktuelle valget.

Om brukeren har satt at det aktuelle valget skal ha en tidsbegrensning og mottakeren nøler for lenge med å ta et valg, skal valget forsvinne og det aktuelle filmklippet skal fortsette å spille.

Når en interaktiv film spilles av skal det kunne vises flere valg innenfor tiden det tar å spille av et filmklipp. Om mottakeren velger et av alternativene til et valg som vises før slutten av det aktuelle filmklippet, vil filmspilleren laste et nytt filmklipp og vise dette for mottakeren.

Brukeren skal kunne bestemme om filmvisningen skal pauses eller spille videre, når et valg vises.

Det skal være mulig for brukeren å sette at et alternativ skal kunne linkes til et annet punkt i strukturen, slik at man skal slippe å lage en del av historien på nytt om man ønsker at flere alternativer (ikke nødvendigvis i samme valg) skal føre til den samme historieutviklingen. Det skal riktignok ikke være mulig å linke til et sted i strukturen hvor filmavspilleren allerede har spilt gjennom, dette for å forhindre at man kan se ett og samme filmklipp på nytt og gå i løkke.

Brukeren kan sette et av alternativene i det siste valget til et filmklipp til å være et standardvalg, om tidsavgrensning er satt for dette valget. Om mottakeren venter til valget har blitt vist så lenge som tidsavgrensningen tillater (som er satt av brukeren), vil spilleren begynne å spille av standardvalgets tilhørende filmklipp.

Om et alternativ i et valg er satt til å være et standardvalg, og det aktuelle valget er det siste valget som skal vises med det aktuelle filmklippet, skal brukeren kunne sette det aktuelle alternativet til å være usynlig. Dette vil si at mottakeren ikke skal kunne se dette alternativet i valget, men om han lar ventetiden utløpe, vil spilleren begynne å spille av det usynlige standardvalgets tilhørende filmklipp. Dette valget vil altså ikke være tilgjengelig for mottakeren på annet vis enn ikke velge et alternativ før ventetiden for valget utløper.

Brukeren skal kunne redigere beskrivende informasjon, som tittel, beskrivelse, bilde og rulletekst, om sine interaktive filmer, og velge å publisere dem. Når en film publiseres skal den være tilgjengelig på nettsidene som er åpen for allmennheten.

Opplasting og transkoding

Brukeren skal kunne velge flere filer fra sin klientmaskin som settes i kø for så å bli lastet opp én etter en. Det skal også gis lett forståelige tilbakemeldinger om hvor mye som er lastet opp til tjenersiden.

Databasen skal til enhver tid få beskjed om hvilke filmklipp som lastes opp fra klientsiden, hvor de lastes opp til og hvor mye som er lastes opp, i bytes og prosentvis.

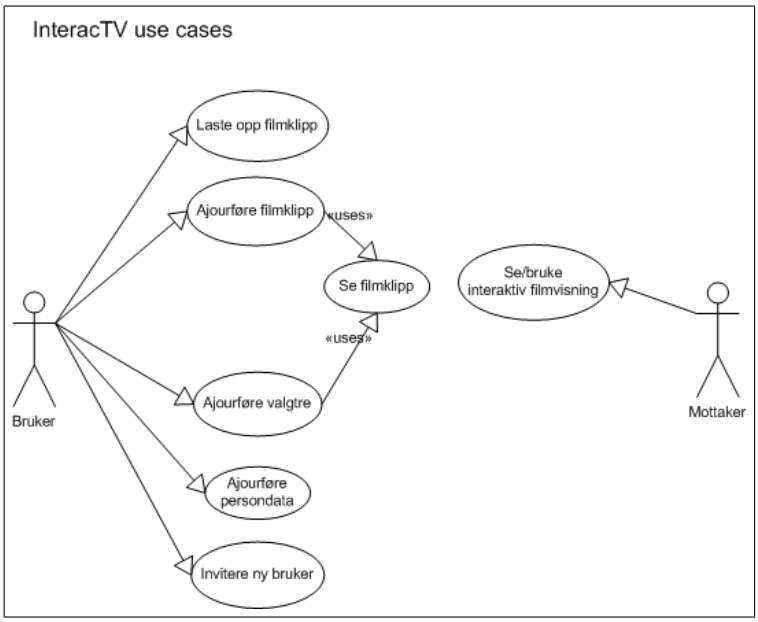
Databasen skal også til en hver tid få beskjed om hvilke opplastede filmklipp som er oppdaget av transkodingsskriptet, hvilke som transkodes, og hvilke som er ferdig transkodet.

2.3 Use cases

Ut fra kravene beskrevet ovenfor, utviklet vi use cases. Use casene vi har kommet fram til er vist i figur 2.1 på neste side. Overordnede use case-beskrivelser for alle disse, og detaljerte use case-beskrivelser for et utvalg av disse, finner man i vedlegg D på side 72.

Det bør riktignok nevnes at disse use cases ble utviklet tidlig i prosjektfasen. Nye krav til funksjonalitet har kommet underveis i prosjektet, og noen av use casene ville derfor vært enda mer komplekse om disse kravene ble satt tidligere i prosjektfasen.

Use casene har, sammen med tekstene med de funksjonelle krav, likevel vært gode verktøy for komme igang med utviklingen av prosessen.



Figur 2.1: Use case-diagram

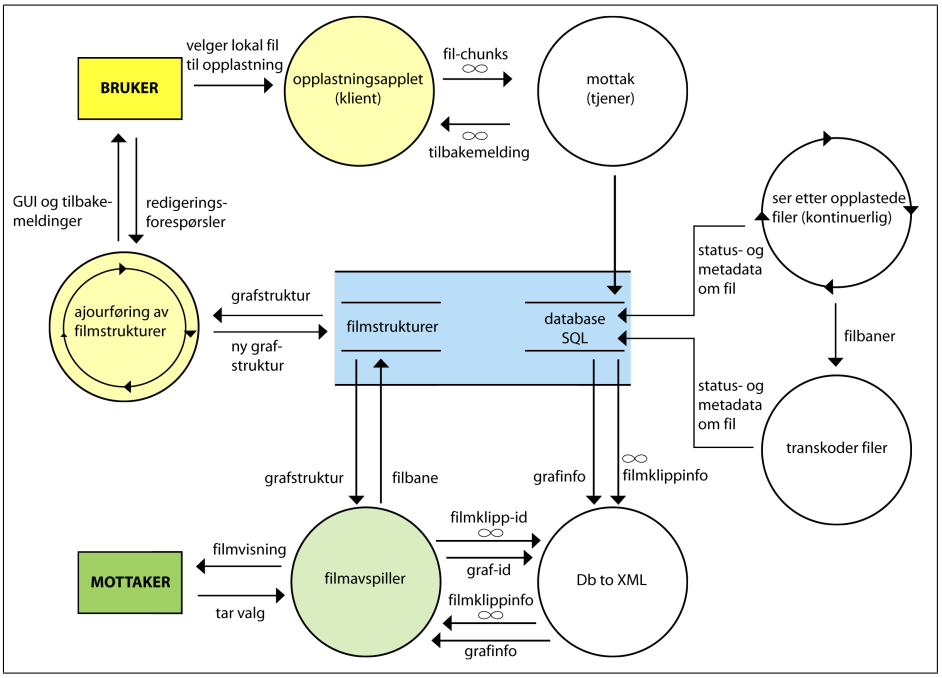
Kapittel 3

Design

Dette kapittelet beskriver hvordan systemet er bygd opp og hvordan de forskjellige prosessene henger sammen med hverandre.

3.1 Systemets arkitektur

Figur 3.1 viser et kontekstdiagram over hvordan systemet skal fungere. Diagrammet beskriver hvilke data som går til og fra de fire hovedprosessene i systemet: Filopplastning, transkoding av filer, redigering av interaktive filmstrukturer og avspilling av interaktive filmer.



Figur 3.1: Kontekstdiagram over systemet

Øverst i figuren ser vi opplastnings-prosessen[1]. Opplastningen skjer ved at en klient-applikasjon laster opp filer i fragmenter, til en applikasjon på tjenersiden, som setter de sammen til hele filer. Tjenerapplikasjonen, som tar imot filfragmentene, kommuniserer med databasen.

Til høyre i figuren er transkodingsdelen[2] av systemet. Denne prosessen kjører automatisk og ser etter filer som skal transkodes. Når transkodingen er inititialisert kommuniserer applikasjonen med databasen fortløpende.

Til venstre i figuren beskrives grafredigeringsdelen[3] av systemet. Prossesen viser GUI til brukeren, som kan bestemme hvordan applikasjonen skal redigere grafene.en beskrives grafredigeringsdelen[4] av systemet. Prossesen viser GUI til brukeren, som kan bestemme hvordan applikasjonen skal redigere grafene.

Nederst i figuren ser vi filmavspillerdelen[5] av systemet. Ut ifra en parameter den er blitt gitt, henter den først informasjon om valg-grafen ved å lese

[1]Opplastingen er beskrevet ytterligere i underkapittel 4.1 på side 21.
[2]Transkodingen er beskrevet ytterligere i underkapittel 4.2 på side 29.
[3]Grafredigeringen er beskrevet ytterligere i underkapittel 4.3 på side 33.
[4]Grafredigeringen er beskrevet ytterligere i underkapittel 4.3 på side 33.
[5]Filmavspilleren er beskrevet ytterligere i underkapittel 4.4 på side 41.

data generert av databasekommunikasjonsapplikasjonen *Db to xml*, med graf-id som parameter. Deretter leses valg-grafen inn i programmet ved at den aktuelle valg-graf-XML-filen leses. Når dette er gjort hentes informasjon om hvert enkelt filmklipp i valg-grafen ved at databasekommunikasjonsapplikasjonen blir kjørt igjen, nå med filmklipp-id'ene som parametre. Når alt dette er gjort kan filmvisningen begynne.

Nettsidene

I tillegg til prosessene beskrevet over er det også nødvendig å lage et system for generell visning av nettsidene, innlogging av registrerte brukere, og å knytte de tidligere nevnte prosessene sammen med nettsidene.

Fysisk arkitektur

For at systemet skal kunne vise filer uten problemer og ikke få svekket ytelse i forbindelse med opplasting og transkoding av filmfiler, har vi valgt å fordele arbeidet på to tjenermaskiner. Dette er fordi transkodingen av filmer er en ressurskrevende prosess, og med den begrensede datakraften vi har fått tildelt i dette prosjektet, hadde det ikke vært fornuftig å kjørt flere prosesser på den samme maskinen som transkoder.

Den ene tjenermaskinene er satt til å ta imot filopplastinger og transkode filmer, og har derfor fått navnet *transcoder*. Den andre maskinen har ansvar for alle andre prosesser, det vil si visningen av filmer, redigering av valg-grafer, og visning av nettsider. På denne maskinen ligger også databasen, alle transkodede filmklipp og alle valg-grafer. Denne maskinen kalles *webserver*.

3.2 XML-datastruktur

En viktig del av systemet er lagringen av strukturene til de interaktive filmene. Disse strukturene er vist i figur 3.1 på forrige side som *filmstrukturer*. For at brukeren skal kunne redigere strukturen til en interaktiv film, og for at filmavspilleren skal kunne hente og forstå strukturen for å spille av filmen, må disse dataene lagres på et strukturert vis som alle prosessene i systemet forstår. Men før man kan bestemme hvordan dataene i denne strukturen skal lagres må man vite hvilke data disse filmstrukturene faktisk skal inneholde.

3.2.1 Krav til strukturen

Filmstrukturene skal inneholde informasjon om alle filmklipp som potensielt er en del av den aktuelle filmvisningen. Den skal inneholde informasjon om hvilke valg som tilhører hvilke filmklipp. Valgene har flere andre attributter, som når de skal begynne å vises, hvor lenge de skal vises og valgfri tekst som vil vises over valgalternativene i filmavspilleren. Hvert valg har alternativer, med tekst, som peker til andre filmklipp, eller til eksterne nettsider. Det skal også være mulig å sette ett valg til å være standardvalg, og i tillegg usynlig.

3.2.2 Høstens struktur

Vår tanke var hele tiden å lagre denne strukturen i XML-format. XML er et standardisert datalagringsformat. Vi kunne ha lagd vårt eget format, men vi mente det ville være ryddigere å benytte seg av eksisterende teknologi.

Selv om vi hadde valgt å lagre strukturene i XML-format var det fortsatt flere muligheter for hvordan strukturene kunne lagres. I utgangspunktet hadde vi i prosjektet som ble gjennomført høstsemesteret 2006, tenkt å lagre strukturen som et tre, hvor alle noder kunne ha andre noder som barn av seg. Et tre er en type graf hvor nodene har bare en *parent*[2].

Hver node ville inneholde et filmklipp. Vår tanke var at vi burde lagre hvert valgs etterfølgende historier inne i det aktuelle valget, i den aktuelle noden. På denne måten ville alle noder ligge inne i *root*-noden.

Denne strukturen er vanskelig å få oversikt over om man prøver å lese XML-filen, og det ligger mye data i den. Vi følte likevel at strukturen var riktig, siden den minnet mest om slik vi så for oss at strukturen ville bli representert i filmavspilleren[6], etter at strukturen er lest fra fil.

Denne XML-strukturen ble videreutviklet under vårsemesterets hovedprosjekt, for å legge tilrette for ny funksjonalitet.

3.2.3 Vårens struktur

Etter samtale med veileder gikk det riktignok opp for gruppen at å fysisk legge nodene inni hverandre i XML-filen, slik vi hadde tenkt, var en dårlig løsning. Grafer lagres som regel som lister med noder, med pekere i nodene som peker til andre noder i listen. Etter denne samtalen med veileder så vi ingen grunn til at vår struktur skulle være noe annerledes, spesielt med tanke på at et av de nye kravene til systemet var at det skulle være mulig å linke til andre noder i strukturen. Det ville vært helt meningsløst å beholde trestrukturen om det likevel skulle gå an å linke til andre noder i treet. Derfor ble XML-strukturen gjort om til en liste med noder etter hverandre, med unike id'er som attributter, hvor hver link, i valgene, refererer til en nodes id.

Figur 3.2 på neste side viser et eksempel på en slik struktur. I strukturen ligger data lagret som beskriver hvilke filmklipp som tilhører hvilke noder, hvor mange valg som tilhører hver node, når og hvor lenge valgene skal vises, og om tekst skal vises til valget. Det ligger også informasjon om valgenes alternativer, med alternativenes tekst, link til en annen node eller eventuelt link til ekstern nettside. Når filmavspilleren har spilt ferdig filmklippet til en node, uten valg, er filmvisningen ferdig.

I strukturen ligger filmklippene bare lagret som nummer (i `media`-elementenes `mediaid`-attributter). Dette er fordi informasjon som filbane og lignende til filmfilene ligger lagret i en database, og nummeret er referanser til det aktuelle filmklippet. Grunnen til dette er at det skal være mulig å endre informasjon til et filmklipp uten å måtte redigere alle filmstrukturer som har brukt det aktuelle filmklippet. Grunnen til at XML-elementet heter `media` er fordi det tas høyde for at det i senere versjoner av systemet kan det være mulig å importere andre elementer enn bare filmklipp.

Siden disse strukturene inneholder informasjon om hva som skal skje når et valg skal vises og hva som skal skje om et valg blir tatt, vil det som tidligere er

[6] Mer om hvordan strukturen er representert i underkapitell 4.4 på side 41.

```
<story author='Jon Vatne' title='Eksemplarisk historie.'>
  <node xml:id="n1">
    <media mediaid="666"/>
    <branch duration="20">
      <text>Hva vil du?</text>
      <link href="node://n2" default="true">Jeg vil se et
filmklipp</link>
      <link href="http://www.hig.no">Jeg vil til en nettside
</link>
    </branch>
  </node>
  <node xml:id="n2">
    <media mediaid="667"/>
    <branch start="-120" duration="10">
   <link href="node://n4">Stopp filmklippet jeg ser nå og gå
videre til filmnode n4</link>
       <link href="http://www.google.com">Sjekk ut google</link>
   </branch>
   <branch duration="20">
     <text>Hvordan skal det siste filmklippet du ser være?</text>
     <link href="node://n3">Det skal være fullspekket med action
</link>
     <link href="node://n4">Polsk fjernsynsteater!
</link>
     <link href="node://n5" default="true" invisible="true">Dette
valget er usynlig og satt til å være standardvalg</link>
   </branch>
  </node>
  <node xml:id="n3">
    <media mediaid="999"/>
  </node>
  <node xml:id="n4">
    <media mediaid="999"/>
  </node>
  <node xml:id="n5">
    <media mediaid="2007"/>
  </node>
</story>
```

Figur 3.2: Eksempel på en valgraf representert i XML-format

omtalt som strukturer, nå bli omtalt som *valg-grafer*, *grafer*, eller *option graphs*. Dette navnet kommer av at vi fra tidlig i prosjektperioden, høstsemesteret 2006, brukte navnet *valgtrær*.

3.3 Databasens arkitektur

Databasens arkitektur ble tidlig utviklet ut fra erfaringer og ideer fra høstsemesterets prosjekt. Riktignok ble det lagt til en god del, og noe ble forandret, men strukturen er i hovedsak den samme som vi først så for oss. Figur 3.3 viser databasens fire tabeller: *movieclips*, *optiongraphs*, *uploadpasses*, *users*.

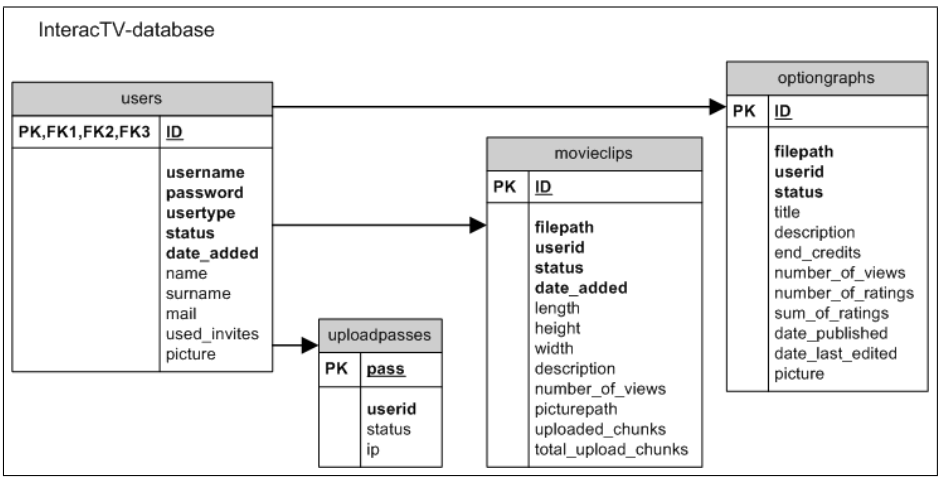
Databasen er lagd ved hjelp av MySQL.

movieclips : Tar vare på informasjon om hvert enkelt filmklipp, som for eksempel filbane, høyde og bredde i piksler, lengde i sekunder etc.

optiongraphs : Inneholder informasjon om valg-grafene: Filbane, beskrivelse, rulletekst, dato for når grafen sist ble redigert og publisert, informasjon om hvor mange som har sett den aktuelle filmvisningen, og hvilke karakterer mottakerne eventuelt har gitt den.

uploadpasses : En tabell som brukes for å verifisere brukere som laster opp filmklipp. Mer om dette i underkapittelet 4.1 på side 21.

users : Inneholder data om systemets forskjellige brukere. Her ligger blant annet informasjon om brukernavn, fornavn, etternavn, kryptert passord, epostadresse, status (definerer om brukeren er vanlig bruker, moderator, administrator, eller slettet).



Figur 3.3: ER-diagram av databasen

Kapittel 4

Utvikling

Vi har valgt å skrive om de forskjellige hovedprosessene i systemet i den rekkefølgen som de ble utviklet i dette prosjektet.

Rekkefølgen kan du se i den reviderte framdriftsplanen i vedlegg C.2 på side 71.

4.1 Opplasting

Det skal være enkelt for brukeren å laste opp sine filmklipp. Derfor var det ønskelig med en opplastingsprosessen skulle være implementert i nettløsningen, fremfor for eksempel en FTP-opplastningsløsning som krever mer innsats og kunnskap av brukeren.

Nettapplikasjonsprogrammeringsspråk som PHP er i utgangspunktet ikke egnet til bruk av opplastning av store filer. Som standard kan for eksempel ikke PHP laste opp større filer enn 2MB [3]. Denne verdien kan riktignok endres, men språket prosesserer uansett hele filen som blir lastes opp, og anbefales derfor ikke å brukes om filene kan overskride 10MB. Skal det være mulig å laste opp filmfiler vil filstørrelsene mest sannsynlig overskride 10MB. Man må ta høyde for at brukerne ikke tenker på komprimering av filmen, og for eksempel prøver å laste opp filmfiler med DV-kvalitet. Ett minutt med film i DV-kvalitet tar, til sammenligning, over 100MB [4].

Under høstsemesterets prosjekt kom vi fram til at den beste løsningen ville være å lage en applet som lar brukeren velge en fil lokalt på klientmaskinen. Applet'en vil så sende filen til en appliksjon på tjenersiden, som tar imot og behandler filen.

Vi fikk aldri realisert disse ideen i høstsemesteret, og det var heller ikke klart for oss hvordan selve filoverføringen skulle gjøres. Vi hadde en idé om at filen som skal lastes opp, blir lest inn av appleten, stykket opp i «chunks», og at hver chunk blir sendt én etter én til en applikasjon på tjenersiden som setter sammen chunkene til en kopi av filen på klientmaskinen.

Ett av målene for vårsemesterets hovedprosjekt, var å skape denne opplastningsprosessen - dermed måtte ideene realiseres. Resten av dette underkapittelet beskriver utviklingsprosessen og valg som ble tatt underveis.

4.1.1 Klientsiden

Språkvalg

Det mest naturlige valget for hvilken form opplastningsappleten skulle ha, var en Java Applet. Siden måten Java-språket ble sett på som å ha mange tilgjengelige biblioteker og være utbredt og godt dokumentert. Ikke minst ble det ansett som svært sansynlig at de fleste av systemets brukere vil ha støtte for å kjøre JavaApplets på sin klientmaskin. Selv om gruppen ikke har erfaring med Java-programmering ble Java likevel valgt som språk for opplastningsappleten.

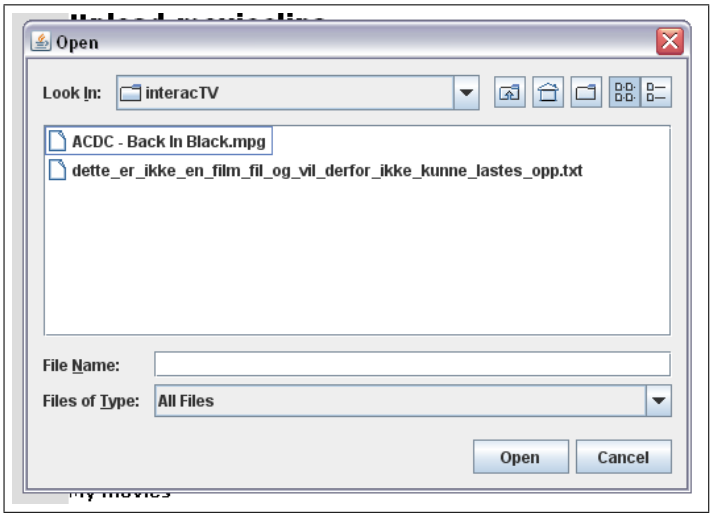
En løsning som ble ansett som bedre, med tanke på kompabilitet og kontinuitet i systemet, var at opplastningsappleten ble lagd i ActionScript og Flash. Filmavspilleren har hele tiden vært ment på denne måten, noe som vil si at alle systemets brukere og mottakere må ha støtte for Flash for å kunne se de interaktive filmene. En filopplastningsløsning i Flash ble riktignok forkastet, dette på grunn av at gruppen bygget seg opp en oppfatning under høstsemesterets prosjekt om at ActionScript har en del begrensninger og er vanskelig å jobbe med.

Prosesser

For enkelhets skyld ville applet'en i første omgang ha tre oppgaver: La brukeren velge en fil, lese inn filen i programmet, og laste opp filen i chunks til tjenereren.

Velge fil

Siden gruppen hadde svært liten erfaring med Java-programmering, gikk det med en del timer til å lære hvordan enkle Java Applets utvikles. Når det første «Hello world»-programmet hadde blitt unnagjort kunne vi gå videre til å lage prosessen for velging av fil. Siden det legges vekt på at denne delen av systemet skal være enkel å bruke, var det klart at et GUI for velging av lokale filer måtte brukes. Etter en del leting og noe testing falt til slutt valget på `JFileChooser`[1] (vist i figur 4.1), som er arvet fra Javabibliotekets `javax.swing`-klasse.



Figur 4.1: JFileChooser-klassens GUI

Sertifisering

For at applet'en skal ha tilgang til lokale filer, viste det seg at den må ha et sertifikat. Dette sertifikatet ble lagd ved hjelp av *keytool*, som følger med Suns utviklingsverktøy. Seritfikatet ble så koblet til appleten ved hjelp av *jarsigner*, som også følger med Suns utviklingsverkøy. At dette i det hele tatt måtte gjøres, og ikke minst hvordan det måtte gjøres, leste vi i én av de mange manualene vi fant på internett [5]. Denne typen sertifikat kan riktignok ikke verifiseres av nettleseren, og må aksepteres av brukeren for at appleten skal kjøres korrekt.

[1] JFileChooser: http://java.sun.com/j2se/1.4.2/docs/api/javax/swing/JFileChooser.html

Lese fil

Når brukeren altså har valgt en fil, må noe skje med denne filen. Applet'en sjekker MIME-typen og filendingen til filen for å se om filen faktisk er en filmfil. Kjenner appleten igjen filen som en filmfil begynner den å lese inn filen. Da vi testet denne funksjonen fant vi ut at nettleseren låste seg hvis vi valgte veldig store filer. Dette er fordi nettleseren ikke har nok tilgjengelig minne til å lese inn hele filen inn i appleten, om filen er for stor. Vi løste dette ved å lese filen inn i chunks på maksimum 25MB, som forhåpentligvis er små nok porsjoner til at de får plass i minnet. Et valg måtte tas på hvor store disse chunkene skulle være; ble de satt til å være for små, måtte appleten jobbe mer og den ville blitt tregere, ble de satt til å være for store ville appleten låse seg. Når en chunk er lest inn, prosesseres binærdataene i chunken (de lastes opp til tjeneren), og neste chunk leses inn. På denne måten jobber appleten seg gjennom hele filen.

Hver av de innleste chunkene blir så stykket opp ytterligere i mindre chunks for opplasting. Disse opplastnings-chunkene ble satt til å være maksimum 256Kb, for at tjeneren ikke skulle få for mye data å håndtere.

Filoverføring

Når vi hadde fått til å lese en lokal fil og stykke den opp i små nok chunks til å sendes, begynte vi med tjenerkommunikasjonen. I første omgang sendte vi `POST`-data til en PHP-fil som skrev disse til en tekstfil på tjeneren. Dataene som skulle sendes var `total_size` (filmfilens totale størrelse), `chunk_nr` (hvilken opplastningschunk som lastes opp nå), og `uploadpass` (brukes til brukervalidering på tjeneren). Disse verdiene brukes av applikasjonen på tjenersiden for å finne ut hvor langt opplastningen er kommet og hvem som laster opp. Når vi hadde bekreftet at dette virket var det på tide å sende både tekstdata og binærdata.

Vi hadde mye problemer med å finne en måte å laste opp binærdataene slik vi hadde tenkt oss, altså i `FILES`-variabelen gjennom HTTP-protokollen. Vi fant ingen eksempler eller relevant fagstoff om dette, og alternative metoder ble forsøkt. Vi brukte en del timer på å enkode chunkenes binærdata til tekststrenger, for så å dekode de på tjenersiden. Heldigvis kom vi over en klasse en av utviklerne i Sun hadde publisert [6], som gjorde akkurat det vi var ute etter.

Denne klassen var godt dokumentert og virket perfekt, og ble derfor implementert og brukt i applet'en. I motsetning til de andre eksemplene vi fant om dette emnet, definerte denne klassen mer detaljert hvordan fil-headeren skulle sendes til tjeneren. `POST`-variabler sendes slik:

```
--boundary\r\n
Content-Disposition: form-data; name="<fieldName>"\r\n
\r\n
<value>\r\n
```

Og `FILES`-variabler sendes slik:

```
--boundary\r\n
Content-Disposition: form-data; name="<fieldName>";
filename="<filename>"\r\n
Content-Type: <mime-type>\r\n
```

```
\r\n
<file-data>\r\n
```

`boundary` er en tekststreng som består av 20 bindestreker (-) og en heksadesimal representasjon av nåtiden, i millisekunder, og fungerer som en grense som skiller mellom variablene som sendes. `<fieldName>` er navnet til variabelen `POST`- eller `FILES`-variabelen vil ligge i, på tjeneren. `<value>` er `POST`-variablens verdi. `<filename>` er orginalnavnet til filen. `<mime-type>` er filens MIME-type og `<file-data>` er binærdataene.

4.1.2 Tjenersiden

Språkvalg

I utgangspunktet mente vi at systemet på tjenersiden, for å ta imot og håndtere filopplastingene fra klientsiden, burde være lagd med JSP. Grunnen var blant annet at vi tenkte at siden appleten på klientsiden ville være lagd i Java, burde også systemet på tjenersiden benytte seg av Java-språket for at kommunikasjonen skulle fungere. Etter samtaler med veileder ble vi riktignok gjort oppmerksom på at dette ikke var faktum. Det ble likevel ansett som enklere å ha samme språk på tjener- og klientsiden, blant annet fordi det slik ble sett på som enklere å finne fagstoff om slike løsninger.

Etter samtale med Øivind Kolloen[2] skiftet vi riktignok synet på hvor vanskelig en løsning med et annet programmeringsspråk på tjenersiden, ville være. Det ble slått fast at så lenge klientapplikasjonen for eksempel sendte data i `POST`- og `FILES`-variabler via http-protokollen, ville det være likegyldig hvilket programmeringsspråk som ble benyttet på klient- og tjenersiden.

JSP ble ansett som en bedre løsning med tanke på ytelse og håndtering av `SESSION`-variabler i forbindelse med brukervalidering på kryss av tjenermaskiner. PHP ble likevel valgt som språk på tjeneren grunnet flere faktorer: JSP ble ansett som mye mer tungvindt enn PHP, både fordi gruppen har mer erfaring med sistnevnte, fordi vår løsning ville krevd at vi måtte kompilert «javabønner» manuelt (som tar mye tid), fordi JSP generelt sees på som et veldig komplekst utviklingsmiljø, og fordi fordelene med JSP ikke ble ansett som kritiske nok for systemet.

På grunnlag av blant annet dette ble det tatt et valg om at PHP skulle brukes som språk i tjenerapplikasjonen som tar imot fil-chunks fra klienten. Hovedgrunnen til dette var riktignok at PHP blir brukt så mye som mulig ellers i systemet, se for eksempel underkapittel 4.3 på side 33. Å forhindre mange forskjellige språk i systemet er viktig med tanke på å forhindre feil og unødvendig tidsbruk under utvikling.

Simulering

Det første som ble gjort i forbindelse med filopplastingen på tjenersiden, var å kopiere en fil fra én mappe til en annen, på tjenermaskinen. Denne kopieringen skjedde på samme måte som det har blitt forespeilet at opplastningen skulle virke, nemlig at filen blir stykket opp i chunks og at hver av disse blir skrevet

[2] Øivind Kolloen, Høgskolelærer ved HiG

til slutten av en annen, ny fil. På denne måten ble filen kopiert i mange operasjoner og i små porsjoner.

Oppgaver

Når simuleringen av opplastingen var gjort, var det på tide å faktisk bruke de dataene som blir sendt fra klienten. Som tidligere nevnt, sender klienten blant annet filens totale størrelse i bytes (`total_size`) og hvilken chunk som lastes opp nå (`chunk_nr`), som `POST`-variabler.

Forutsatt at både klientapplikasjonen og tjenerapplikasjon opererer med samme verdi på opplastnings-chunk-størrelse, er det ut i fra disse variablene mulig å finne ut om chunken er: a) Opplastningens første chunk. b) Opplastningens siste chunk c) En av de mange chunkene i mellomtiden.

Brukervalidering

For at applikasjonen skal kunne behandle de opplastede data på fornuftig vis, gjenstår det først å finne ut hvilken bruker det er som laster opp filen. Siden filopplastningen skjer til en annen tjenermaskin enn den webserveren ligger på, er det ikke mulig å sjekke om brukeren er logget inn ved å bruke `SESSION`-variabler, siden disse ligger på den andre tjenermaskinen. Derfor ble en alternativ måte for brukergjenkjenning utviklet.

Med hver chunk sendes `POST`-variabelen `uploadpass`; denne variabelen har blitt sendt med opplastnings-applet'en som en parameter, og fungerer som en nøkkel til opplastningen. Nøkkelen brukes for å identifisere hvilken bruker som laster opp filen. Ved å kjøre en spørring, mot *uploadpasses*-tabellen i databasen, finner tjenerenapplikasjonen brukerens ID i *users*-tabellen. Brukerens ID er nå kjent for applikasjonen, og filopplastningsmappen blir så satt til å være orginalbanen, pluss en mappe med samme navn som brukerid'en (*uploading*/`brukerid`/).

Første chunk

Om chunken er den første chunken av en fil tjenerapplikasjonen tar imot, betyr det at en ny opplastning har begynt. I så fall blir data om filen lagt til i databasens *movieclips*-tabell[3]. Database-id'en til brukeren (`userid`), som har lastet opp filen, er den viktigste informasjonen som blir registrert. I tillegg blir filmklippets `status` satt til at filen er i ferd med å lastes opp (`0`), filstørrelsen (`size`) blir satt til å være lik `POST`-varibalen `total size` og når filen blir registrert (nå) legges den til i `date_added`. Filmklippets filbane (opplastningsmappen pluss orginalt filnavn) blir også lagd til i `filepath`. Id'en til raden som nettopp ble lagt til, blir så hentet ut.

Deretter blir det opprettet en fil med den filbanen som nettopp ble registrert i databasen, som chunkene kan skrives til. Det blir også opprettet en metadatafil i samme mappe som inneholder database-id'en til filmklippfilen (`clip_id`) og database-id'en til brukeren (`user_id`). Denne metadatafilen er en tekstfil, som heter det samme som filmklippfilen, bare med en annen filending (.fvm). Denne filen vil senere bli brukt av blant annet transkodingsprosessen til å koble klippet til en bruker.

[3]Oversikt over databasens tabeller finner man under underkapittelet 3.3 på side 19

Deretter skrives binærdataene i chunken til den nylig opprettede filen, og raden som nettopp ble lagt i databasens *movieclips*-tabell, oppdateres med hvor mange chunks som er lastet opp.

Chunkene i mellomtiden

Om chunken verken er den første chunken eller den siste chunken, skal det ikke skje annet enn at chunkens binærdata blir skrevet til slutten av filen, som er i ferd med å bli bygget opp på tjenermaskinen, og at filens rad i databasens *movieclips*-tabellen blir oppdatert med hvor mange chunks som hittil er lastet opp.

Før databasen kan oppdateres, må riktignok applikasjonen vite hva filmfilens id er i *movieclips*-tabellen. Denne id'en leses fra metadatafilen som ble opprettet da den første chunken ble tatt imot. Applikasjonen vet metadatafilens filbane, siden den vet filmfilens filbane (*uploading*/`brukerid/filnavn`).

Siste chunk

Om chunken er den siste chunken av filen som lastes opp, skrives først binærdataene til filen, og filmklippets database-id leses fra metadatafilen, slik som med de tidligere chunkene. Deretter flyttes filmfilen og filmfilens metadatafil til mappen *uploaded*, for ferdig opplastede filmklipp. Deretter oppdaterer filmklippets rad i databasen, slik at radens `status` settes til å være ferdig opplastet (`1`) og filbanen (`filepath`) settes til å være der filen ligger i *uploaded*-mappen, i tillegg til at antall opplastede chunks (`uploaded_chunks`) blir oppdatert.

Når dette er gjort er filen ferdig lastet opp, og det er opp til resten av systemet å behandle den opplastede filen.

4.1.3 Implementering i nettsidene

Mer om brukervalidering

Når opplastnings-applet'en blir vist på nettsidene blir også `uploadpass`-parameteren sendt med til appleten. Under er et eksempel på hvordan HTML-koden, for implementeringen av appleten, kan se ut:

```
<APPLET CODE="LastOpp.class" archive="LastOpp.jar"
  codebase="http://128.39.80.94/" WIDTH=438 HEIGHT=300>
  <param name=uploadpass value="0.05622458695835">
</APPLET>
```

Parameteren `codebase` må legges til for at nettleseren skal forstå at appleten skal kjøres fra en annen tjenermaskin (transkodingsmaskinen), enn den webserveren kjører på. Appleten må også lastes ned fra samme maskin som den skal kommunisere med tjenerapplikasjonen på [7].

Verdien som legges i parameteren med navn `uploadpass` hentes fra tabellen *userpasses* i databasen. Under er SQL-spørringen som kjøres for å finne denne nøkkelen. Spørsmålstegnet (`?`) byttes ut med den innloggede brukerens ID.

```
SELECT pass FROM uploadpasses
WHERE userid =? ORDER BY status LIMIT 1
```

Hvis ingen av radene i tabellen refererer til den aktuelle brukeren, opprettes en rad med brukerens id, og en tilfeldig generert nøkkel (`pass`). Deretter hentes denne nøkkelen ut, og sendes med appleten.

Statusvisning i nettsidene

Når brukeren er logget inn, vil systemet vise hvor mye som er lastet opp av filen. Systemet finner ut at filen er i ferd med å lastes opp, ved å sjekke at `status`-feltet er `0` i det aktuelle klippets rad i *movieclips*-tabellen. Systemet finner også ut hvor mange prosent av filen som er lastet opp ved å bruke `total_size`- og `chunk_nr`-verdiene til filmklippets rad i *movieclips*-tabellen.

4.1.4 Forbedringer

Klientsiden

- I skrivende stund viser applet'en en god del debug-informasjon. Det er ønskelig å ikke vise denne, men heller vise en enkel «progress bar». Vi ser for oss at den ideelt sett bør være ganske liten og nøytral, og de eneste tilbakemeldingene som strengt tatt må vises for brukeren er hvor mye av filen som er lastes opp og en melding om at filmen er ferdig opplastet.
- Når en bruker har klippet til potensielt flere titalls filmklipp og skal laste opp disse, vil det være veldig tungvint å velge én fil som skal lastes opp, vente til den er ferdig lastes opp, for så å velge og laste opp neste fil.

  Det vil derfor være ønskelig å kunne velge filer for så å sette disse i en kø, som lastes opp fortløpende én og én. Når denne funksjonaliteten er på plass bør det også være mulig å fjerne filer fra køen og endre rekkefølgen på køen.

Vi ser ikke på noen av disse forbedringen som alt for kompliserte, men på grunn av lite kompetanse innen Java og liten tid, har dette blitt nedprioritert i dette prosjektet.

Tjenersiden

- Det kjøres ingen sjekk på at filene som lagres på tjeneren faktisk er like de som ble valgt på klientmaskinen for å lastes opp. Og det er naivt å tro at ingenting galt kan skje med *noen* av fil-chunkene under opplasting. For eksempel kan nettforbindelsen til klienten forsvinne.

  En mulig løsning på dette problemet kan være å sende med en MD5-hash-streng av alle binærdataene i en chunk, sammen med chunken. Da kan tjenerapplikasjonen selv lage MD5-hash ut fra binærdataene den har fått tilsendt, og er MD5-strengene like, er chunkene mest sansynlig like. Dette kan også gjøres for eksempel innlesningschunkene i appleten (25MB).

  Det burde også sjekkes om det er den riktige chunken som har blitt lastes opp. Dette kan gjøres ved å sjekke om `chunk_nr` stemmer overens med den nåværende størrelsen til filen det lastes opp til.

Skulle det vise seg at en chunk ikke er lik den på tjeneren som den på klienten, eller at en chunk ikke har kommet frem, bør det også være et system for å laste den aktuelle chunken opp på nytt.

Nettsidene

- Slik opplastningsstatusen vises for brukeren nå, blir den ikke oppdatert fortløpende - brukeren må oppdatere nettsiden for å se en ny prosentverdi for hvor mye som er lastet opp av filen. En videreutvikling av dette ville være å bruke for eksempel AJAX til å fortløpende oppdatere denne verdien.
- Opplastningsnøkler, som blir sendt med som parametere til opplastningsappleten, bør automatisk slettes med jevne mellomrom for å forhindre at gamle nøkler kan bli misbrukt til å få en opplastning validert.

Ingen av disse manglene ble sett på som kritiske nok til å måtte gjøres noe med i dette prosjektet.

4.1.5 Andre kjente mangler

- Det kontrolleres ikke at det faktisk er den aktuelle brukeren som laster opp filmklippet. Om en person har nok kjennskap til systemet, og kjennskap til å lage HTML-sider, og har tilgang til en opplastingsnøkkel, er det fullt mulig å laste opp en fil til systemet som en av systemets registrerte brukere, uten å faktisk være innlogget.

  Dette sikkerhetshullet blir riktignok ikke sett på som særlig stort, siden den eneste skaden det kan gjøre er at noen filmklipp som brukeren egentlig ikke har lastet opp, er tilgjengelige for brukeren til å bruke i interaktive filmer. Det blir heller sett på som en *feature* fremfor en *bug*.

  Det eneste gruppen ser på som et problem, er om to eller flere personer, med samme opplastingsnøkkel, skulle laste opp filer med samme filnavn samtidig. Dette blir riktignok sett på som svært usannsynlig, og er derfor ikke tatt til følge.

4.2 Transkoding

For at systemet skal være så enkelt som mulig å bruke, bør ikke brukeren være påkrevd å konvertere filmfilene til et spesielt format før han laster de opp. Riktignok må filmklippene være enkodet i et spesielt format for at filmavspilleren, som utvikles i Flash, skal kunne spille de av. Dette filformatet er FLV[4].

Dette vil si at filmfilene må konverteres til FLV, automatisk fra det formatet brukeren har lastet opp, på tjenermaskinen. En god løsning, med tanke på at det ikke har blitt stilt alt for store krav til ytelsen av systemet ennå, ble utviklet under høstsemesterets prosjekt. Det ble utviklet et shell-skript som kort sagt sjekker etter filer i en gitt mappe, transkoder alle filmfilene i mappen, og flytter de til en ny mappe.

I neste underkapittel kommer en beskrivelse av hvordan denne prosessen fungerer. Beskrivelsen er ikke alt for detaljert, siden det ikke har vært gruppens fokus under hovedprosjektet å endre på det som allerede virket som det skulle.

4.2.1 Hvordan det virket

I realiteten består transkodingsprosessen av to shell, som begge er skrevet i BASH. Det ene av disse, *nisse.sh*, har som oppgave å sjekke om det har kommet filmfiler i en gitt mappe (*uploaded*). Finner det filer, flyttes alle disse i en ny mappe (*working*). Og for hver fil som har en tilhørende metadatafil, kjøres det andre shellet, *flvmagic.sh*, med blant annet filnavnet som parameter.

nisse finner ut hvor den skal flytte filene etter transkoding, ut fra informasjonen i metadatafilene.

flvmagic har som oppgave å faktisk transkode en gitt filmfil og flytte den til en gitt mappe. Dette skjer ved at shellet kjører et Linux-program, *ffmpeg*, som ved hjelp av en del parametere kan konvertere de fleste filmfiler til FLV-format:

```
ffmpeg -i $workdir$originalfile -b 600k -r 25 -ar 44100
-acodec mp3 -ab 128 $targetdir$newfile
```

Her blir filmen transkodet til en FLV-fil med 25 bilder per sekund, 600 kbps bitrate og lyden blir satt til å være MP3 med 128 kbps bitfrekvens og 44,1 KHz samplingsfrekvens. Disse parameterne prøvde vi oss fram til under høstsemesterets prosjekt, og ble sett på som et godt kompromiss mellom filstørrelse og kvalitet. Det ble derfor ikke sett på som nødvendig å endre disse parameterne under vårsemesterets hovedprosjekt.

Etter at *flvmagic* har transkodet filen, henter det ut en del metadata om filmfilen. Disse dataene blir tilgjengelig ved at *flvmagic* kjører Linux-programmet *mplayer*, med en del parametere som tilsier at *mplayer* skal skrive ut en del metadata om filen til skjerm. Slik ser denne kommandoen ut:

```
mplayer -identify $targetdir$newfile -nosound -vc dummy
        -vo null -msglevel all=-1 | grep "ID_" > $targetdir$metafile
```

Disse blir så fisket tak i av *flvmagic* ved hjelp av *grep*-funksjonen. *flvmagic* genererer også et «thumbnail-bilde» av filmen ved å kjøre *mplayer* med noen spesielle parametere.

[4] Flash Video

Etter at *flvmagic* har gjort jobben sin, kjører *nisse* på nytt *flvmagic*, denne gangen med neste fils filbaneparametere. Når alle filene i mappen er gått gjennom på denne måten er *nisse* ferdigkjørt, og klar til å kjøres på nytt.

nisse blir automatisk kjørt, om det ikke allerede kjører, av en cron-job. Cron-jobben er satt til å kjøres hvert minutt (det minste tidsintervalet en cron-job kan ha), og det eneste den gjør er å sette igang *nisse*.

Cron er en tidsbasert planlegginsfunksjonn som finnes i Unix-baserte systemer. Den drives av konfigurasjonsfilen *crontab* som kjører kjører shell-kommandoer på gitte tidspunkter ut ifra en definert tidsplan. Vår *crontab*-fil ser slik ut:

```
* * * * * /usr/local/bin/nisse.sh > /tmp/output 2>&1
```

4.2.2 Hvordan virker det

Under vårsemesterets hovedprosjekt ble det satt et nytt krav til transkodingsprosessen. Dette kravet gikk ut på at *nisse* og *flvmagic* skulle kommunisere mer med databasen, slik at det er mulig å hente ut informasjon om hvilke filer som er oppdaget av *nisse* og satt i transkodingskø, hvilke som transkodes i dette øyeblikk, og hvilke som er ferdig transkodet.

Det bør nevnes at hovedprosjektgruppen har svært lite kunnskap til Linux og BASH, og at endringene i shellene derfor ikke var noen intuitiv og særlig enkel oppgave.

Første databasekommunikasjon

Den første databasekommunikasjon som skulle gjøres var, når nye filmfiler hadde blitt oppdaget og satt i kø av *nisse*. Dette shellet gikk i utgangspunktet gjennom filene én etter én på en måte som gjorde at når det tilkaller *flvmagic*, har den ikke mulighet til å vite hvor mange flere filer i mappen som skal transkodes. På grunn av BASH-språkets begrensninger og gruppens mangel på erfaring i dette språket, ble løsningen en ganske enkel og ufin en. Før løkken hvor *flvmagic* tilkalles, kjøres først en løkke som kjører på akkurat samme betingelser som den tidligere nevnte løkken. Innmaten i denne løkken er derimot en helt annen. Det denne løkken gjør er nemlig å endre dataene i de aktuelle radene i databasens *movieclips*-tabell.

Hvordan databasekommunikasjon burde skje ved hjelp av BASH-språket, var det riktignok ingen innlysende løsning på. En del metoder ble prøvd ut, og etter en del leting og testing, kom vi fram til følgende metode:

1. Databasespørringen blir skrevet til en midlertidig fil.
2. En MySQL-kommando i Linux blir kjørt med den midlertidige filens filbane som parameter.

På denne måten blir SQL-spørringen, skrevet til filen, kjørt mot databasen. Det databasespørringen gjør, er å sette filmklippets `status` til å være `2` (oppdaget og satt i kø av transkodingsskriptet) og filbanen til å være filens nye filbane (*.../working/*). Under er et utdrag av innholdet i den aktuelle løkken:

```
echo "UPDATE movieclips SET status=2, filepath='$workdir$filename'
      WHERE ID=$clip_id LIMIT 1\g" >> $tempfil
mysql --host=$MYSQLSERVER $MYSQLDBNAME --user=$MYSQLUSER
      --password=$MYSQLPASSWORD < $tempfil
```

Andre databasekommunikasjon

Den neste databasekommunikasjonen som skal skje, er at databasen skal få beskjed om at filen er i ferd med å transkodes. Denne spørringen kjøres rett før *flvmagic* blir tilkalt i den andre løkken (som tidligere er nevnt), og kan sees på som den «opprinnelige» løkken.

Spørringen blir kjørt på samme måte som forrige; det eneste den gjør er å sette `status`, i det aktuelle filmklippets rad, i *movieclips*-tabellen i databasen, til å være `3`, som vil si at filmklippet transkodes nå.

Tredje databasekommunikasjon

Databasen må også oppdateres når filmklippet er ferdig transkodet til FLV-format. Denne spørringen blir gjort inne i *flvmagic*, og kjøres etter at filen er flyttet til webserver-tjenermaskinen.

Spørringen setter det aktuelle filmklippets rad, i *movieclips*-tabellen i databasen, sin `status` til å være `4` (ferdig transkodet og klar til bruk), og filbanen blir satt til å være den nye filens filbane på webserver-maskinen. I tillegg legges det til data om lengden på filmklippet i sekunder, høyden og bredden på klippet i piksler og filbanen til en thumbnail-fil som *flvmagic* har generert. Spørringen kjøres på samme vis som de forrige, og ser slik ut:

```
UPDATE movieclips SET status=4,
       filepath='$targetdir$newfile',
       length='$cliplength',
       height='$clipheight',
       width='$clipwidth',
       picturepath='$targetdir$thumbsfolder$thumbnailfile'
WHERE ID=$5
LIMIT 1"
```

4.2.3 Ytterligere konfigureringer

Bortsett fra det som allerede er nevnt ble det i tillegg endret noen småting for at transkodingsprosessen skulle virke på riktig måte.

Det ble for eksempel endret en del i koden, siden *flvmagic* i utgangspunktet ikke kunne opprette mapper på webserver-maskinen. At *flvmagic* skal ha mulighet til dette, er en nødvendighet med tanke på at nye brukere ikke nødvendigvis har fått opprettet en egen mappe for filmfiler på webserver-maskinen. Det ble brukt noen timer, spesielt på å finne hva problemet var, og hvordan det skulle løses.

For at *ffmpeg* skulle virke korrekt, ble også en god del uønskede tegn fjernet fra filnavnene til filene som skulle transkodes.

4.2.4 Forbedringer

- Det er tydelig at det hadde vært fordelaktig å lage hele transkodingsprossen i et kraftigere språk enn BASH, med tanke på hvor «hårete» løkkene og databasespørringene har blitt gjort i den nåverende løsningen. Python er det alternativet som i skrivende stund virker mest interessant.

- Shellene bør også kjøre flere sjekker på om det som faktisk var ment å gjøres faktisk har blitt gjort. Det bør altså kjøres flere sjekker på feilsituasjoner som kan oppstå. For eksempel bør det lages thumbnail-bilder på alternative vis om det skulle vise seg at den første metoden ikke fungerer. Det samme gjelder for selve transkodingen av filmfilen og uthenting av metadata.

4.3 Grafredigering

Grafredigering var en av prosessene i systemet som bare var i idéfasen under høstsemesterets prosjekt. Det var riktignok ingen tvil om at det burde være et nettbasert system for redigering av valg-grafene (filmstrukturene).

Som tidligere nevnt i underkapittel 3.2 på side 16, lagres valg-grafen i en XML-struktur, og disse må altså kunne redigeres fra et nettgrensesnitt.

Kravene til hva som skal kunne gjøres i grafredigeringsapplikasjonen er ganske klart definerte i kapittel 2 på side 10 og i use case-beskrivelsen i D.1.4 på side 75 (under «Ajourføring av valgtrær»). Mulighetene for hvordan vi velger å løse disse kravene er likevel mange.

Det var klart for oss at applikasjonen måtte være i stand til å lese inn XML-strukturen, redigere den etter brukerens ønsker, for så å kunne lagre den.

4.3.1 Fremgangsmåte

Idéer

Vi hadde utarbeidet oss noen tanker om hvordan systemet skulle være. Vi så blant annet for oss at brukeren skulle kunne se en grafisk oversikt over alle nodene i en graf, og ha mulighet til å legge til filmklipp eller valg eller annet, hvor som helst i grafen.

Vi hadde også et par idéer med tanke på effektiviteten til applikasjonen. For eksempel hadde vi en idé om at den aktuelle interaktive filmens valg-grafs XML-struktur skulle lagres enten som `SESSION`- eller `POST`-variabler på tjeneren. Dette var for å unngå at applikasjonen skulle trenge å skrive til fil og lese inn XML-strukturen for hver endring som skulle gjøres, og kun skulle behøve å skrive til fil når strukturen var ferdig redigert, og dermed bli mer effektiv.

En annen idé vi hadde, var at denne prosessen skulle utvikles i JSP, fordi vi mente JSP hadde god støtte for å jobbe med XML-strukturer og mulighet for å ta vare på XML-strukturen, uten nødvendigvis å skrive til fil og lese inn strukturen for hver gang.

Angrepsvinkel

Etter samtaler med veileder under vårsemesterets hovedprosjekt ble det slått fast at problemer burde løses på enklest mulig måte. I første omgang burde funksjonaliteten være på plass, deretter kunne man eventuelt forbedre måten det ble gjort på.

Dette førte til en tankegang som gikk ut på å gjøre utviklingen så enkel som mulig. Alt som ble sett på som viktig, var å få funksjonaliteten på plass, så kunne vi heller tenke på faktorer som brukervennlighet, effektivitet og ytelse senere. Denne tankegangen førte til to avgjørelser som var med på å sparke igang utviklingen av dette inkrementet:

1. XML-strukturen kunne godt leses inn og skrives til fil for hver endring som skulle gjøres. Etter samtaler med et par ressurspersoner på HiG, kom vi fram til at det uansett ikke ville ha noen kritisk virkning for systemets effektivitet, spesielt siden vi ikke hadde noen spesielle krav til dette.

2. Et annet kompromiss vi gjorde, på grunn av den nye tankegangen, var at det, for enkelhets skyld, bare burde gå an å redigere en node i grafen av gangen.

Planen ble altså å lage en applikasjon som laster inn XML-strukturen til en gitt valg-graf, viser et grensesnitt som lar brukeren redigere en gitt node, og som lagrer endringene til XML-filen for hver gang brukeren registrerer en endring. Det naturlige valget for språk var PHP, siden gruppen har best kjennskap til dette nettapplikasjonsutviklingsspråket, og fordi kravene ikke tilsa at det var nødvendig å bruke noe annet språk. At PHP brukes så mye som mulig ellers i systemet er også en viktig faktor.

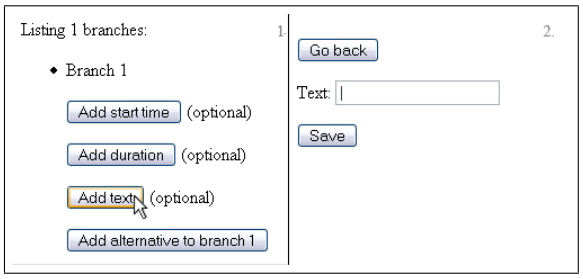
4.3.2 Hvordan det virker

Denne redigeringsapplikasjonen ble delt inn i to lag; et teknisk lag som inneholdt all funksjonaliteten som trengtes, og et presentasjonslag som viser GUI og tilkaller funksjoner i det tekniske laget.

Vi begynte med å først utvikle det tekniske laget. Dette laget ble lagt inn i en klasse med navn *GraphHandler*. Funksjonene i *GraphHandler* ble fortløpende testet under utvikling ved at funksjonskall (med test-data), ble hardkodet i koden. Når funksjonene så ut til å virke, gikk vi videre til å utvikle neste funksjon, og så videre.

Når de fleste funksjonene i det tekniske laget var på plass, begynte vi å utvikle presentasjonslaget. Dette laget ble lagt i klassen *GraphEditor*, og inneholder all GUI.

For at redigeringsapplikasjonen skulle være så enkel som mulig å utvikle, ble redigeringsmulighetene for brukeren som regel gjort slik at brukeren først trykker på en knapp for redigering av et element, så vil en ny side lastes hvor brukeren har mulighet til å gjøre ønskede endringer, og endringene vil bli utført når brukeren trykker på «Save» i dette vinduet. I dette vinduet har brukeren også mulighet til å gå tilbake til der han nettopp var ved å trykke «Back». Eksempel på en slik prosess er illustrert i figur 4.2.

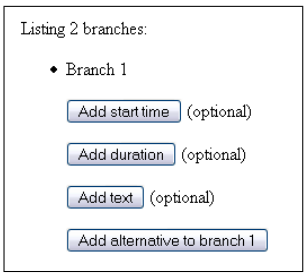


Figur 4.2: 1. Brukeren velger at han vil legge til tekst. 2.Brukeren har mulighet til å legge til tekst til valget

Legge til valg

Når brukeren er i redigeringsvisningen til en node, har han mulighet til å legge til et valg ved å trykke på «Add branch». Når et valg er lagt til, kan bruk-

eren velge å legge til **når** dette valget skal begynne å vises, ved å trykke på «Add start time», og i neste vindu skrive inn hvor mange sekunder før slutten av filmklippet valget skal begynne å vises. Brukeren har også mulighet til å bestemme hvor lenge valget skal vises, ved å trykke på «Add duration» og i neste vindu skrive inn ønsket antall sekunder. Han har også mulighet til å legge til om valget skal ha tekst som skal vises over svaralternativene, for eksempel et spørmål, ved å trykke på «Add text». Disse valgene er illustrert i figur 4.3.



Figur 4.3: Brukerens muligheter for en nodes valg

Om starttid og varighet for valget er satt, får brukeren også mulighet til å bestemme om filmen skal pauses mens valget vises. Dette er funksjonalitet som ble lagt til relativt sent i prosjektperioden, etter innspill om ønsket funksjonalitet fra medstudenter.

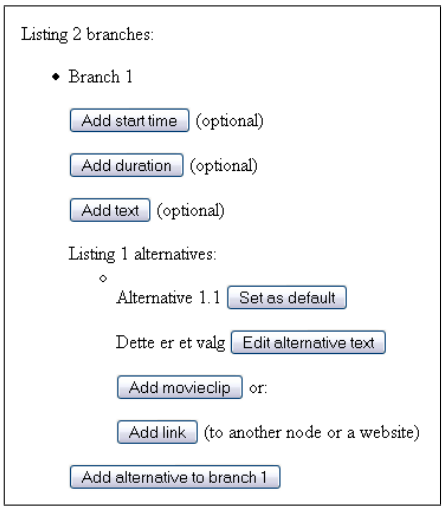
Legge til alternativ

Når brukeren har opprettet en branch, kan han legge til alternativer ved å trykke på «Add alternative to branch X». Om brukeren trykker på denne knappen vil han i neste vindu bli bedt om å skrive inn teksten dette alternativet skal ha, siden alle valgs alternativer må ha en tekst. Når brukeren så trykker «Save», vil brukeren bli sendt tilbake til noden han nettopp kom fra, men nå vil et av valgene ha et alternativ mer. Brukeren kan redigere informasjonen om alternativet, og visningen vil se omtrent ut som den vist i figur 4.4 på neste side.

Her har brukeren mulighet til å definere alternativet som standard, slik at dette valget automatisk vil bli valgt om valgets ventetiden skulle utløpe. Brukeren har også mulighet til å redigere alternativ-teksten han nettopp skrev inn, samt å legge til et filmklipp, eller legge til en link til: a) En annen node. b) En URL.

Settes et alternativ til å være standardvalget (default), vil brukeren også ha mulighet til å sette valget til å være usynlig. I så fall vil ikke dette valget vises i filmavspilleren, og den eneste måten valget kan bli vist, er om brukeren lar ventetiden løpe ut.

Velger brukeren å linke alternativet til en annen node eller en nettside, vil han komme til et vindu som vist i figur 4.5 på neste side. Her har brukeren mulighet til å velge fra en liste med noder som brukeren selv har satt tittel på, eventuelt kan han skrive inn adressen til en nettside.



Figur 4.4: Brukerens muligheter for alternativene til en nodes valg

Figur 4.5: Brukerens muligheter når han har valgt å legge til en link

Legge til filmklipp

Velger brukeren derimot å legge til et filmklipp ved å trykke på «Add movieclip» vil brukeren få en oversikt over alle filmklipp han har lastet opp som er ferdig transkodet.

Filmene vises ved hjelp av en Flash-fil vi lagde i dette øyemed. Denne lille Flash-applikasjonen henter først informasjon om filmklippet fra databasen (mer om lignende prosesser i underkapittel 4.4 på side 41), så viser det filmklippets thumbnail-bilde. Om brukeren trykker på bildet byttes bildet ut med filmfilen som begynner å spilles av. Trykke brukeren på filmen igjen, vil filmen pauses.

Under hvert filmklipp kan brukeren trykke på en knapp med teksten «Add movieclip». Trykker brukeren på en av disse klippene vil flere ting skje. En ny node vil bli opprettet i grafen og den nye nodens filmklippverdi blir satt til å være database-id'en til det filmklippet som nettopp ble valgt. Det aktuelle alternativet blir så automatisk linket til den nye noden.

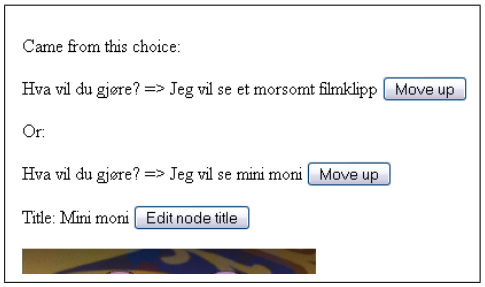
Når det har blitt lagt til et filmklipp til et alternativ, eller et alternativ har blitt linket til en node, vil visningen av den aktuelle nodens valg, se omtrent ut som vist i figur 4.6.

Figur 4.6: Visning etter at et filmklipp er lagt til et alternativ

Trykker brukeren på knappen «Go deeper», vil noden, som alternativets link peker til, bli satt som ny «aktuell node», og redigeringsvisningen vil vises med fokus på nettopp denne noden. Da vil det altså være mulig å legge til branches og lignende, til denne noden.

Når man er i en node som ikke er den første noden, vil man se alle valg man kan ha sett, med hvilke valgalternativ man, isåfall har tatt, for å komme til den aktuelle noden. Det er også mulig å gå tilbake til nodene disse valgene er i, ved å trykke på knappen «Move up». Eksempel på dette er vist i figur 4.7 på neste side. Funksjonen som viser denne visningen er relativt krevende og kjører en ganske tungvindt algoritme, men yttelsen av systemet virker likevel å være helt akseptabel. Denne figuren viser også at det er mulig å sette en tittel på den aktuelle noden. Denne tittelen vil bare være synlig for brukeren i valg-graf-redigeringsprosessen, og dens eneste hensikt er at man skal kunne sette et

alternativ til å linke til denne noden.



Figur 4.7: Brukerens mulighet til å gå høyere opp i grafens hierarki, altså til en av de nodene som har pekt til den aktuelle noden, og mulighet til å endre tittel på den aktuelle noden.

Ytterligere funksjonalitet

I tillegg til funksjonaliteten allerede beskrevet, finnes det også en slags hovedmeny for valg-graf-redigering.

Her har brukeren mulighet til å legge til en ny graf. Om brukeren trykker på dette alternativet, vil det i realiteten ikke lages en ny graf, men en ny graf vil bli registrert i databasen med den informasjonen brukeren skriver inn. Grafen vil ikke skapes før brukeren velger å gå inn å redigerer strukturen.

Brukeren kan også velge en graf fra en en liste som viser alle hans grafer, og har mulighet til å enten redigere beskrivende informasjon om grafen, eller redigere filmens struktur (valg-grafen). Når brukeren redigerer informasjonen om grafen (samme visning som når han legger til en ny graf), har han mulighet til å legge til tittel på filmen, beskrivelse, og rulletekst. Det er også mulig å publisere filmen i denne visningen.

4.3.3 Forbedringer

Det er ikke tvil om at mange forbedringer bør gjøres til denne delen av systemet. Her kommer en oversikt over mangler som vi ikke rakk å utbedre.

Robusthet

1. Noe vi vil prøve å forhindre i en interaktiv filmvisning er at ett og samme filmklipp kan vises flere ganger i en filmvisning. Dette vil mest sannsynlig gi en merkelig deja vú-opplevelse for mottakeren, som ikke er ønskelig.

   Dette kan forhindres ved at ingen av nodene som er over den aktuelle noden i grafens hierarki vil være mulig å linke til, om man vil linke et alternativ til en node. Dette vil si at det ikke skal gå an å linke til noen av de nodene som kan ha blitt spilt før den aktuelle noden.

Det må også forhindres at når det skal legges til et filmklipp til et alternativ, listes bare de filmklippene som ikke er med i noen av nodene som er over den aktuelle noden i grafens hierarki, opp.

Disse sjekkene kan være relativt krevende, og det kan derfor være et alternativ å kjøre slike sjekker når grafen skal publiseres, og at det da kommer en feilmelding om hva som er galt. Da kan det også være ønskelig at brukeren kan velge å publisere filmen likevel.

Denne robustheten ble spesifisert i kravspesifikasjonen, men andre oppgaver har blitt prioritert i prosjektperioden, siden denne funksjonen ikke ble sett på som kritisk nok til å måtte gjøres noe med med det første.

2. Alle noders valg kan i skrivende stund settes til å ha et standardvalg. Denne muligheten bør kun være mulig for det valget som skal vises sist i noden, og om valget er satt til å ha en tidsbegrensning. Grunnen til dette er at et standardvalg bør ikke tas om det er flere valg igjen i noden som skal vises senere, og er det ikke satt noen tidsbegrensning for valget, vil ikke filmavspilleren vite hvor lenge den skal vente før standardvalget skal tas.

Funksjonalitet

1. Det er ønskelig at det skal gå an å manøvrere seg gjennom grafen på en raskere måte. Slik redigeringsverktøyet fungerer i skrivende stund ser vi for oss to måter dette kan bli gjort på, begge kan være implementert samtidig.

   En løsning kan være å ha en «dropdown-meny» som lister opp alle noder som brukeren har gitt en tittel. Ved å trykke på en av disse, vil den valgte noden bli satt som aktuell node.

   En annen løsning kan være å vise et minikart over grafens struktur, altså et bilde av hvordan nodene er koblet sammen. Trykker man på en node, vil den valgte noden bli satt som aktuell node. En slik løsning er en god del mer avansert enn den første, men den gir mulighet til å hoppe til alle noder i grafen, uavhengig om de er gitt en tittel eller ikke. Et slikt kart kan tegnes opp for eksempel ved hjelp av et program som heter *Graphviz* [8], og bør være enkelt relativt enkelt å gjennomføre. Å bestemme hvilken link som skal vises ut ifra hvilke koordinater i bildet som ble trykket på er mer komplisert, men fullt gjennomførbart.

Brukervennlighet

1. En stor ulempe med tanke på effektiviteten i valggraf-redigeringen, er at nettsiden lastes på nytt veldig ofte. At et nytt vindu lastes når elementer skal redigeres burde vært forhindret. At data om grafen og dens struktur endres uten at siden lastes på nytt, kan løses på flere måter for eksempel ved hjelp av AJAX. Da kunne redigeringsvinduet kommet fram i rikig kontekst, med en gang redigerknappen har blitt trykket på. Og når brukeren trykker «Save», endres verdien i XML-filen og i visningen, uten at siden faktisk lastes på nytt.

2. En annen løsning som kan være god med tanke på å optimalisere brukervennligheten, og spesielt med tanke på kontinuitet i systemet, er å utvikle redigeringsverktøyet i Flash. Grunnen til at dette ikke er gjort, er den samme som nevnt i 4.1.1 på side 21, nemlig at gruppen har bygget seg en oppfatningen av at ActionScript har en del begrensninger og er vanskelig å jobbe med.

3. Et annet viktig grep som bør gjøres, er at designet i redigeringsapplikasjonen må bli mer intuitivt. Slik løsningen er i skrivende stund, er den mest sannsynlig ganske vanskelig å forstå og lære seg for nye brukere som ikke har spesiell kunnskap til systemet.

4.4 Filmavspiller

For at det skal være noe poeng i prosessene for opplasting, transkoding og valg-graf-redigering, må det finnes en applikasjon som kan spille av de interaktive filmene. Hele resten av systemet bygger opp under nettopp denne filmavspilleren. Denne spilleren er også sannsynligvis den applikasjonen som vil bli brukt mest, siden det antas at systemet potensielt har flere mottakere enn brukere.

I korte trekk skal denne spilleren hente strukturen til en interaktiv film (valg-grafen), og begynne å spille av filmklippet til den første noden. Hvert valg i nodene skal vises for brukeren, og trykker brukeren på et alternativ i et valg, skal filmspilleren begynne å spille av den noden alternativet linker til. Slik vil spilleren holde på helt til den kommer til en node uten valg. Da er filmvisningen ferdig og om brukeren har definert en rulletekst, vil denne vises her.

Det har hele tiden vært enighet om at denne filmavspilleren burde lages i Flash, siden veldig mange plattformer har støtte for Flash og fordi all den nødvendige funksjonaliteten er mulig å oppnå ved hjelp av Flash.

4.4.1 Høstens løsning

I høstsemesterets prosjekt utviklet vi en slik filmavspiller. Denne spilleren baserte seg riktignok på den gamle XML-strukturen, hadde begrenset funksjonalitet og en del feil og mangler, og brukte en del dårlige løsningen med tanke på blant annet innlesning av XML-filer og visning av valg.

Vi syns derimot at måten spilleren var *tenkt* at skulle jobbe på, var god. Meningen var at filmstrukturen først skulle leses inn i programmet. Så skulle programmet gå gjennom alle nodene og hente data om hver nodes filmklipp, fra databasen. Dette skulle skje ved at et lite PHP-program, med filmklipp-id som parameter, ble kjørt og lest inn i programmet. Denne PHP-filen, *DB2xml.php*, skulle formateres som en XML-fil for at innlesningen skulle være enkel. Denne tankegangen oppstod fordi vi mente det ville være både vanskelig og dumt å legge selve databasekommunikasjonen i Flash-programmet, og at det ville være enklere å lese inn XML-filer i Flash-programmet, enn å bruke Flash-programmet til å kommunisere med databasen. Den nevnte *DB2xml*-filen kan også returnere data om andre tabeller, ved å kjøre andre SQL-spørringer. Hvilke spørring som skal kjøres og hvordan resultatet skal formateres, bestemmes av hvilken parametere som blir sendt med til filen.

4.4.2 Vårens nye krav

Ett av målene for vårsemesterets hovedprosjekt var derfor å lage en helt ny filmavspiller, som brukte den nye XML-strukturen, og viste filmen og valgene på et bedre vis enn den første versjonen av filmavspilleren.

Det var også ønskelig å implementere en god del ny funksjonalitet. De fleste av kravene beskrevet i 2.2 på side 11 var ikke implementert i høstens versjon, og det måtte derfor utvikles løsninger for disse.

I tillegg til kravene definert i kravspesifikasjonen, oppstod det underveis i prosjektperioden nye ønsker som ble tatt til følge. Et eksempel på dette, er et

forslag om at filmavspilleren burde kunne pauses når et valg vises om brukeren ønsket dette.

4.4.3 Innlesing av den interaktive filmen

I høstsemesteret gjorde vi oss en del erfaringer med tanke på programutvikling i ActionScript og Flash. Vi opplevde for eksempel at det var ryddig, og ikke minst nødvendig med tanke på å kunne bruke klasser, å legge koden til klasser i egne .ai-filer.

I tillegg har det hele tiden vært enighet om at programmet bør være objektorientert siden det skal inneholde grafstrukturer.

Lesing av XML-filer

Det første programmet skulle gjøre, var å lese inn strukturen til en gitt graf.

Et av de største problemene vi støtte på under høstsemesterets prosjekt, var at ActionScript og Flash ikke leste inn data fra XML-filer slik vi hadde forventet det. Det viste seg nemlig at Flash i utgangspunktet vil kjøre ferdig alle koden i en keyframe før dataene fra XML-filen(e) ble lest inn. Dette bød på mange problemer siden det jo nettopp var disse dataene koden baserte seg på.

Løsningen vår virket til slutt på den måten at strukturens XML-fil ble lest inn i Flash-filmens første frame, deretter ble data om filmklippene lest inn i frame nummer to, og den interaktive filmen ble vist i tredje frame. Dette var ikke en yndet løsning, spesielt med tanke på at innlesningen av filmklippdata faktisk ikke alltid fungerte som den skulle.

Vårens løsning på dette problemet kom vi fram til etter litt intensiv grubling. Vi mente at løsningen ikke var spesielt pen da idéen først oppstod, men om det skulle virke ville ikke det ha noe å si, og i etterkant er vi fonøyd med denne løsningen. Uansett går løsningen ut på at datainnlesningsprosessen initialiseres ved at en funksjon kjøres, og at denne funksjonen tilkaller en annen funksjon når den har gjort jobben sin. Etter den har gjort sin jobb, tilkalles neste funksjon, fra den funksjonen. Her er et beskrivende eksempel:

```
function foersteFunksjon() {
  //gjør noe innhold
  ...
  //kjør neste funksjon
  andreFunksjon();
}

function andreFunksjon() {
  //gjør noe innhold
  ...
  //kjør neste funksjon
  nesteFunksjon();
}

//initialiser prosessen ved å kjøre første funksjon
foersteFunksjon();
```

Vanligvis ville vi ha gjort funksjonskallene etter hverandre på denne måten:

```
//kjør første funksjon
foersteFunksjon();

//kjør andre funksjon
andreFunksjon();

//Kjør neste funksjon
nesteFunksjon();
```

Ved å løse det på den måten først beskrevet, forsikret vi oss om at dataene neste funksjon baserte seg på, var tilgjengelige når neste funksjon blir kalt.

Innlesing av informasjon om strukturen

Det første programmet gjør er å opprette en instans av klassen `OptionGraph`. Dette objektet skal inneholde all informasjon om en interaktiv film. I constructoren til denne klassen, initialiseres prosessen med å lese inn all data om strukturen. Det første som skjer i denne prosessen, er at programmet brukeren en parameter den har fått tilsendt som er den interaktive filmens database-id, for å hente ut informasjon om en aktuelle strukturen fra *optiongraphs*-tabellen i databasen. Her brukes den tidligere nevnte PHP-filen *DB2xml*, med en parameter ved navn `getGraph`, satt til å være lik grafens database-id. PHP-filen blir «parset» av Flash-programmet, og data om den interaktive filmen blir lagt til i `OptionGraphs`-objektet.

En av dataene som blir returnert i spørringen som blir gjort er filbanen til den aktuelle filmens XML-struktur. Når programmet vet hvor denne filen befinner seg, leser den inn filen og legger til all data i `OptionGraphs`-objektet. Strukturen på denne filen vil være som vist i figur 3.2 på side 18.

Innlesing av strukturen

For hver node i XML-filen, vil programmet opprette en ny instans av klassen `OptionNode` og legge denne til i en array ved navn `nodes`, i `OptionGraph`-objektet. I disse objektene vil nodens `xml:id` og det tilhørende filmklippets database-id legges til. I tillegg vil hvert valg i denne XML-noden bli lagt til i en array med `NodeBranch`-objekter, i `OptionNode`.

I hvert `NodeBranch`-objekt legges det til informasjon om det aktuelle valget. Denne informasjonen er: Når det eventuelt skal begynne å vises (`start_time`), hvor lenge det skal vises (`branch_duration`), valgets tekst (`branch_text`) og om filmen skal pauses når valget vises (`pause_movie`). Etter at alle valgene i en node er lest inn, sorteres nodens array med valg etter når de er satt til å vises. Den som som skal vises først, blir satt til slutt, slik at programmet senere kan «poppe» fra «toppen» av array'en.

I tillegg legges selvfølgelig informasjon om valgets alternativer til. Disse blir lagret i en todimensjonal array på denne måten:

```
links[] = [<linktekst>, <type link>, <linkverdi>]
```

Her lagres linkens tekst, linkens type (`node` eller `http`), og hvilken node eller nettside den skal linke til, i den rekkefølgen.

Når nodene er i ferd med å leses inn, sjekkes det også om nodenes `title`-attribute er satt til å være «`start`». Dette vil i så fall bety at redigeringsapplikasjonen har merket denne noden til å være den første noden som skal vises. Er dette tilfellet blir `OptionGraph`-objektets `start_key`-variabel satt til å være lik det nummeret i arrayen den aktuelle noden settes i.

Retting av linker

Etter at alle disse dataene om filmen er lagt til i `OptionGraph`-objektet, kjører programmet en funksjon som endrer alle node-linkene i alternativene, til å referere til `nodes`-array-key'en til en node. Disse linkene er i utgangspunktet satt til å referere til en nodes XML-id, som er en tekststreng som kommer fra `xml:id`-verdiene fra XML-filens noder. Dette fører til økt ytelse med tanke på at det vil være mye lettere for programmet å spille av en node om det vet array-key'en til den noden den skal vise, enn om det må gå gjennom alle noder i grafen og sammeligne en tekststreng i hver av disse.

4.4.4 Avspilling av filmen

Når første node skal spilles av, blir programmets aktuelle node-variabel (`curr_node`) satt til å være lik `OptionGraph`-objektets `start_key`-variabelen, som er satt under innlesing av strukturen. Deretter kjøres `OptionGraphs` `playNode`-funksjon, med `curr_node` som parameter. Et interval settes til å ha ansvar for å oppdatere visningen av totalt avspilt tid i sekunder: Mer om intervaler senere.

playNode-funksjonen

`OptionGraph`-objektets `playNode`-funksjon, er funksjonen som initialiserer og spiller av nye filmklipp. Det gjør den ved å opprette et `NetConnection`-objekt, med et tilhørende `NetStream`-objekt som laster ned filmklippet, og kobler dette til video-objektet på scenen i Flash-spilleren. Deretter settes filmen igang ved å kjøre `NetStream`-objeketets `play`-funksjon med filbanen som parameter. Filbanen ligger lagret i den aktuelle nodens `OptionNode`-objekt.

I tillegg til å igangsette selve visningen av et filmklipp, har denne funksjonen en del andre oppgaver. Under kommer en liste med noen av dem:

- Programmets aktuelle node-variabel blir satt til å være array-key'en til den noden som nå spilles.
- Noden blir lagt til i en nodehistorikk-array som holder rede på hvilke noder som har blitt spilt, og i hvilken rekkefølge.
- Hvor mye som eventuelt har blitt spilt av forrige nodes filmklipp legges også til i en array. Den holder rede på hvor mye som ble spilt av hver nodes filmklipp, for at det skal være mulig å regne ut hvor mange sekunder film som har blitt spilt siden den første nodens filmklipp ble påbegynt.
- Funksjonen tar også en kopi av arrayen som tar vare på valgene (`branches`), i den aktuelle noden. Dette er fordi det skal være mulig å «poppe» fra arrayen uten at man mister informasjon om de for godt.

- I tillegg igangsettes et intervall som hele tiden sjekker om neste valg skal vises. Mer om dette senere.
- Det igangsettes også et annet intervall som har i oppgave å lage en grafisk visning av hvor mye av filmklippet som er lest inn i filmspillerens buffer. Dette intervalet kjøres helt til hele filen er lest inn i bufferet, og viser en grå, smal stripe som «krabber» horisontalt bortover skjermen fra venstre bildekant. Dette er en visuell beskrivelse av hvor mange prosent av filmen som ligger i bufferet. Når stripen går til midten av x-aksen er 50% av filmklippet lest inn i bufferet, går stripen helt fra venstre bildekant til høyre bildekant er 100% av filmklippet lest inn i bufferet.
- Aspektet til video-objektet i Flash-spillerens scene, justeres om det skulle vise seg at filmklippet har et annet aspekt en det som er satt som standard for filmspilleren. På denne måten vil for eksempel ikke filmklipp i widescreen-format se utstrukket og merkelig ut.
- I tillegg blir automatisk «play»-knappen skjult og pauseknappen vist, siden filmen uansett spilles av i dette øyeblikk.

Visning av valg

Som tidligere nevnt, igangsettes et intervall i `playNode`-funksjonen, som sjekker om neste valg skal vises. Dette blir gjort ved at en variabel blir tilegnet et interval hvor det defineres at funksjonen `waitForBranch` kjøres fire ganger i sekundet. Dette ser omtrent slik ut:

```
waitForBranchInterval = setInterval(waitForBranch, ( 250 ) );
```

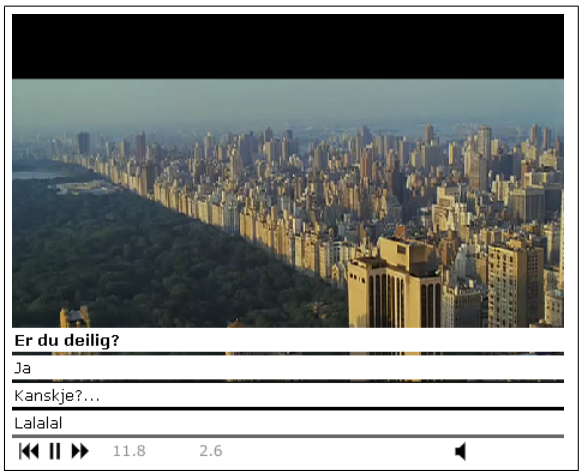
Denne funksjonen sjekker om `NetStream`-objektets `time`-egenskap er større eller lik det tidspunktet det neste valget i noden skulle vises. Når betingelsene for dette inntreffer, kaller denne funksjonen på `displayBranch`-funksjonen som popper den aktuelle branchens nodens `branches`-array, og bruker dataene som ligger i dette objektet til å vise et valg.

Om brukeren trykker på et av alternativene i et valg, vil spilleren enten kjøre `playNode`-funksjonen med node-referansen i den aktuelle linken som parameter, eller, om alternativet linker til en nettside, pause filmen og åpne nettsiden i et nytt nettleservindu.

Mer om visning av valg

Valget kan bli vist på flere forskjellige vis ut fra hvilke data som er lagret i det aktuelle valgets `OptionBranch`-objekt, i trå med kravene som er satt til valgene i underkapittel 4.3.2 på side 34.

Når valget har blitt vist igangsettes et nytt intervall som har som formål å fjerne valget når det er tid for dette. Dette skjer ved at funksjonen `waitForBranchHide` blir satt til å kjøres fire ganger i sekundet. Denne funksjonene velger enten å fjerne valget når: a) `NetStream`-objektets `time`-egenskap er blitt mer enn det den var da visningen av valget starter pluss valgets varighetsverdi. b) Når telleren som teller ned det antall sekunder som er definert som varigheten, har nådd 0. Varigheten ligger lagret i `OptionBranch`-objektets `branch_duration`-variabel.



Figur 4.8: Eksempel på visning av et valg i filmavspilleren

Situasjon **a** intreffer om valget er satt til å vises samtidig som filmavspillingen skjer og valget skal fjernes før slutten av filmklippets slutt. Situasjon **b** intreffer i de andre tilfellene: Altså når filmen har blitt satt til å pauses mens valget vises, eller at slutten av filmklippet er nådd før valget tidsbegrensing utløper.

Om et valg ikke er definert til å begynne før slutten av filmklippet, vil dette valget vises når denne funksjonen forstår at filmklippet er ferdigspilt. Vi hadde i utgangspunktet problemer med å få programmet til å forstå at et filmklipp var ferdigspilt, dette er fordi `NetStream`-objektets `time`-egenskap ikke nødvendigvis er det samme som FLV-filens `duration`-verdi. Vi prøvde flere løsninger for å finne ut om slutten av filmklippet var nådd. Den beste løsningen vi kom fram til var å sjekke om avspilt tid var *nesten* like mye som filmklippfilens varighet. Måten den gjør dette på, er å se om differansen mellom filmens lengde substrahert på `time`-egenskapen (runnet nedover) er mindre enn én prosent av filmens lengde (runnet oppover). Dette er absolutt ingen ideel løsning, men den ser ut til å virke godt hittil. Spørringen ser slik ut:

```
if( Math.floor(optionGraph.nodes[curr_node].media_length - ns.time) <
    Math.ceil(optionGraph.nodes[curr_node].media_length * 0.01) ) {
     ...
```

Om den aktuelle noden ikke har noen valg, vil filmen være ferdig når den aktuelle nodens filmklipp er ferdigspilt, og da vil funksjonen vise den aktuelle grafens rulletekst.

Rulleteksten vises ved at `OptionGraph`-objektets `end_credits`-variabels verdi blir lagt til en tekstboks i Flash-movieclip'et `end_credits_mc`, deretter settes denne movieclipet til å kjøre sine neste frame. Det som da skjer er at teksten blir flyttet gradvis oppover, fra å være under bildet til å være over. Dette skaper en rulleteksteffekt som vi kjenner igjen fra kinofilmer.

4.4.5 Annen funksjonalitet

Forrige klipp

Vi har lagt til en knapp som gjør det mulig begynne med å se starten av filmklippet til den aktuelle noden. Trykker man på den to ganger etter hverandre, spilles filmklippet til den forrige viste noden fra starten.

Denne funksjonaliteten var en av de funksjonene som ikke var planlagt før prosjektperioden, men ble likevel prioritert da den ble sett på som å være et sannsynlig ønske av potensielle mottakere.

Løsningen ble etter en del strev slik at for hver nye node som spilles av, blir referansen til denne noden lagt til i en array. Når mottakeren trykker to ganger på forrige-knappen, blir den siste referansen fjernet og den nest siste referansen blir brukt som parameter i `playNode`-funksjonen. Trykkes det én gang på forrige-knappen, blir den siste referanse brukt som parameter i `playNode`-funksjonen.

Arrayen som så vidt er blitt nevnt tidligere (som tar vare på den totale spilte tid), får også fjernet den siste instansen i array'en når forrige node blir spilt.

Spole framover

Det det ble også lagt til en knapp som gjorde det mulig å «spole» framover. Det viste seg riktignok vanskelig å få filmklippet til å spilles av fortere enn normalt når denne knappen var trykket ned. Siden denne funksjonaliteten uansett ikke ble sett på som viktig, ble det brukt liten tid på dette, og det ble fort inngått et kompromiss som gikk ut på at når mottakeren trykker på denne knappen, hopper filmen tre-fire sekunder fram i tid.

Tidsvisning

Vi har gjort slik at total spilt tid til enhver tid vises nederst i spilleren. Vi har også gjort slik at hvor mye tid som er igjen av visningen av det aktuelle valget vises til høyre for dette. Dette har vi gjort for at mottakeren skal ha bedre oversikt over hvor langt han har kommet i filmen, og hvor lang tid han har på seg til å ta et valg.

4.4.6 Valg rundt GUI

- Enkelte vil kanskje savne en «scrubber» i filmspilleren som gjør det mulig å raskt spole til hvor man vil i filmklippet. Gunnen til at denne funksjonaliteten ikke er implementert, er tanken om at mottakeren ikke bør trenge å være oppmerksom på at filmvisningen faktisk består av flere filmklipp. For ham skal det virke som han faktisk er med på å endre handlingen i en ordentlig film, og ikke bare surfer innom forskjellige filmklipp på for eksempel nettstedet youtube.com.
- Det er riktignok mulig å gå tilbake til forrige node. Dette er fordi vi har tenkt at det bør finnes en mulighet til å gå bakover på et vis om mottakeren angrer på et valg han har tatt.
- Det er derimot ikke mulig å *spole* bakover. Grunnen til dette er at det da blir veldig vanskelig å vite hvilke valg som eventuelt skal vises i den

perioden som er spolt forbi, siden disse er poppet fra en array, som ikke er tilgjengelig i sin helhet med mindre noden kjøres på nytt ved hjelp av `playNode`-funksjonen.

4.4.7 Forbedringer

Design

- Hele spilleren bør ha et penere design for å være mer appellerende for potensielle mottakere. Vi ser for oss at dette kan gjøres ved at valg og valgtekst blir satt til å være delvis gjennomsiktige. Og at gradienter legges på de hvite feltene i spilleren som skaper en dybdeeffekt.
- Når et valg blir tatt av enten brukeren, eller blir automatisk valgt fordi valgets tidsbegrensning har utløpt, bør det vises bedre at det aktuelle valget har blitt valgt. Vi ser for oss at dette kan gjøres ved at en blinkeeffekt blir satt på det aktuelle valget i omtrent et halvt sekund etter at valget har blitt tatt.
- Når et valg skal skjules, enten fordi det har blitt valgt et alternativ, eller fordi tidsbegreningsen har utløpt, bør valgtekst og alternativer forsvinne gradvis ved å bli mer og mer gjennomsiktig. Det samme gjelder når valgene skal vises, de bør fade inn. Dette vil gjøre at bevegelsene i spilleren ikke vil være så brå som de er i skrivende stund.
- Det bør vises flere forklarende ikon. For eksempel burde det blitt vist et timeglass ved telleren som teller ned hvor mye lenger det aktuelle valget skal vises. Og en klokke kunne blitt vist ved siden av total tid.

Disse designgrepene ble alle nedprioritert i prosjektperioden siden de ikke ble sett på som kritiske til systemet, men det sees ikke bort fra at de vil bli gjort noe med kort tid etter prosjektet er levert.

Funksjonalitet

- En funksjon som trolig vil bli savnet av enkelte er en volumekontroll som gjør det mulig å regulere volumet på lyden. Denne funksjonen ble ikke sett på som kritisk nok til å måtte bli gjort noe med under dette prosjektet.
- En annen funksjonalitet vi ser for oss kan være ønskelig er at brukeren bør kunne bestemme om mottakeren skal ha mulighet til å gå tilbake til starten av nodens filmklipp, og forrige nodes filmklipp, om mottakeren skal ha mulighet til å pause filmen når valg vises, og om han skal ha mulighet til å spole framover.

  På denne måten kan brukeren ha mulighet til å påvirke mottakerens opplevelse ytterligere.

  Denne funksjonaliteten blir ikke sett på som for vanskelig å implementere, men ble ikke prioritert i dette prosjektet siden det ikke var sikkert hvor ønskelig en slik funksjonalitet var, og fordi det var mange andre, mer kritiske funksjonaliteter som skulle på plass.

- Veileder kom på slutten av prosjektperioden med et forslag til ny funksjonalitet. Nemlig at det hadde vært interessant om det hadde vært mulig å koble flere filmklipp til en node. Disse ville da spilles av samtidig, og brukeren vil kunne hele tiden velge hvilket klipp han skal se. Dette kan være hensiktsmessig med tanke på for eksempel flerkameraproduksjoner hvor mottakeren skal ha mulighet til å velge kameravinkel.

  Implementeringen sees ikke på som avansert, men det tas høyde for at det vil oppstå veldig mange uante problemer, og at ytelsen av programmet vil bli kraftig rammet. Det anses derfor at det vil ta en del tid å teste og utvikle en slik funksjonalitet til akseptabel standard.

- Det bør også være mulig for brukeren å velge at et valgs alternativ ikke nødvendigvis skal linke til noe, og når dette valget eventuelt blir tatt bør filmen bare fortsette å spilles av. Dette sees på som veldig enkelt og raskt å implementere, grunnen til at det ikke er gjort er fordi idéen oppstod helt på slutten av prosjektperioden.

- Det burde også være mulighet for å registrere de forskjellige rutene som har blitt tatt gjennom de forskjellige grafene i databasen. På denne måten vil det kunne være mulig å se hvor mange som har sett den samme unike visningen. En slik funksjonalitet sees ikke på som alt for vanskelig å implementere, men idéen oppstod alt for seint i prosjektperioden til at vi kunne fått tid til den.

4.5 Nettsiden

For at prosessene i systemet skulle kunne brukes måtte de knyttes sammen i et internett-grensesnitt. Nettsidenes struktur og CMS-delen av systemet, har likevel ikke vært hovedfokus i dette prosjektet, siden de andre prosessene ble sett på som mer kritiske.

Listemenyer

Meningen er at en mottaker skal kunne få tilgang til å se alle publiserte interaktive filmer. Alle slike filmer må derfor listes opp for mottakeren og det må være mulig å trykke på og se disse.

Brukeren har mulighet til å logge seg inn i «systemet bak». Her vil han ha mulighet til å redigere informasjon om sine grafer og ikke minst strukturene på grafene. Han vil også ha mulighet til å legge til nye grafer.

XHTML

All HTML-kode og PHP-kode som generer HTML har blitt utviklet med tanke på at HTML-koden skal validere som XHTML. Dette gjør at det er større sannsynlighet for at visningen av sidene ser lik ut på alle klientmaskiner.

Visning av Flash-programmet

For visning av den interaktive filmen har vi valgt å bruke JavaScript-klassen `SWFObject` [9] som importerer Flash-filer i HTML-koden, og som gjør koden ryddigere og at den validerer som XHTML.

CSS

Det har i skrivende stund ikke blitt gjort mye redigering av sidene. Noe er riktignok gjort og det er i hovedsak å plassere objekter i HTML-koden omtrent der de er tenkt at de skal være.

Figur 4.9 på neste side viser hvordan interacTVs nettsider ser ut i skrivende stund.

4.5.1 Forbedringer

Funksjonalitet

- Brukere bør ha mulighet til å invitere nye brukere.
- Brukere må ha mulighet til å redigere personlig informasjon på sin profil.
- Brukerens opplastede filmklipp bør gjøres lettere tilgjengelig for brukeren til visning og redigering. Med redigering menes at beskrivende informasjon om klippet kan redigeres eller at klippet kan slettes.
- Når en interaktiv film blir vist for en bruker bør det registreres i databasen at denne filmen har blitt sett en gang mer.

Figur 4.9: Nettsiden i skrivende stund

- Når en film er ferdigspilt bør det være mulig for en mottaker å sette karakter på filmen.

Noen av disse forbedringene ble spesifisert i kravspesifikasjonen, men ble likevel nedprioritert siden CMS-delen av systemet aldri ble sett på som hovedfokus i prosjektet, fordi andre prosesser ble sett på som mer kritiske enn disse, og selvfølgelig på grunn av tidsnød.

Design

Det er ønskelig at alle nettsidene i systemet har det samme designet, og at dette designet skal være intuitivt og estetisk. Det må nevnes at dette ikke har blitt fokusert på i dette prosjektet.

Kapittel 5

Drøftinger

I dette kapittelet har vi skrevet subjektive meninger om hva vi mener om prosjektet og resultatet. Her blir temaer drøftet uten den objektiviteten som er brukt i tidligere kapitler.

5.1 Resultater

Vi mener vi har oppnådd målene i underkapittel 1.1.1 på side 6. Begrepet *enkel* i det første effektmålet er riktignok et relativt begrep, og vi skal være enig i at mangelen på brukervennlighet i valg-graf-redigering ikke nødvendigvis gjør denne delen av systemet spesielt enkel. Vi føler uansett at prosjektet har forenklet skapelsen av interaktive filmer for visning på internett.

Vi har oppfylt de fleste kravene i kravspesifikasjon, pluss noen flere. Vi har lagd et system med mer funksjonalitet enn det som i utgangspunktet var tenkt i starten av prosjektperioden. Riktignok har vi valgt å ikke oppfylle absolutt alle kravene i kravspesifikasjonen. Dette kunne nok ha latt seg gjennomføre, men vi har valgt å fortløpende revurdere prioriteten av forskjellige krav, noe som til tider har ført til implementering av nye krav, framfor de originale.

Vi mener også at hovedfunksjonaliteten i systemet fungerer som det skal. Vi ser også at en del av løsningene kunne og burde vært gjort på andre måter for å øke yttelsen av systemet. Dette blir riktignok ikke sett på som veldig kritisk med det første, siden systemets yttelse har vært tilfredsstillende.

Med tanke på hvor lite vi har fokusert på brukervennlighet i redigeringsapplikasjonen og i nettsidene, føler vi at disse endte opp med relativt god brukervennlighet. Vi ser selvfølgelig at den kan forbedres, men med de lave forventninger vi hadde satt til brukervennlighet, må vi si oss fornøyd med resultatet for denne gang.

Det samme gjelder for designet på utseende av systemet.

Skape en film

I kapittel 2.1 på side 11 nevner vi at vi har lyst til å lage en interaktiv film som kan vise potensiale av systemet. Vi har riktignok vært nødt til å utsette denne oppgaven til etter prosjektperioden. Dette har vært en relativt tung avgjørelse,

men det lot seg bare ikke gjøre i prosjektperioden. En del planlegging har likevell blitt lagt ned i denne oppgaven, og det vil sannsynligvis lages en slik film kort tid etter at denne rapporten er levert.

5.2 Videre arbeid

Som det har kommet fram i en del av underkapitlene i kapittel 4 på side 20, er det er forbedringspotensiale i de fleste av systemets prosesser. Dette i kombinasjon med at systemets tema og bruksområde blir sett på som veldig interessant av gruppen og bekjente, gjør sannsynligheten stor for at dette systemet vil bli videreutviklet i fremtiden.

Det vi ser på som det største problemet med tanke på videre drifting, vil være å ha tilgang til serverkraft. Skulle systemet utvides og gjøres mer tilgjengelig, kunne det være en idé å få med noen kommersielle aktører for å blant annet kunne ha midler til å leie servere.

Som grunnlag for videre jobbing, med tanke på hovedprosjekter, føler vi at det bør kreves en god del ny funksjonalitet av systemet om dette systemet skulle videreutvikles i et lignende prosjekt. Hvis ikke ser vi for oss at arbeidet ikke ville ha blitt «nyskapende» nok, selv om forbedringspotensialet absolutt er tilstede.

En annen mulighet vi ser for oss er at det kunne bli gjort for eksempel et medie management-hovedprosjekt, hvor for eksempel markedsføring av systemet stod i fokus og hvor man kunne kartlagt den potensielle brukermassen for et slikt system.

5.3 Teknologier benyttet

I tillegg til de teknologiene tidligere nevnt i rapporten, tilegnet vi oss kunnskap om teknologier som ikke var spesifisert i noen krav.

Eksempel på slike teknologier er SVN. Vi hadde aldri brukt noen form for revisjonskontrollsystem tidligere, og det har vært svært lærerikt og til ekstremt god hjelp med tanke på backup og utvikling generelt.

En annen teknologi som var ny for oss før dette prosjektet, var LaTeX. Vi brukte det for første gang da vi lagde forprosjektrapporten. Denne rapporten er skrevet i LaTeX og det kommer definitivt ikke til å være den siste gangen vi bruker det.

Underveis i prosjektet tok vi et valg om at all kodekommentering skulle være på engelsk. Før dette prosjektet har vi aldri vært særlig konsekvent på dette, men på grunn av prosjektets størrelse, og potensialet for videreutvikling, ble det fokusert en del på at koden burde være godt dokumentert og lett leselig. Da dette valget ble tatt, tok vi også et valg om at det ikke var noen grunn til å skrive kommentarer på norsk.

Av samme grunn begynte vi også å bruke dokumentasjongenereringsverktøyet *doxygen*. Dette er et verktøy som genererer oversiktlig dokumentasjon av koden, men for at det skal fungere riktig må kommentarene i koden formateres på et spesielt vis. Dette førte til at det ble mer kontinuitet i koden, og at den ble lettere å lese. Vi har prøvd å kommentere så mye som mulig på engelsk i

doxygen-format, men tidsnød gjorde at noen funksjoner og klasser ble utelatt. Dokumentasjonen kan ses på rapportens CD-ROM.

5.4 Evaluering av gruppas arbeid

Vi føler vi har vært flinke til å jobbe, og at det blitt jobbet mye med prosjektet.

Det har gått fint å være én på prosjektgruppen med tanke på motivasjon til å jobbe. Vi har lagt mange timer ned i dette prosjektet og er godt fornøyde med innsatsen.

Det har vært mye lettere å motivere seg til å utvikle systemet enn å skrive rapport. Rapportskrivingen har generelt vært ganske tungt, særlig siden det har bare vært én student på gruppa, og progresjonen i skrivingen har derfor gått tregt. Noe som også har vært vanskelig på grunn av gruppens størrelse, har vært å få objektiv oversikt over problemer. Det har vært vanskelig å tenke bredt siden man ikke har fått innspill fra andre gruppemedlemmer med andre synsvinkler.

Å være én på gruppa har vært fordelaktig med tanke på friheten og den helhetlige kontrollen over prosjektet det medfører.

5.5 Prosjekt som arbeidsform

Å jobbe med hovedprosjekt har vært en av de mest lærerike, krevende og morsomme opplevelsene vi har opplevd på HiG, og med omfanget et slikt prosjekt har, vil det bli husket i lang tid.

Det har vært veldig vanskelig å gå fra å utvikle én del av systemet til å begynne å utvikle en annen del, både fordi det byr på en ny kontekst, og fordi det er en vond følelse å gå fra noe man ikke føler seg helt ferdig med, fordi man må velge å gjøre ferdig mer kritiske deler av systemet.

5.6 Subjektiv opplevelse av hovedprosjektet

Vi syns det har vært en veldig schizofren opplevelse å konstant refre til seg selv i 1. person flertall i rapporten. til seg selv i 1. person flertall i rapporten.

Kapittel 6

Konklusjon

Hovedprosjektet har for meg vært en ny måte å jobbe på med tanke på prosjektets omfang og verdi. Det har vært en fri måte å jobbe på som har gitt meg mye faglig kunnskap og mange verdifulle erfaringer. Hele opplevelsen har vært veldig givende siden det meste har vært på eget initiativ.

Jeg føler jeg har fått brukt den kunnskapen jeg har opparbeidet meg gjennom mine tre år på Høgskolen i Gjøvik til å skape et produkt som både er interessant og representativt for mine evner.

Tanken om interaktive filmer virket spennende for et år siden, men etter å ha jobbet med det i et halvt år, virker det faktisk enda mer interessant. Jeg er fornøyd med resultatet, men ser også fram til å jobbe videre med systemet.

Jeg føler dette hovedprosjektet har vært en ideell avslutning på mitt bachelorstudie ved HiG. Det har vært hektisk, men det har definitivt vært verdt det.

Bibliografi

[1] Emnebeskrivelse for imt3911. http://hig100.hig.no/fagplaner/fagbeskrivelse.php?fagkode=IMT3911&aar=2006&sprk=1.

[2] Tree (data structure). http://en.wikipedia.org/wiki/Tree_%28data_structure%29. From Wikipedia, the free encyclopedia.

[3] Handling file uploads: Common pitfalls. http://no2.php.net/manual/en/features.file-upload.common-pitfalls.php.

[4] Dvd-video disc workshop. http://stream.uen.org/medsol/dvd/pages/dvdvid_tools_camcorders.html, 2005.

[5] How to open/read/write a local file from an applet. http://www.captain.at/programming/java/.

[6] Useful code of the day: Multipart form file upload. http://forum.java.sun.com/thread.jspa?forumID=31&threadID=451245.

[7] The java tutorials: Security restrictions. http://java.sun.com/docs/books/tutorial/deployment/applet/security_practical.html.

[8] Graphviz - graph visualization software. http://www.graphviz.org/.

[9] Geoff Stearns. Swfobject: Javascript flash player detection and embed script. http://blog.deconcept.com/swfobject/.

Tillegg A

Definisjoner

ActionScript	Et skripte-/programmeringspråk, primært forbundet med Adobe Flash-programmer.
AJAX	Ajax, eller Asynkron JavaScript og XML, er en webutviklingsteknikk for å lage interaktive nettsider. Dette gjøres ved at sidene utveklser litt og litt data med serveren i bakgrunnen, i steden for å laste hele siden på nytt hver gang brukeren gjør en forandring.
Applet	En programkomponent som kjører i et annet program, for eksempel i en nettleser.
BASH	Bourne-again shell, skriptspråk for utvikling av UNIX-shell.
CMS	Countent Management System
DOM	Document Object Model (DOM) er en beskrivelse av hvordan en HTML- eller XML-trestruktur er representert.
Doxygen	Dokumentasjonsgenerator.
ffmpeg	lyd- og video-konverteringsverktøy
Flash	Adobe Flash (tidligere Macromedia Flash) et verktøyet for å publisere animasjoner, multimedia og applikasjoner henholdsvis på Internett.
FLV	Flash Video
FTP	FTP (File Transfer Protocol) er en standard, operativsystemuavhengig protokoll for overføring av filer i et TCP/IP-basert nettverk.
HTTP	HTTP (Hypertext Transfer Protocol) er protokollen som primært benyttes på nettet for uveksling av informasjon.
Java	
JSP	JavaServer Pages Technology
MD5	MD5 (Message-Digest algorithm 5) er en sjekksumalgoritme lager en 128-bits sjekksum, som vanligvis oppgis som et 32-tegns heksadesimalt tall.
MIME	Multipurpose Internet Mail Extensions, brukes til å beskrive filtyper
mplayer	MPlayer er et mediespilleprogram som støtter de aller fleste formater, både innen lyd og video.
MySQL	MySQL er en SQL-basert databasetjener som er lisensiert under GPL.
PEAR	PHP Extension and Application Repository
PHP	PHP: Hypertext Preprocessor, et dynamisk, tolket og løst typet programmeringsspråk hovedsakelig brukt for å utvikle dynamiske nettsider.
SVN	Subversion er et versjonskontrollsystem som brukes til å holde rede på utviklingshistorien til en samling med filer og kataloger.
swfObject	JavaScript som legger til et Flash-program i HTML-koden på en måte som gjør at html-koden kan validere som XHTML.
Use Cases	En type verktøy for å spesifisiere krav til systemet og deler av systemet.
XML	Extensible Markup Language, et standardisert format for strukturerering data.

Tillegg B

Forprosjektrapport

B.1 Mål og rammer

B.1.1 Bakgrunn

Høstsemester 2006 ble det i emnet Fordypning i medieteknikk (IMT391), ved Høgskolen i Gjøvik, gjennomført et prosjekt hvor målet var å lage et nettbasert system rundt visning, opplastning og håndtering av det gruppen kalte *interaktiv film*.

Med interaktiv film menes en filmvisning hvor *mottakeren* kan være med å bestemme handlingen i filmen ved å ta valg, som på forhånd er gitt av en *bruker*. Med *mottaker* mener vi en person som ser og kommuniserer med filmvisningen på nettsiden. Med *bruker* mener vi en som har tilgang til sidene som gjør det mulig å skape, redigere og dermed legge til rette for en slik filmvisning. Mottakeren vil bli stilt ovenfor disse valgene på gitte plasser i filmvisningen, og ut ifra hva mottakeren velger utspiller forskjellige handlinger seg. Tanken med dette er å skape en dynamisk filmvisning hvor forskjellige mottakere kan få helt forskjellige filmopplevelser ut ifra hvilke valg de tar underveis i filmvisningen. Potensielt sett vil dette for eksempel si at en mottaker kan se en lang filmvisning hvor en av karakterene i filmen opplever masse i mange forskjellige omgivelser, og at en annen mottaker tar andre valg som gjør at denne karakteren for eksempel dør tidlig i filmen, og det er historien til andre karakterer som er filmen for denne mottakeren. Teoretisk sett kunne disse to mottakerne ha sett akkurat den samme filmen, men ut ifra at de i et ledd i filmvisningen har tatt forskjellige valg blir også filmen de ser forskjellig.

For at slike filmvisninger skal være tilgjengelige bør det også finnes et system som er enkelt å bruke, og som gir brukerne mulighet til å laste opp filmklipp til nettstedet og å sette disse sammen til en filmvisning, med valg og valgmuligheter og spesifisering av sammenhengene mellom de forskjellige valgene og filmklippene.

B.1.2 Effekt- og resultatmål

Effektmål

- Tilby en enkelt måte å laste opp filmklipp og klargjøre filmvisninger for brukere.
- Gjøre det mulig å gjennomføre og vise interaktive filmer for mottakerne.

Resultatmål

- Lage et nettbasert system som gjør det mulig for registrerte brukere å laste opp filmklipp i forskjellige filformater og filstørrelser, for så sette sammen og lagre disse i en sammenheng, med valg og valgmuligheter, noe som vil legge til rette for visningen av den interaktive filmen.
- Lage en spiller som kan implementeres på en nettside og som kan vise filmklipp og valg, for mottakere, i den rekkefølgen som er angitt i visningsstrukturen som er lagret på tjener.

B.1.3 Målgruppe

Mottakerne er de som besøker nettstedet for å se, oppleve og bruke, og altså å ta valg som påvirker historien, i de interaktive filmene.

Brukere er de som benytter seg av "systemet bak" for å laste opp filmklipp og koble sammen disse med valg og valgmuligheter for å skape en filmvisning som kan sees og brukes av *mottakerne*.

Sensor vil bruke prosjektrapporten sammen med kode for å kunne evaluere prosjektets omfang og kvalitet.

B.1.4 Rammer

Hovedprosjektet har en arbeidsmengde tilsvarende 20 studiepoeng. 20 studiepoeng er beregnet til å tilsvare en arbeidsmengde på ca 25 timer i uka for en student. Siden prosjekt kun vil bli utført av en person vil det si at arbeidsmengden for prosjektet skal tilsvare ca 25 timer i uka.

B.2 Omfang

B.2.1 Oppgavebeskrivelse

Vi skal lage et nettbasert system som gjør det mulig for registrerte brukere å logge seg inn, laste opp filmklipp fra klienten til tjener, for så å skape rammeverket som skal til for å vise en interaktiv filmvisning. Vi skal også lage en løsning som gjør det mulig å spille av denne filmvisningen for mottakeren og som gjør det mulig for mottakeren å kommunisere med avspilleren slik at mottakeren kan ta valg som er med å bestemmer historien i filmen.

Vi skal lage et brukersystem som gjør det mulig for brukere å logge seg sikkert inn og ut, og endre persondata og bilde.

Vi skal lage en løsning for å laste opp store filmfiler til serveren. Dette er i utgangspunktet vanskelig siden tjenersideprogrammeringsspråk som PHP har dårlig støtte for filer over 2MB. En løsning vil være å implementere en Java Applet i nettsidene, som kjøres fra klienten og stykker opp den aktuelle filmfilen til mindre filer, og sender disse en og en til tjeneren. Da må det også lages en applikasjon på tjenersiden som setter sammen disse filene.

Vi skal lage en løsning for å spille av interaktive filmer for mottakeren. Mottakeren skal ha mulighet til å ta valg han/hun blir stilt ovenfor, og filmvisningen skal automatisk spille videre ut ifra hva mottakeren har valgt. I realiteten vil spilleren spille av mange forskjellige filmklipp, men for mottakeren bør det virke som om han/hun er med å bestemme handlingen i en sammenhengende film. Mottakeren skal ha mulighet til å pause filmvisningen og justere volumet på lyden.

Brukerne må ha mulighet til å opprette og redigere det vi kaller *valgtrær*. Hensikten med valgtrærne er å lagre sammenhengen mellom filmklipp, valg og valgmulighetene for en filmvisning. Valgtrær er altså oppskriften på en filmvisning. Når en bruker redigerer slike valgtrær skal han/hun ha mulighet til å legge til filmklipp han/hun selv har lastet opp. Etter hvert filmklipp som er lagt til i valgtreet skal det være mulighet for å legge til et valg. Til hvert valg må det legges til minst to (svar)alternativet. Til hvert alternativ må det legges til et tilhørende filmklipp.

Det er også ønskelig å gi brukerne enda flere muligheter. For eksempel vil vi at det skal være mulig å sette et alternativ til å vises underveis i avspillingen mens filmen spilles av. Bruker skal altså kunne spesifisere når i filmklippet et valg skal vises for mottaker, slik at ikke alle alternativene nødvendigvis vises når filmklippet er ferdigspilt. Disse valgene kan kun vises en begrenset periode, og det må angis når de skal dukke opp og hvor lenge de skal vises. Om mottakeren ikke trykker på alternativet forsvinner det og det er da ikke mulig å velge lenger. Det må nevnes at det må være minst to valg som vises for mottakeren når filmklippet er ferdigspilt.

Det er og ønskelig å kunne angi om valgene som skal vises når filmklippet er ferdigspilt skal begynne å bli vist før filmklippet er helt ferdigspilt. Da må det angis hvor lenge før slutt disse valgene skal begynne å bli vist.

Det skal også være mulig å sette et av valgene, som vises til slutt, som standardvalg. Det vil si at om mottakeren nøler for lenge med å velge, vil spilleren selv velge standardvalget og spille det tilhørende filmklippet. Det valget som eventuelt blir satt som standardvalg må også ha angitt hvor lenge spilleren skal vente før standardvalget automatisk blir tatt. Det bør også være

mulig å velge at standardvalget skal være "usynlig", altså at mottakeren ikke kan se valget i *valgperioden* og ikke kan ta valget på annen måte enn å *ikke* trykke på noen av alternativene og la "nøletiden" utløpe. Da er det viktig at det fortsatt er minst to andre valg som vises for mottakeren i valgperioden når filmklippet er fedigspilt.

Det må også være mulig å redigere informasjonen som vises for mottakere om en filmvisning. Det skal for eksempel være mulig å redigere data som tittel på filmvisningen, beskrivende tekst og bilde/illustrasjon/plakat. Brukeren skal også ha mulighet til å redigere en tekst som utgjør rulleteksten til filmvisningen som mottakeren vil se når han/hun har sett seg gjennom en hel filmvisning.

For at filmklippene skal kunne vises i filmviseren må filmklippene være i et filformat, og bruke en codec[1], som spilleren kan spille av. Siden veldig mange nettbrukere har en Flash-plugin[2] installert på klienten sin og siden en filmavspilling gjennom en Flash-fil ikke krever å åpne noen andre programmer enn nettleseren er det derfor hensiktsmessig og ønskelig å lage avspilleren ved hjelp av Adobe Flash[3]. For å spille av filmfiler på en dynamisk måte i Flash må filmfilene være i FLV-format[4]. Derfor må også alle filmklipp som lastes opp til tjeneren automatisk transkodes til FLV-format, for at brukeren skal slippe å tenke på eller ha kunnskap om transkodingen.

For at andre lett skal få en oppfatning av hvilke muligheter som ligger i dette systemet er det og ønskelig å lage en interaktiv film som viser potensialet systemet har. Å vise en slik filmvisning, som helt fra planleggingsfasen er ment å være nettopp en *interaktiv* filmvisning, vil nok også være den enkleste måten å forklare utenforstående hva som menes med konseptet *interaktiv film*.

B.2.2 Avgrensing

Siden prosjektet er relativt stort og det kun er *en* student som skal gjennomføre det, er det ikke sikkert vi har tid til å implementere all funksjonaliteten som er ønskelig. Derfor vil arbeidet først konsentreres rundt hovedfunksjonaliteten som filmavspiller med de nødvendigste funksjonene, brukersystem med inn- og utlogging, oppretting og redigering av valgtrær/filmvisninger, opplastning av filmklipp og transkoding av disse til FLV-format.

Når alt dette er gjort vil utbedringen begynne i form av mer funksjonalitet i filmspilleren og bedre brukervenlighet i redigeringsapplikasjonen for valgtrær, utvidet funksjonalitet i brukersystemet som blandt annet gir mulighet til å invitere nye brukere, og generelt penere layout i spilleren og etter hvert alle andre ledd av systemet.

Når det gjelder å produsere en film vil dette bare bli gjort hvis hovedfunksjonaliteten og filmproduksjonen anses som gjennomførbar innenfor tidsrammen. En del kompromisser vil også kunne gjøres i filmproduksjonen med tanke på kompleksitet, varighet og generell kvalitet. Uansett vil planleggingen av filmen være en tilleggsaktivitet som går parallelt med resten av prosjektet, og ikke avsettes mange timene per uke. Innspilling og redigering bør ikke trenge å ta lengre tid en én uke til sammen.

[1] **Co**der-**Dec**oder - http://en.wikipedia.org/wiki/Codec
[2] Flash Player - http://www.adobe.com/products/flashplayer/
[3] Adobe Flash - http://www.adobe.com/products/flash/flashpro/
[4] FLV: Flash video

B.3 Prosjektorganisering

B.3.1 Roller og ressurser

Hovedprosjektet utføres kun av Jon Vatne fra klassen 04HBMETEA. Under hovedprosjektperioden vil arbeidet utføres på grupperommet A034. Prosjektet har fått tildelt to datamaskiner av IT-tjenesten som vil benyttes i prosjektperioden.

Oppdragsgiver Høgskolen i Gjøvik.

Prosjektveileder Øyvind Kolås

B.3.2 Rutiner og regler

- Gjøremål skal daglig loggføres.
- Mesteparten av arbeidet vil bli utført på grupperommet. Arbeidsdagene vil normalt være ukedager fra kl.10:00 til 16:00.

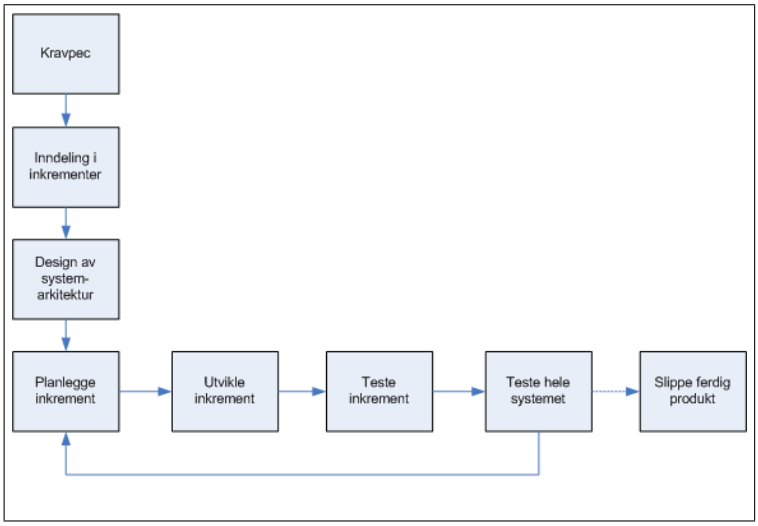
B.4 Planlegging, oppfølging og rapportering

B.4.1 Utviklingsmodell

I dette prosjektet er det, på grunn av arbeidet som er utført høsten 2006 i emnet Fordypning i Medieteknikk og på grunn av at mange av ideene rundt prosjektet kommer fra den som utfører prosjektet, lett å spesifisere kravene tidlig. Systemet kan lett deles inn i inkrementer siden det er flere klart definerbare prosesser og applikasjoner som utgjør systemet.

På grunnlag av dette var det naturlig å velge en inkrementell utviklingsmodell. Ved starten av hver iterasjon vil hva som skal gjøres i det aktuelle inkrementet bli avklart og planlagt i detalj. Det vil også være lett å tidlig se om man ligger etter i forhold til framdriftsplanen, og dermed forhåpentligvis kunne forhindre at man ikke har et fungerende produkt på slutten av prosjektperioden. På slutten av iterasjonene vil den nylig utviklede delen av systemet bli implementert og testet med resten av den ferdige løsningen.

En fossefallsmodell ble utelukket med tanke på at *all* kravspesifikasjon da må avklares før man begynner å utvikle systemet. Vi ser ikke bort fra at nye ideer vil dukke opp underveis, og med en fossefallsmetode vil det ikke være mulig å finpusse de aktuelle delene av kravspesifikasjonen og designskissen før vi setter i gang utviklingen av en spesifikk del av løsningen. XP-modellene ble også vurdert, men ble valgt bort siden flere viktige practices i XP ikke passet for dette prosjektet.



Figur B.1: Prosjektmodell

B.4.2 Planer for statusmøter

Tirsdag hver uke kl.11:00 skal det avholdes statusmøte med veileder hvor det vises framgang siden sist og status i forhold til Gantt-skjema.

B.5 Organisering av kvalitetssikring

B.5.1 Programmeringsspråk

- ActionScript 2.0 er programmeringsspråket som brukes i Adobe Flash (versjon 8, og Macromedia Flash 7), og vil naturligvis derfor bli brukt til programmeringen i filmviseren.
- På tjenersiden vil PHP 5[5] bli brukt som språk. JSP[6] ble lenge vurdert som det beste alternativet, med tanke på JSP kan håndtere DOM-strukturer på en bedre måte når endringer skal gjøres i et `session`-tidsløp. En annen grunn til at vi anså JSP som den beste løsningen var at det ble sett på som nødvendig å bruke JSP for å ta imot filoverføringen fra Java Applet-programmet fra klienten. For å ha kontuinitet i systemet burde da også bare JSP brukes. I teorien er det ikke tilfellet at JSP må brukes til å ta imot filoverføringen, et hvert tjenersideprogrammeringsspråk skal i utgangspunktet greie å ta imot og sette sammen flere pakker med binærdata. Det som var mest avgjørende for valget av språk, for webutviklingen, var at JSP er et veldig omfattende og komplisert i forhold til PHP, som på sin side er mye bedre kjent for den som skal utvikle systemet.
- Java[7] er et plattformuavhengig programeringsspråk som relativt enkelt kan implementeres som en Java Applet[8] i nettleseren og vil derfor bli brukt til å lage klienten som stykker opp og laster opp filmklipp til tjeneren.
- Unix-shellet bash[9] vil brukes for å scripte transcoding og håndtering av filmklipp på tjenersiden.

B.5.2 Plattform

- Plattformen som kjøres på tjenerne vil være Linux-distrubusjonen Ubuntu GNU. Grunnen er at Linux er gratis, lite ressurskrevende og ledende på markedet for "internett-hosting".
- Klientenapplikasjonene bør være plattformuavhengige, derfor vil en Java Applet bli brukt for filopplastningsapplikasjonen og Flash for filmavspilling.

B.5.3 Kodekonvensjon

For å gjøre koden så lettlest som mulig vil kodekonvensjonen som er spesifisert i vedlegg E på side 81, benyttes i utviklingen under prosjektperioden.

B.5.4 Dokumentasjonsformat

Rapporter og dokumenter skrives i LaTeX[10], og skrives på norsk. Dokumentene vil lagres i `.tex`-format og bli gitt fornuftige navn, og ved innlevering vil det

[5] PHP: Hypertext Preprocessor - http://www.php.net/
[6] JavaServer Pages - http://java.sun.com/products/jsp/
[7] Java, objektorientert programmeringsspråk - http://java.sun.com/
[8] http://java.sun.com/applets/
[9] Bourne-again shell - http://www.gnu.org/software/bash/
[10] http://www.latex-project.org/

leveres PDF-filer, generert fra `.tex`-filene.

B.5.5 Revisjonshåndtering

Det vil bli brukt Subversion[11] på kode og dokumentasjon. IT-tjenestens Subversion-server (som vi bruker) lagrer repositoriene på NASet[12], som gjør at vi vil nyte godt av IT-tjenestens backup-rutiner, og har dermed redusert behovet for å ta backup lokalt.

B.5.6 Risikoanalyse

Nr	Risiko	Sannsynlighet	Konsekvens	Tiltak
1	Stor arbeidsmengde	Høy	Moderat	Bruke mye tid på prosjektet.
2	Langvarig sykdom	Moderat	Fatal	Ingen løsning.
3	Ikke overholde tidsfrister	Lav	Fatal	God prosjektplan, hyppigere statusmøter, avgrensninger og mer enn planlagt arbeidstimer.
4	Tap av data	Moderat	Moderat	Lagre viktig data på Subversion hos IT-tjenesten.

Tabell B.1: Risikoanalyse

B.5.7 Kritiske suksessfaktorer

- Lage et godt fungerende produkt innen den gitte tidsrammen.
- Lage en god rapport innenfor den gitte tidsrammen.

[11] http://subversion.tigris.org/
[12] Network Attached Storage

B.6 Plan for gjennomføring

Siden det prioteres å først gjøre ferdig hovedfunksjonaliteten vil ikke nødvendigvis all jobbing i for eksempel Flash gjøres i løpet av en iterasjon. Under er en oversikt over hva som skal gjøres, sortert ut ifra avveininger som er gjort med tanke på inkrementets omfang og hvor kritisk inkrementet er for systemet.

1. Sette opp tjenere.

   Har i skrivende stund installert operativsystem og LAMP[13] på begge datamaskinene, mangler å sette opp databasen på webserver-maskinen og installere nødvendig programvare for transkoding på transcoder-maskinen.

2. Utvikle database.

3. Utvikle løsning for transkoding og lagring av filmklipp-filer.

   Hoveddelen av denne jobben er gjort under prosjektet i emnet IMT391, men det må taes valg rundt oppløsning på lyd og video, og mer informasjon må sendes til databasen.

4. Utvikle valgtre-redigeringen med fokus på funksjonalitet og helt uten fokus på brukervennlighet eller design.

5. Utivkle hovedfunksjonaliteten for filmspilleren.

   Mye av denne jobben er gjort under prosjektet i emnet IMT391, men det bør ryddes opp i kode og utbedres feil siden mange av løsningen i det forrige prosjektet var veldig "hårete".

6. Utvikle filopplastningapplikasjonene.

7. Videreutvikle filmspilleren.

8. Videreutvikle valgtre-redigeringen.

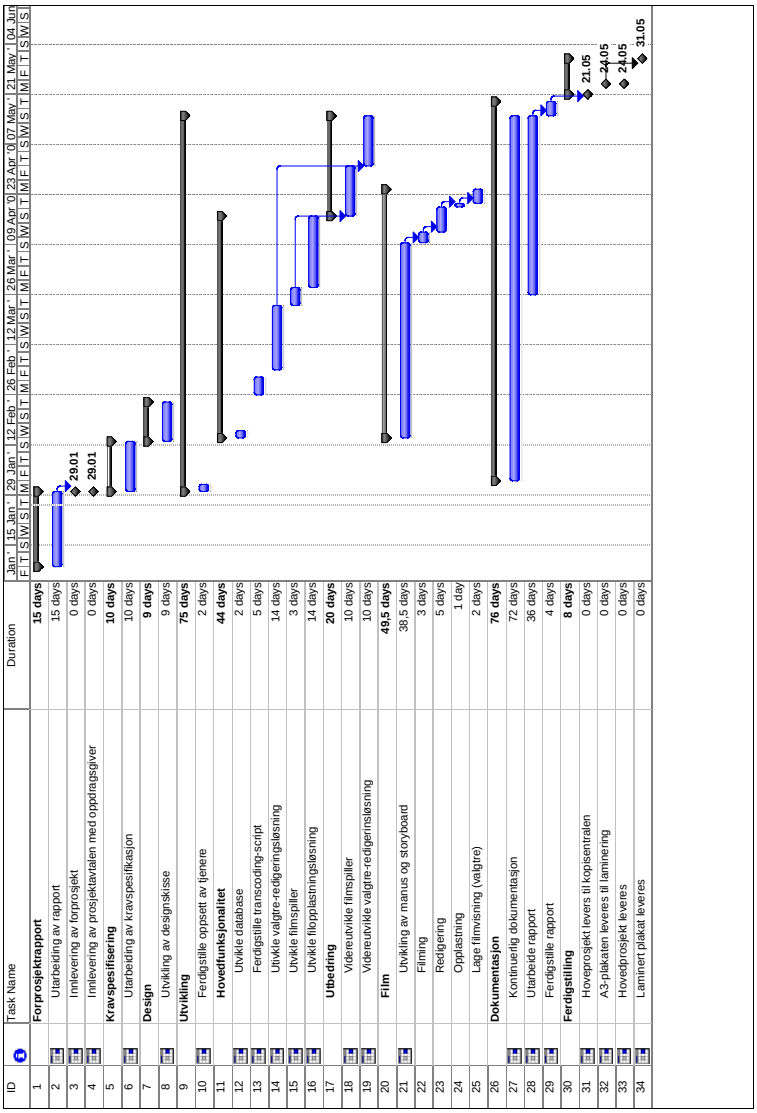
Utvikling av manus og storyboard vil foregår paralellt med andre oppgaver.

[13] LAMP (Linux, Apache, MySql og PHP/Perl/Python), software bundle som brukes på webservere for å skape dynamiske nettsted

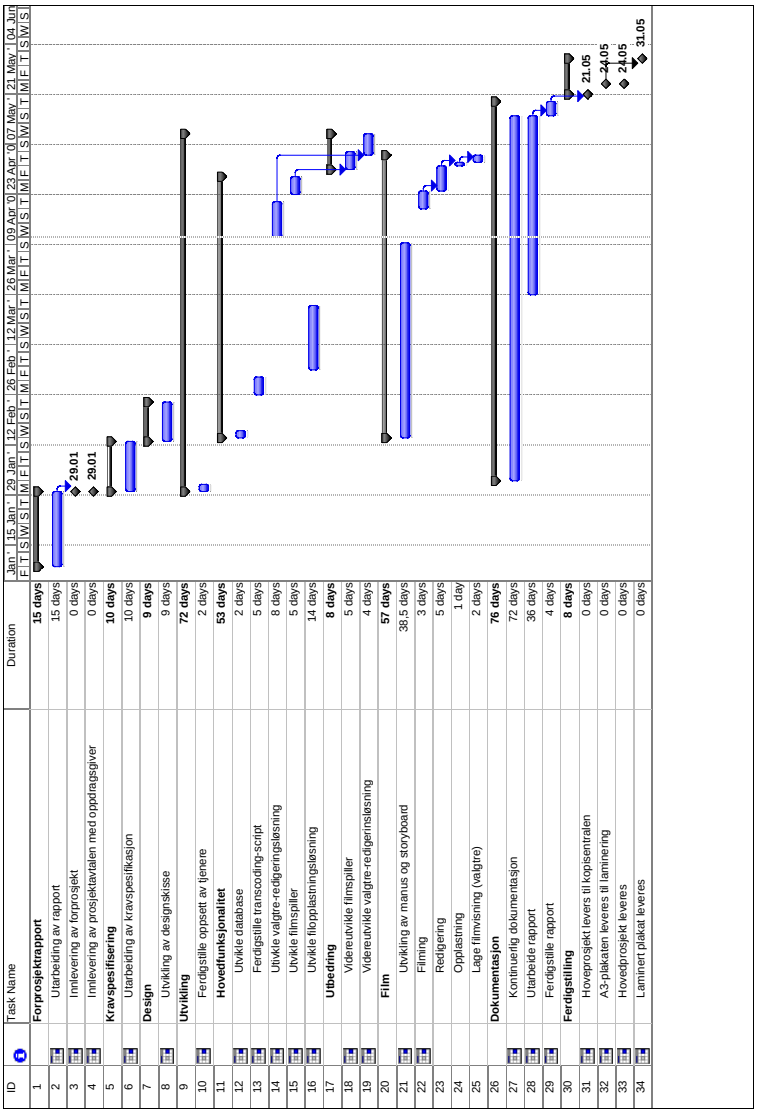
Tillegg C

Framdriftsplan

C.1 Gantt-skjema



C.2 Revidert gantt-skjema anno 11.04.07



Tillegg D

Use cases

D.1 Overordnede Usecase-beskrivelser

For alle Use Cases hvor *bruker* er aktør er det en pre-betingelse at brukeren er registrert i databasen og logget inn i systemet.

D.1.1 Se/bruke interaktiv filmvisning

Mål Vise interaktiv film, med valg, i henhold til informasjonen lagret valgtre (filmvisningstrukturen).

Aktør Mottaker

Beskrivelse Filmavspilleren laster inn den aktuelle valgtrestrukturen med alle valg og filbaner til filmklippene blandt annet. Spilleren viser første filmklipp (om det skal vises et filmklipp først) og viser alle eventuelle valg som skal vises underveis i filmen, og på slutten viser alle valg som skal vises da. Ut ifra hvilket valg brukeren tar går spilleren videre ned i valgtreet og viser etterfølgende filmklipp og valg. Når spilleren ikke har flere valg å spille skal den vise rulletekst.

Pre-betingelser

1. Mottakeren har en kompatibel Flash-spiller.
2. Mottakeren er på en nettside som inneholder den interaktive filmavspilleren.
3. Parameter blir sendt med til filmavspilleren for å fortelle hvilket valgtre den skal bruke.

Post-betingelser

1. Antall visninger av enkeltklipp og visninger av den aktuelle filmvisningen blir oppdatert i databasen.
2. Brukeren har kunnet gi karakter på filmvisningen.

D.1.2 Laste opp filmklipp

Mål At brukeren kan laste opp hvilken som helst type filmfil fra sin klient og inn i systemet på tjenersiden.

Aktør Bruker

Beskrivelse På systemets nettsider skal det implementeres en applet, som er et program som i realiteten lastes ned til og kjøres fra klientsiden. I dette programmet kan brukeren velge en fil fra klienten som skal lastes opp til tjeneren. Når brukeren har valgt en fil sjekker programmet om filen er en kompatibel filmfil. Om det er det stykker den opp filen og sender den til tjener i "passelig små porsjoner". Tjenersystemet på sin side tar imot alle disse fragmentene av filen og setter de sammen slik den i utgangspunktet var på klienten. Så transkoder tjenersystemet filen til en FLV-fil og registrerer dette i databasen.

Pre-betingelser

1. Brukerens nettleser har støtte for den type applet som brukes.
2. Brukeren tillater kjøring av applet'en som sender filen fra klienten.

Post-betingelser Klippet skal kunne vises og legges til i forbindelse med ajourføring av valgtrær/filmvisninger.

D.1.3 Ajourføre filmklipp

Mål Gi mulighet for brukeren til å redigere tittel og beskrivende tekst til filmklipp, og mulighet til å slette filmklippet.

Aktør Bruker

Beskrivelse Bruker skal ha mulighet til å enkelt se en oversiktlig visning av alle ferdig opplastede og transkodede filmklipp, samt og enkelt ha mulighet til å spille av disse klippene og gi de besrivende tittel og en mer utfyllende tekst. Denne tittelen og teksten vil kun vises for den innloggede aktuelle brukeren. Brukeren skal også ha mulighet til å slette opplastede filmklipp, denne funksjonen bør ikke kunne utføres ved en feiltagelse.

Pre-betingelser For at et filmklipp skal kunne slettes kan det ikke være brukt i noen filmvisning.

Post-betingelser Tittel og beskrivende tekst skal vises når oversikt over brukerens opplastede filmklipp vises i *ajourføring av valgtrær*.

D.1.4 Ajourføre valgtrær

Mål Opprette og redigere valgtrær som brukes til å kjøre en filmvisning.

Aktør Bruker

Beskrivelse Brukeren skal ha mulighet til å opprette valgtrær. I disse valgtrærne skal brukeren kunne legge til filmklipp og valg, med valgalternativer. For hvert valgaternativ som legges til skal det være et tilhørende filmklipp. Og for hvert filmklipp kan det legges til et tilhørende valg. Et filmklipp skal ikke kunne brukes to ganger i samme unike visning. Brukeren skal kun kunne legge til filmklipp han/hun har lastet opp og som er ferdig transkodet. Brukeren skal kunne velge om han/hun skal lagre, og jobbe videre på prosjektet seinere (når flere filmklipp er lastet opp for eksempel) eller om filmvisningen skal publiseres.

Om brukeren velger å publisere filmvisningen må han/hun sette tittel på filmvisningen og har mulighet til å legge til bilde, rulletekst beskrivende tekst om filmen.

Post-betingelser Når et valgtre er publisert skal filmvisningen være tilgjengelig på nettsiden for alle som vil se.

D.1.5 Se filmklipp

Mål Bruker skal enkelt få oversikt over innholdet på det aktuelle filmklippet.

Aktør Bruker

Beskrivelse Brukeren skal enkelt kunne få oversikt over hva videoklippet inneholder. Dette betyr at det skal være enkelt og raskt å kunne starte og stoppe (samt å spole i) en avspilling av et klipp.

Pre-betingelser Brukeren har en kompatibel Flash-spiller.

D.1.6 Ajourføre persondata

Mål Bruker skal kunne endre data om seg selv.

Aktør Bruker

Beskrivelse Bruker skal ha mulighet til å endre passord, fornavn, etternavn, epost og personlig bilde. Det skal ikke være mulig å endre passord uten å oppgi gammelt passord. Passord skal også kamufleres når det skrives inn slik at andre ikke kan se hva brukeren skriver. Passord skal også lagres kryptert i databasen.

Post-betingelser Brukerens endringer skal vises på profilvisningen, til brukeren, for mottakerne.

D.1.7 Invitere ny bruker

Mål "Rutinerte" brukere skal kunne invitere nye brukere.

Aktør Bruker

Beskrivelse Brukere som har publisert filmvisninger, kan invitere like mange nye brukere som filmvisnigner han/hun har publisert. Dette foregår på den måten at brukeren skriver inn epostadressen til den personen han/hun vil invitere, og en e-post blir automatisk generert og sendt til denne personen. E-posten vil inneholde informasjon om systemet som helhet, invitasjonen og hva det innebærer, og hva som skal gjøres videre for å opprette en brukerkonto.

Moderatorer skal også ha mulighet til å sende slike invitasjoner.

Post-betingelser Personen som har fått tilsendt e-post skal ha mulighet til bli bruker i systemet.

D.2 Detaljerte Usecase-beskrivelser

D.2.1 Se/bruke interaktiv filmvisning

Mål Vise interaktiv film, med valg, i henhold til informasjonen lagret i valgtreet (filmvisningstrukturen).

Aktør Mottaker

Interessenter Brukeren som har lastet opp den interaktive filmvisningen.

Pre-betingelser

1. Mottakeren har en kompatibel Flash-spiller.
2. Mottakeren er på en nettside som inneholder den interaktive filmavspilleren.
3. Parameter blir sendt med til filmavspilleren for å fortelle hvilket valgtre den skal bruke.

Post-betingelser

1. Antall visninger av enkeltklipp og visninger av den aktuelle filmvisningen blir oppdatert i databasen.
2. Brukeren har kunnet gi karakter på filmvisningen.

Main success scenario

	Bruker	System
1		Laster inn valgtrestruktur
2		Laster inn informasjon om filmklipp fra databasen
3		Spiller av første filmklipp i filmvisningen
4		Viser tilhørende valg
5	Velger et alternativ	
6		Spiller av tilhørende filmklipp
7		Viser rulletekst
8		Viser *angi karakter*-visning
9	Angir karakter på filmvisningen	

Varianter

3a Hvis det i valgtreet er angitt at det første som skal vises ikke er et filmklipp, men et valg.

1. Gå direkte til punkt 4.

4a Hvis noen tilhørende valg er satt til å vises *før* slutten av filmklippet:

1. Vis aktuelt valg i angitt tidsperiode parallelt med visningen av filmklippet.
   (a) Hvis mottaker trykker på det aktuelle valget.
      i. Avslutt visning av filmklippet som nå spilles og skjul valg.
      ii. Start avspilling av tilhørende filmklipp.
   (b) Hvis mottaker ikke trykker på valget iløpet av den angitte tidsperioden.
      i. Fortsett visning av filmklippet som nå spilles og skjul valg.

5a Mottakeren nøler så lenge med å velge at angitt tidsbegrensning[1], for valgperioden, utløper.

1. Ut fra hvilket valg som er satt som *standardvalg*, blir det tilhørende filmklippet spilt av (gå til punkt 6).

6a Hvis det er flere valg nedover i hierarkiet, i den aktuelle delen av valgtreet:

1. Vis aktuelt valg og etter hvert tilhørende filmklipp også videre (gå til punkt 4).

Unntakshåndtering

1a Valgtre er ikke sendt med som parameter.

1. Vis feilmelding.

1b Valgtre er ikke lagret på tjener.

1. Vis feilmelding.

2a Databasespørring er mislykket.

1. Vis feilmelding

7a Ingen rulletekst er lagret for det aktuelle valgtreet.

1. Gå videre til punkt 8.

[1] Det er ikke noe krav å sette noen tidsbegrensing for et valg.

D.2.2 Laste opp filmklipp

Mål At brukeren kan laste opp hvilken som helst type filmfil fra sin klient og inn i systemet på tjenersiden.

Aktør Bruker

Pre-betingelser

1. Brukerens nettleser har støtte for den type applet som kjøres.
2. Brukeren tillater kjøring av applet'en som sender filen fra klienten.

Post-betingelser Klippet skal kunne vises og legges til i forbindelse med ajourføring av valgtrær/filmvisninger.

Main success scenario

Punktene fra og med punkt 11 er prosesser som skjer automatisk på tjeneren.

Varianter

7a Filfragmentet er ikke det første som kommer fra den aktuelle filen.

1. Filfragmentet legges til filen, som er i ferd med å bygges opp av fragmentene som sendes fra klienten.

9a Hele filen er ikke kopiert til tjeneren i sin helhet.

1. Gjentar prosessen fra og med punkt 6.

11a Ingen filer ligger i `upload`-mappen.

1. Venter et minutt og gjentar punkt 11.

Unntakshåndtering

4a Filen er ikke en kompatibel filmfil.

1. Viser feilmelding og lar brukeren velge en annen fil.

7a Nettforbindelsen bryter under overføring.

1. Gir tilbakemelding til klientapplikasjonen.
2. Klientapplikasjonen sender samme fragment på nytt.

	Bruker	Klientsapplikasjon	Tjenersystem
1			Viser applet
2		Lastes ned og kjøres fra klienten	
3	Velger fil		
4		Validerer fil	
5		Stykker opp fil	
6		Sender fil-fragment	
7			Tar imot fil-fragment
8		Viser fremgang visuelt for brukeren	
9			Legger den ferdige filen i en `upload`-mappe
10			Legger data om filen i DB[2]
11			Lager en tilhørende meta-datafil
12		Gir tilbakemelding	

	Bruker	Kl.app.	Tjenersystem
13			Leter etter filer i `upload`-mappen
14			Flytter alle filer fra `upload`-mappen til en `work`-mappe
15			Legger data om filene til i DB
16			Begynner konvertering av fil til FLV-format og oppdaterer data om filen i DB
17			Flytter FLV-filen til en "*ferdig*-mappe" og sletter orginalfilen
18			Oppdaterer data om filen i DB
19			Gjør det samme som fra og med punkt 16 med alle gjenværende filer i `work`-mappen
20			Går tilbake til punkt 13

Tillegg E

Kodekonvensjon

- Variabel- og klassenavn skal følge en standard der klassenavn begynner med stor bokstav og variabelnavn som inneholder flere ord skrives `longVariable`.
- Konstanter skal alltid skrives med store bokstaver. eks: `IMPORTANT_CONSTANT`

Funksjoner

```
func(x, y){
    ...
}
```

Kontrollblokker

```
if ( <expression> ){
    ...
}
```

Klasser og instansiering av klasser

```
class Foo
{
public:
    void aFunction();
private:
    char someMoreData;
    void yetAnotherFunction(data);
};
```

Tillegg F

Kodeeksempler

Her er noen eksempler på hordan deler av noen av noen av prosessene har blitt kodet.

Utelatt kode er vist med tre punktum (...) og linjeskift.

F.1 ta_imot.php

Denne filen tar imot fil-chunks fra JavaApplet'en.

```
<?php
...

$fileending   = '.fvm'; //fileending of the filmclip-metadata-files
$CHUNKSIZE = 1024*256; //upload chunksize in bytes
$ABS_PATH_START = '/var/www/';
...

if(isset($_POST['chunk_nr'])) {
  if(isset($_FILES['blob']) ) {
    $new_file = $_FILES['blob']['name'] ;
    $chunk_nr = $_POST['chunk_nr'] ;

    //find out who is uploading
    //This query gets the ID of the user with the corresponding uploadpass
    $sql = 'SELECT uploadpasses.userid as userid
            FROM uploadpasses
            LEFT JOIN users ON uploadpasses.userid = users.ID
            WHERE uploadpasses.pass = ?
            LIMIT 1';
    $data = array(("".$_POST['uploadpass']));
    $sth = $db->prepare($sql);
    $resultat = $db->execute ($sth, $data);
    ...
    if ($rad = $resultat->fetchRow()) {
      $user_id = $rad['userid'];
      //make the user-directory if it doesn't excist
      if(!mkdir(($NEW_PATH.$user_id), 0770) && !is_dir($NEW_PATH.$user_id)){
        fprintf($f, "\tFeil: Mappe \"%s\" finnes ikke og greide
ikke lage den\n", ($NEW_PATH.$user_id));
      }else{
        chmod(($NEW_PATH.$user_id), 0770);
        $NEW_PATH = $NEW_PATH.$user_id.'/';
      }
      $new_filepath = $NEW_PATH.$new_file;
  //get total size of uploading file
      $total_size = $_POST['total_size'];
      //calculate number of chunks to be uploaded
      $chunks_total = ceil($total_size/$CHUNKSIZE);
  //if this is the first chunk
        if($chunk_nr <= 0){
        //  register the file i the database
//    with filepath, userid, number_of_upload_chunks
          $sql = 'INSERT INTO movieclips (filepath, userid, status,
date_added, total_upload_chunks, size) values (?, ?, 0, now(), ?, ?)';
          $data = array($new_filepath, $user_id, $chunks_total, $total_size);
```

```
          $sth = $db->prepare ($sql);
          $resultat = $db->execute ($sth, $data);
          ...
          //  get the clip_id of the clip we just registered
          $sql = 'SELECT LAST_INSERT_ID() as clipID FROM movieclips';
          $resultat = $db->query($sql);
          $rad = $resultat->fetchRow();
          $clip_id = $rad['clipID'];
          //  make new file so it overwrites files with the same name
          if (!$handle2 = fopen($new_filepath, 'w')) {
            ...
          }
        }else{
          //if it's not the first chunk
          //  open file so we can put stuff at the end
          if (!$handle2 = fopen($new_filepath, 'a')) {
            ...
          }
        }
        //read data from the uploaded file
        if(!$fp=fopen(($_FILES['blob']['tmp_name']), 'r')) {
          ...
        }
        $innhold=fread($fp, $_FILES['blob']['size']);
        fclose($fp);
        //write data into the new file
        if (fwrite($handle2, $innhold) === FALSE) {
  ...
        }
        //close the new file
        fclose($handle2);
//if it's the first chunk
        if($chunk_nr <= 0){
        // make movieclip-metadatafile. it cointains clip_id and user_id
          if(!file_put_contents(($new_filepath.$fileending),
"clip_id: $clip_id\nuser_id: $user_id")){
            ...
          }
        }else{
          // if it's not the first chunk
          //   get the clip id from the movieclip-metadata-file (.fvm)
          if(!$fvmf=fopen(($new_filepath.$fileending), 'r')) {
            ..
          }
          $buffer = fgets($fvmf);
          fclose($fvmf);
          $bufpieces = explode(" ", $buffer);
          $clip_id = $bufpieces[1];
        }
        //update database
```

```
        $sql = 'UPDATE movieclips SET uploaded_chunks =? WHERE ID=? LIMIT 1';
        $data = array(($chunk_nr+1), $clip_id);
        if($chunk_nr >= ($chunks_total-1)){
          //if this is the last upload-chunk of the file
          //  TODO: check if the file has not been corruptet
          //  move the movie- and metadata-file to the uploaded-directory
          if(rename($new_filepath, ($FINISHED_PATH.$new_file))){
            //  and the metadatafile
            if(rename(($new_filepath.$fileending),
($FINISHED_PATH.$new_file.$fileending))){
              // alter SQL-query
              $sql = "UPDATE movieclips SET uploaded_chunks=?, filepath=?,
                      status = '1' WHERE ID=? LIMIT 1";
              $data = array(($chunk_nr+1), ($FINISHED_PATH.$new_file), $clip_id);
            }else
              fprintf($f, "\t FEIL: %s\n", ("greide ikke flytte metadatafil"));
          }else
            fprintf($f, "\t FEIL: %s\n", ("greide ikke flytte filmclipfil"));
          }
          //  update database
          $sth = $db->prepare ($sql);
          $resultat = $db->execute ($sth, $data);
          ...
    }else{
  ...
    }
  }
}else
  ...
?>
```

F.2 GraphEditor.php

Her kommer et utvalg av koden i `GraphEditor`-klassen, som viser hvordan noen av endringene til en valg-graf blir utført.

```
<?php
...
include_once('GraphHandler.php');
...
/**
 * This class is in charge of the GUI of the graph editing.
 */
class GraphEditor {

    ...

    /**
     * This function controls what is being shown to the user, and makes sure
     * that changes get saved.
     */
    function display() {
      if (isset($_SESSION['userid']) && $this->xmlfile != NULL) {

      ...

      }if(isset($_POST['submit_edit_alternative'])){
        $this->displayBackToActivenode();
        $this->displayAlternativeEdit($_POST['active_node'],
$_POST['option_nr'], $_POST['alt_nr']);
      }else if(isset($_POST['submit_add_alternative'])){
        $this->displayBackToActivenode();
        $this->displayAlternativeEdit($_POST['active_node'],
$_POST['option_nr'], '');
      }else if(isset($_POST['submit_save_alternative'])){
        $this->saveAlternativeText($_POST['active_node'], $_POST['option_nr'],
$_POST['alt_nr'], $_POST['text']);
      }

      ...

    }

    ...

    /**
     * Displays the given node's branch's given link text editor.
     * @param node_id a string, node identifyer
     * @param option_nr an integer, branch identifyer
     * @param alt_nr an integer, link identifyer
     */
```

```
    function displayAlternativeEdit($node_id, $option_nr, $alt_nr) {
      if($alt_nr != 'new' && $alt_nr != '' && isset($alt_nr)){
        //get alt_text
        $this->graphHandler->load($this->xmlfile);
        $alt_text = $this->graphHandler->getLinkValue($node_id, $option_nr,
$alt_nr);
      }
      echo "<form method=\"POST\">\n";
      echo "  <input type=\"hidden\" name=\"active_node\" value=\"".$node_id.
"\">\n";
      echo "  <input type=\"hidden\" name=\"option_nr\" value=\"".$option_nr.
"\">\n";
      echo "  <input type=\"hidden\" name=\"alt_nr\" value=\"".$alt_nr."\">\n";
      echo "  <p>Alternative text: <input type=\"text\" name=\"text\"
value=\"".$alt_text."\"></p>\n";
      echo "  <input type=\"submit\" name=\"submit_save_alternative\"
value=\"Save\" /></form></p>\n";
      echo "</form>";
    }

    /**
     * Sets a new link text to a given node's given branch's link.
     * @param node_id a string, node identifyer
     * @param option_nr an integer, branch identifyer
     * @param alt_nr an integer, link identifyer
     * @param alt_text a string, the new link text value
     */
    function saveAlternativeText($node_id, $option_nr, $alt_nr, $alt_text) {
      $this->graphHandler->load($this->xmlfile);
      $this->graphHandler->editAlternative($node_id, $option_nr, $alt_nr,
$alt_text, 'dontchange');
      $this->graphHandler->save($this->xmlfile);
      $this->displayNode($node_id);
    }

    ...

}
?>
```

Tillegg G

Logg

09.01.06 Møte med kolås. Kartla hva som skulle gjøres på et generelt nivå mtp forprosjektrapport og endelig rapport. Avtalte at jeg skulle møtes om en uke og at jeg til da skulle ha lagd en skisse av forprosjektrapporten.

10.01.07 Formaterte laptop, og installerte diverse software

11.01.07 installert LaTEX og TeXnicCenter og satt meg litt inn i dette. har også begynt så smått på forprosjektrapport-skissa

12.01.07 satt meg inn i LaTEX og kartlagt ytterligere hva som skal gjøre i prosjektet (kravspek.) mtp forprosjektrapporten kl.1330->1830

15.01.07 kl.1130->1830 satte meg ytterligere inn i LaTEX og kartla og spesifiserte mer av det som skal gjøres i prosjektet.

16.01.07 kl.1100->2000 Møte med Kolås: Kartla ytterligere hva som skal være med i forprosjektrapporten og fikk låne et par eksempler. Mulige løsninger og ny funksjonalitet ble også diskutert (jsp vs php, og hvordan avspillingen kan skje mtp på at det kan være ønskelig at noen valg vises tidligere i filmklippvisningen enn de andre). Fikk også et godt tips angående valgtre-håndteringen som gikk ut på at først bør det vektlegges at funksjonaliteten virker og heller jobbe med brukervennlighet når funksjonaliteten er i boks.

Satte også opp de to datamaskinene som skal brukes som servere (transcoder"og "webserver") med Ubuntu operativsystem, LAMP (Linux, Apache, MySql, PHP) serversystem, Secure Shell (ssh) for sikker tilkobling fra eksterne maskiner.

17.01.07 kl.0900->1700 Diskutert med Ø.Kolloen hva som er mest hensiktsmessig å bruke av PHP og JSP. Har endret syn på hvordan løsningen bør være, mente i utgangspunktet at JSP burde være måten å gjøre det på, men fordelene med PHP taler for PHP.

Fordeler med PHP: Mer erfaring med PHP enn JSP. Mye mer lettvindt (ikke så omfangsrikt) og enklere enn JSP. Slipper kompilere for hver endring som gjøres. At JSP potensielt er mer effektivt vil nok ikke ha så mye å si i dette

systemet (hadde det vært Google-størrelse på systemet så kanskje... youtube bruker f.eks. php) Siden valgtreredigeringssiden i første omgang vil reloades for hver endring vil det nok være like greit å jobbe rett på et XML-dokument, og da vil det nok være enklere å bruke PHP.

Ulemper med å velge PHP: Jeg ser for meg at det vil være vanskelig å lage et server-program som tar imot flere filer og setter de sammen til en fil i PHP, selv om det i teorien høres enkelt ut. Siden det er mulig i JSP å sette klasser til å ha en varighet på en session ville det vært bedre å lagre unna DOM-strukturen mens man jobber på den i JSP enn noen løsning i PHP.

19.01.07 kl.1700->1830 Jobbet litt mer med forprosjektrapport og ls i LaTEX

22.01.07 kl.1030->1200 & 1500->1900 Skrevet på forprosjektrapporten.

23.01.07 kl.1000->1915 Møte med Kolås. Pratet om bl.a. om kodekonvesjoner og revisjonshåndtering. og hvordan jeg bør skrive om det som allerede er gjort i fordypning. i omfangsdelen bør alt som skal gjøres for å få et ferdig resultat stå, og i plan for gjennomføringbør det stå hva som allerede er gjort.

Har også satt opp repository.

24.01.07 kl.1000->2100 Skrevet forprosjektrapport. Og begynt på gantt-skjema. Har også opprettet en midlertidig nettside som skal vise prosjektinfo, men som nå bare viser prosjektnavn og mitt navn. Har også lagd en tegning av prosjekmodellen.

25.01.07 kl.1030->1830 Klippet fra SVN-logg: - skrevet forprosjektrapport.tex nesten ferdig. en del finpuss gjenstår - gjort gantt-prosjektplan.mpp så og si ferdig - redigert ProsjektmodellTegning.vsd for å gjøre den litt finere - opprettet ProsjektmodellTegning.eps for å kunne implementere denne illustrasjonen i forprosjektrapport.tex.

26.01.07 kl.0900->1800 Jobbet med forprosjektrapport.

28.01.07 kl.2230->2300 Lagd kodekonvesjon

29.01.07 kl.0930->1200 ferdigstilte forprosjektrapporten, og oppdaterte prosjektnettside

30.01.07 kl.1000->1845 satte opp PHPadmin for styring av MySQL-databasen. lagd use case-diagram lagd databasediagram (ER-diagram)

31.01.07 kl.0900->1530 Vært på Flash-kurs med mediedesign 2.året

01.02.07 kl.1200->1800 Skrev Use Cases

02.02.07 kl.1200->1400 Flash-kurs med mediedesign 2.året

26.02.07 jobbet med databasen

27.02.07 kl.1030->1430 jobbet mer med brukersystemet, målet var å sette opp en greie som kunne lage metadatafil om filmklipp og legge til filmklipp i databasen, for å gjøre klart for å gjøre ferdig transkodingsbiten, men så langt kom jeg ikke. måtte på fargelab-øving kl.1430

28.02.07 kl.0945->1645 jobbet en del med det samme som i går. tenkt en del rundt problematikken rundt at jeg jobber på to forskjellige servere

01.03.07 kl.1000->1615 jobbet med nisse.sh og ferdigstilte den manuelle registreringen av et filmklipp (med databaseregistrering og metadatafil-oppretting)

02.03.07 kl.0900->2030 jobbe som f*** for å få trancodinga til å funke, gikk egentlig relativt greit, men brukte lang tid på å få filene til å bli lagt på webserveren. problemet er at den ikke vil opprette en mappe på webserveren. etter JÆVLI mye tull har jeg nå satt at gruppa har skrivetilgang, men det virker fortsatt ikke!!!

04.03.07 kl.1445->1730 Endelig fått transkodinga til å virke som den skal. problemet var at scp ikke kunne opprette noen nye mapper. brukte derfor ssh til å lage mappen før, sleit litt med dette, men fikk det til

05.03.07 kl.0900->1145 (fargestyring) kl.1515->1830 (flash-kurs) jobbet med å få transkodinga til å virke enda bedre, og å vise output i brukersystemet

06.03.07 kl.0900->2000 fiksa sånn at lengden på klippene lagres i databasen. mangla et dollartegn i bash-koden :S

fant også en annen bug, 64 blir lett for lite tegn når man skal lagre filbanen. så.. hele filbanen har ikke blitt lagret før nå.

har også gjort om litt på status-systemet til movieclips. alle de gamle satusene har nå en verdi mer, og 0 betyr nå at filen er i ferd med å lastes opp

har begynt på opplastningsystemet i form av at jeg har lagd en fil som kopier data fra en fil til en annen, lokalt, chunk for chunk. dette simulerer på en måte opplastningen, og vil det jeg har lagd vil blir brukt til å ta imot filen på tjenersiden

07.03.07 kl.0900->1700 forsker på java applet-tingen. funnet kode for å lese fil og laste opp fil til server. nå må jeg bare få til å lage en applet hvor man kan velge en lokal fil.

08.03.07 kl.0900->1800 jobbet med å utvikle java applet. har fått til å velge en lokal fil og lese data fra denne.

09.03.07 kl.0900->1845 forsket en del på ressursene appleten bruker. har funnet ut at jeg ikke kan laste inn hele filmfiler i appleten siden nettleseren bare har en viss mengde minne. har også funnet ut at ingen ressurser brukes selv om filestreamen står åpen, dvs at denne ikke trenger å åpnes og lukkes for hver chunk-overføring, noe som er en fordel.

har greid å sende post-data til en php-fil på tjeneren! jippi! er bare at noe klikke. skal prøve å finne ut hva det var.

oppdaget en ny problematikk: data som sendes som postvariabler sendes som string. problematikken er da hvordan man skal sende byte-arrayen til server. løsningen blir å encode byte-arrayen til ut8, for så å smelle dette sammen til en string. sende stringen. dele stringen i chunks på 4 tegn og sette disse i en array. decode utf8 til binærtall. og skrive disse til fil. har ikke lest om denne måten å gjøre det på, bare resonert meg fram til den. har også funnet nettsider som forteller hvordan jeg gjør de forskjellige tingenene.

12.03.07 kl.0900->1600 jobbet med appleten

13.03.07 kl.1000->1910 jobbet videre med java applet. har funnet en klasse som laster opp på en god måte.

14.03.07 kl.0900->1945 jobbet med applet og applikasjon på tjenersiden

15.04.07 kl.1030->1200 jobbet med applet og applikasjon på tjenersiden

16.03.07 kl.0930->2100 Nå gjør java appleten jobben sin faktisk

17.03.07 kl.1330->1730 fiksa sånn at bare filer som har MIME-type som begynner på 'image' eller filnavn som slutter på '.wmv' eller '.gvi' kan lastes opp

skulle også fikse progressbar, men har lite peiling på å lage gui i appleter

19.03.07 kl.0945->1230 1600->2045 Fiksa sånn at nisse.sh kjører automatisk. var noe problematisk siden cronjobben ikke skjønte hvor flvmagic lå.

har lagd en phpfil som returnerer data fra databasen om filmklipp i form av xml

har lagd en flashspiller som er ment til preview av de filmklippene en bruker har lastet opp. phpfila som er nevnt over brukes av denne spilleren

har også lagd en funksjon i movieclips.php som viser denne preview-spilleren, og bruker nå denne funksjonen i listClips-funksjonen

20.03.07 kl.0930->1600 fant ut at nå er det på tide å skrive rapport

23.03.07 kl.0930->1630 prøvde å skrive rapport. ikke så mye fremgang

25.03.07 kl.1400->1800 flash-research

26.03.07 kl.0900->1700 rapportskriving

27.03.07 kl.1100->2400 Flash-research. rapportskriving

28.03.07 kl.0900->1700 rapportskriving

29.03.07 kl.1030->1830 Flash-research. Rapportskriving

10.04.07 kl.1200->1600 rapportskriving

11.04.07 kl.1015->1815 rapportskriving og kort møte med kolås angående dette og planer fremmover

12.04.07 kl.1100->2200 rapportskriving. har egentlig konstatert at det jeg allerede har skrevet er godt nok for denne gang, bare endret noen småting.

også gått gjennom en del av koden og kommentert bedre (gjort klart en del for doxygen)

fixet en bug i movieclips.php, nå sorteres filmklippene riktig.

har også begynnt å tenke på arkiteturet til neste inkrement, valgtreredigering. har kommet fram til en del løsninger med tanke på xml-strukturen, som jeg tror vil være til hjelp. føler altså at jeg har kommet fram til mye bedre måter å gjøre ting på enn det som ble gjort i prosjektet i fjor høst. har tegnet masse greier på ark, og lagd en xml-fil (eksempel_paa_xml_struktur.xml), sjekk it out.

13.04.07 (1000->1115) kl.0945->1930 rapportskrivingskurs

prøvd å få klort ned litt mer i rapporten. holder det fortsatt stikkord-style.

begynt å tenke ut en del på det nye inkrementet. og begynt såvidt med xml-håndtering i php. har greid å legge til option inni andre options, med id'er osv.

16.04.07 kl.1100->1850 jobbet med xml/dom-håndtering i php.

17.04.07 kl.1000->1730 møte med kolås: gått gjennom xml-strukturen og funnet ut at den jeg hadde tenkt å bruke var crappy! når jeg uansett skal kunne linke mellom noder andre steder i treet, er det ikke vits i å ha et tre i det hele tatt. strukturen er derfor endre fulstenig, og eksempel på den nye xml-strukturen kan sees i filen eksempel_paa_xml_struktur_3edition.xml. Har også lagd en klasse som har mulighet til å lage alle disse elementene og atributtene. I morgen skal jeg lage en annen klasse, som benytter seg av denne klassen til å utføre det som skal gjøres i de forskjellige use-casene.

18.04.07 kl.0930->1900 lagd GraphHandler og begynnt på Grapheditor. sistnevnte benytter seg av førstnevnte for å manipulere xml-dokumentet. GraphEditor er ment for å ta seg av visningen av redigeringen.

19.04.07 kl.0940->2000 jobbet videre med de to filene nevnt over.

20.04.07 kl.0945->1445 samme som sist

22.04.07 kl.1300->1700 samme som sist. nesten ferdig nå. eneste som egentlig mangler er å gjøre det mulig å legge til movieclip til root-node, og at man kan redigere på flere xml-filer enn bare test.xml. (da er det viktig å sjekke om brukeren har tilgang på denne)

23.04.07 kl.1020->1230 1600->1900 it's all good. bortsett for at jeg bør lage en greie så man kan få oversikt over alle sine grafer, og så gå inn å redigere de. er forresten ikke mulig å endre på ting som allerede er gjort i graphEditoren, men det får jeg ta seinere en gang. har lagt litt til rette for det i graphHandler..

24.04.07 kl.1000->1730 møte med kolås: bestemte meg for å jobbe litt til mer med grafredigeringen, så funksjonaliteten er helt ferdig før jeg går videre. gjorde sånn at det går an å redigere på det meste.

25.04.07 kl.1000->1700 nå går det an å redigere på alt, og legge til og redigere grafer.

det går heller ikke an å slette forskjellig informasjon, men jeg ser ikke på dette som så veldig viktig ennå.

har også fikset en bug angående filopplastning. feil rettigheter ble gitt til mapper som ble opprettet på transcoder-maskinen av ta_imot.php-scriptet, noe som førte til at filene ikke kunne skrives til dette området, og altså ikke lagres. det er hvertfall nå fikset.

i morgen skal jeg begynne å se på flash-greiene.

26.04.07 kl.1100->2015 begynt på flash. gjort ganske masse egentlig. funnet en litt bedre måte å lese xml på. (gjør dette på onload: xml_graph.onLoad = Delegate.create(this, parseXmlGraph);)

driver å lager tre klasser, en klasse som tar vare på alle nodene, en klasse for en node, og en klasse for en branch.

også fikset sånn at det går an å sette et av alternativene i en branch i grafen til å være default.

gjort masse flash i dag. fikk nesten til å lese inn alle data som skulle leses inn. skjønner ikke hvorfor det ikke funker nå, men håper på å finne ut av det i morra.

27.04.07 kl.1530->1700 fiksa sånn at all data leses inn. smooth!

nå har jeg igjen å finne start-node og faktisk vise ting.

30.04.07 kl.1000->1700 programmet finner nå startnode og begynner å spille av første videoklipp

01.05.07 kl.1130->1530 programmet finner ut når branches skal vises

02.05.07 kl.1000->1930 programmet finner nå ut hvor lenge branches skal vises. mye trøbbel rundt dette. også mye trøbbel rundt å vise branch på slutten av filmklippet, men har lagd en slags løsning på det. har såvidt begynt å vise linker, og det er faktisk mulig å trykke på de og komme til en nettside eller en

annen node. har igjen å teste dette, å fjerne knapper når de skal det, og gjøre det hele penere, og sikkert en hel del annet.

03.05.07 kl.1030->1830 begynner å nærme meg... programmet er nesten ferdig. noen fixer igjen. må gjøre ting penere da... jah!

04.05.07 kl.1030->1800 gjorde nettopp noe viktig: hver gang en node skal spilles av, kopieres branch-data. på denne måten kan spilleren poppe fra kopien, og dermed bruke branch-dataen igjen senere (når man vil gå tilbake til et tidligere klipp). Har også gjort det mulig å gå tilbake til node man har sett før.

har jobbet en del med å vise en fornuftig verdi på totalt spilt tid. funnet en veldig god løsning nå syns jeg.

legger merke til at programmet krevet mye kraft. kanskje det kan være lurt å sette net på intervalhyppigheten.

fiksa sånn at det går an å mute og "un-mute".

prøvde å få til sånn at knappene gir litt mere respons når man trykker på de, men fikk ikke til noe. veit ikke helt hvordan det bør gjøres.

har også begynt å lage liste over ting som bør gjøre. skal føye til mange flere ting på den lista, så får vi se hva som kan bli gjort, og hva som må forkastes.

05.05.07 kl.1230->1400, 1630->1800 kommentert masse kode, mest mtp doxygen.

06.05.07 kl.1300->1400 kommenterte kode, og fiksa ting mtp doxygen

07.05.07 kl.1000->2030 har fikset aspekt ratio på flashting i dag. tok mye mer tid enn forventa, prøvde, men det ble bare lok. også fikset en del andre bugs. og kartlagt en del ting som bør gjøres.

09.05.07 kl.1030->1630 nå går det an å velge at filmen skal pauses når en branch vises.

det går også an å velge at en default link skal kunne være usynlig.

fikset en del andre småbugs og.

10.05.07 kl.1015->1800 jobbet med rapport. skrevet om flashprogrammet

11.05.07 kl.1045->2000 rapport.

14.05.07 kl.1000->1830 lagd nettside, som skaper sammenheng mellom ting. nå kan man logge seg inn, og redigere sine filmer. samt å se publiserte filmer når man ikke er logget inn. løsningene er ikke optimale, bør forbedres før løsningen eventuelt skal slippes for allmenhete, but hey, it works.

15.05.07 kl.1000->1800 rapport

16.05.07 kl.1145->1845 skrevet rapport

18.05.07 kl.0900->2115 rapport

19.05.07 kl.1130->1900 rapport

20.05.07 kl.1130->1830 rapport

21.05.07 kl.1015->0015 rapportskriving

22.05.07 kl.0930->0315 rapportskriving

23.05.07 kl.1030->0030 rapportskriving

Tillegg H

Mappestruktur på tjener

Alle mappene eies av `www-data`-brukeren. Er det viktig å ta hensyn til rettighetene til en fil eller en mappe vil octalverdien for rettighetene være vist i parantes bak mappe-/filnavnet.

Webserver-masking

```
/var/www/
    movieclips/
        transcoded/
          <brukerid>/     - inneholder alle filmene til den
                            tilhørende aktuelle brukeren
              thumbs/     - inneholder tilhørende bilder
    optiongraphs/ (0755)  - inneholder alle valgstrukturene
                            i XML-filer.
bruker.php              - inneholder informasjon om brukeren,
                          bl.a. om han er logget inn og hvilke
                          rettigheter han har.
DB2xml.php              - program som ut fra parametere sender
                          spørringer til databasen og formaterer
                          disse i XML-format.
GraphEditor.php         - klasse som inneholder visningslaget til
                          graf-redeingeringen.
GraphHandler.php        - klasse som inneholder det tekniske laget
                          av graf-redigeringen
graphs.php              - klasse som inneholder informasjon grafene,
                          brukes bl.a. til å gjøre forskjellige
                          opplistinger av grafer.
index.php               - hovedsiden. viser forskjellig visning ut
                          fra hvordan mottakeren/brukerere samhandler
                          med siden ved å bruke de andre klassene i
                          denne mappen.
init.php                - klasse som gjør databasekommunikasjon
                          mulig, inneholder nødvendig informasjon
                          om databasen.
interactv.swf           - filmavspilleren (Flash-programmet)
```

```
layout.css              - stilskjema
members.php             - Denne filen er en slags index-fil for
                          innloggede brukere.
movieClipPreview.swf    - Flash-fil for visning av filmklipp,
                          og deres thumbnail-bilder
movieclips.php          - klasse som inneholder visingsfunksjoner
                          om filmklipp
site.php                - klasse som inneholder de vanligste
                          visningsfuksjonene
swfobjekt.js            - JavaSript for implementering av
                          flash-filer.
```

Transkoder-maskin

```
/usr/local/bin/
  flvmagic.sh           - skript som transkoder filmer og flytter
                          de til webserver-maskinen
  nisse.sh              - skript som ser etter filer i
                          uploaded-mappen
/var/www/
    movieclips/
        uploaded/   (0774)  - Her legges alle ferdig
                              opplastede filmfiler.
            working/ 0775)  - Her legges alle filmfiler som
                              skal transkodes.
        uploading/  (0775)
            <brukerid>/     - I denne mappen bygges filen som den
                              aktuelle brukeren er i ferd med å
                              laste opp
  init.php              - klasse som gjør databasekommunikasjon
                          mulig, inneholder nødvendig informasjon
                          om databasen.
  LastOpp.class         - Java Applet'en som brukes til å laste
                          opp filer.
  LastOpp.jar           - Tilhører applet'en og kobler sertifikatet
                          med applet'en.
  MittSert.crt          - Sertifikatet
  MultiPartOutPutStream.class - Klasse brukt av
                                opplastnings-applet'en
  ta_imot.php           - Filen som tar imot fil-chunkene fra
                          klienten og bygger de opp til hele filer i
                          uploading-mappen, og deretter legger de i
```

Tillegg I

CD-ROM'ens innhold

Noen av mappene inneholder mapper som heter «gamle greier». Disse inneholder kode vi som vi har forkastet eller videreutviklet.

Filene og mappene som er merket med * er kommentert i vedlegg H på side 96 og kommenteres derfor ikke her.

```
/
    kode/
        bash/
            flvmagic.sh *
            nisse.sh *

        flash/
            interactv.fla - Kildekoden til filmavspillerprogrammet.
            movieClipPreview.fla - Kildenkoden til Flash-filen som
                                   viser et gitt filmklipp.
            NodeBranch.as - Kildekoden til NodeBranch-klassen som
                            brukes i interactv.fla
            OptionGraph.as - Kildekoden til OptionGraph-klassen som
                             brukes i interactv.fla
            OptionNode.as - Kildekoden til OptionNode-klassen som
                            brukes i interactv.fla
        java/
            LastOpp.java - kildekoden til LastOpp.class-filen og
                           LastOpp.har-filen på transcoder-maskinen.
            mittsert.crt *
            MulitPartFormOutPutStrem.java - kildekoden til
                          MulitPartFormOutPutStrem.class-filen og
                          LastOpp.jar-filen på transcoder-maskinen.

        php/
            transcoder/
                init.php *
                ta_imot.php *
            webserver/
                bruker.php *
```

```
            DB2xml.php *
            GraphEditor.php *
            GraphHandler.php *
            graphs.php *
            index.php *
            init.php *
            layout.css *
            members.php *
            movieclips.php *
            site.php *
            swfobjekt *


kopi av nettsiden/      - som beskrevet i forrige kapittel.

tekst/
    doxygendokumentasjon/ - inneholder doxygen-generert
                            dokumentasjon
        html/             - i HTML-format
        latex/            - og i TeX-format

    ekstra/               - inneholder prosjekten fra
                            høstsemesterets prosjekt. må
                            ikke være en del av evalueringen.

    forprosjektrapport/
        kildefiler/
            forprosjektrapport.tex
        forprosjektrapport.pdf

    rapport/
        kildefiler/
         hovedprosjektrapport.tex
        hovedprosjektrapport.pdf - Hovedprosjektrapporten.
```

Tillegg J

Plakat

BACHELORSTUDENT I MEDIETEKNIKK, JON VATNE, PRESENTERER...

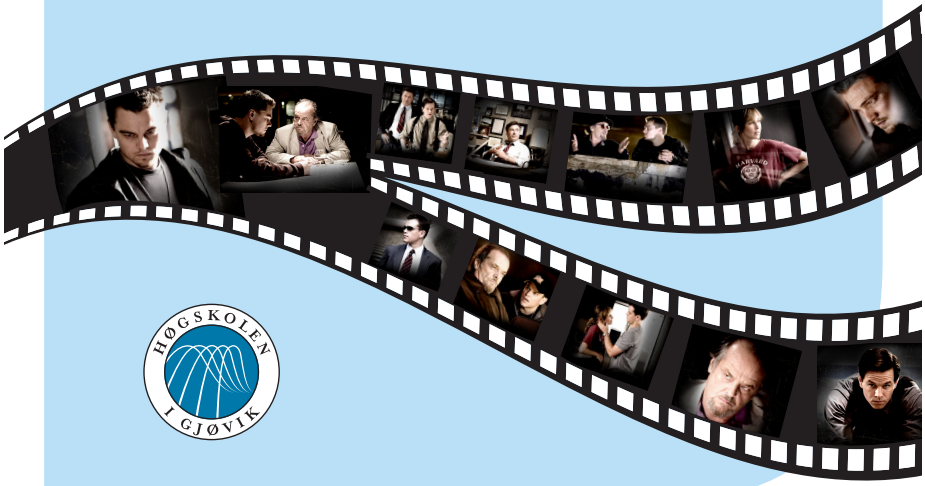
interacTV

HOVEDPROSJEKT '07

STYR HANDLINGEN I FILMEN *MENS* DEN GÅR!

Har du noen gang irritert deg over handlingen i en film? Har du lurt på hva som hadde skjedd hvis...? Med interacTV har du mulighet til å lage film hvor seerne selv kan velge historiens forløp! interacTV er et nettbasert system for redigering og visning av interaktive filmer, hvor mottakeren har mulighet til å foreta valg iløpet av filmvisningen.

Presentasjon av interacTV vil bli holdt torsdag 7. juni klokken 10:00



HØGSKOLEN I GJØVIK