NetGIMP

Bjørn Nyland, Alexander Kvam, Bjørn Reiten, Øystein Huse

23.05.2001

Sammendrag av Hovedprosjekt

Tittel:	netGIMP	NR. :
		Dato :
Deltaker(e):	Alexander Kvam, Bjørn Nyland, Bjørn Reiten og Øystein Huse	
Veileder(e):	Halgeir Leiknes	
Oppdragsgiver:	KpnQWest	
Kontaktperson:	Jon Thingvold	
Stikkord: (4 stk.)	Bildebehandlings verktøy, *Linux*, *GNU*, Åpen Kildekode	
Antall sider:	Antall bilag:	Tilgjengelighet(åpen/konfidensiell): Åpen *GNU* General Public License

netGIMP er internett verktøy for manipulasjon av bilder.
Selve serverdelen av programmet er utviklet for GNU/ Linux plattformen. Klient delen skal derimot kunne kjøres på alle maskiner uavhengig av hvilke operativsystem man kjører. netGIMP er basert på GIMP, som er et bildebehandlingsprogram for Linux/UNIX. I motsetning til Adobe Photoshop, er GIMP et Open source program, hvilket betyr at det er gratis og alle har rett til innsikt i koden. Da Linux ikke er et utbredt operativsystem blant vår målgruppe, vil vi via web å bringe GIMP ut til den vanlige bruker.


netGIMP

Forord

Vi går inn i en tid hvor internett endelig har blitt en del av hjemmet. Bredbånd har såvidt begynt å få fotfeste, og er ikke lenger noe som er forbeholdt bedrifter. Mot slutten av året 2001 er det antatt valgmulighetene innenfor bredbånd vil være flerfoldige og prisene meget overkommelige.

KpnQWest Norway AS, etablert i 1993, var en av de første internett tilbyderne i Norge, og har etterhvert vokst til å bli en av de ledende på markedet. Representert ved Jon Thingvold kom de med et forslag til et hovedprosjekt for avgangsstudenter ved Høyskolen i Gjøvik.

Prosjektet gikk ut på at ved hjelp av et web grensesnitt for *GIMP*, skulle en kunne utføre bildebehandling over internett. Oppdragsgiver var veldig interessert i å vise noen nye sider ved internett, og at dette var gjennomførbart.

Muligheten for å være med på å starte et prosjekt innenfor et område som var såpass uprøvd, gjorde prosjektet veldig interessant og spennende. Med det gode grunnlag for videreutvikling som *open source* arbeid fører med seg, gjorde at Alexander Kvam, Bjørn Nyland, Bjørn Reiten og Øystein Huse, tok denne oppgaven som hovedprosjekt våren 2001.

Vi vil gjerne takke disse personene for støtte og hjelp :

Halgeir Leiknes for en god jobb som veileder.

Jon Tingvold for hjelp med de tekniske spørsmålene som Halgeir ikke kunne svare på.

Ivar Farup for oppstartshjelp til å forstå *scheme*.

Øivind Kolloen for stor hjelp med å få applets & servlets til å fungere slik som de skal.

Gjøvik, 23 mai 2001:

Alexander Kvam Bjørn Reiten

Bjørn Nyland Øystein Huse

Innholdsfortegnelse

1 Innledning/Innhold

1.1 Definisjoner

I rapporten bruker vi mange ord og uttrykk som kan være ukjente for noen, derfor har vi laget en oversikt med forklaring på de utrykkene vi føler kan være uklare.

Ord/Uttrykk	**Definisjon**
AWT	Abstract Window Toolkit, Et klasse bibliotek fra Sun som gir et applikasjons skjelett og *GUI* rutiner for *Java* programmerere.
Corba	Common Object Request Broker Architecture [WebM].
CVS	Current Version System, et system som gjør at flere kan programmere opp mot den samme koden
Emacs	Editor MACroS, en *UNIX* tekst editor utviklet hos MIT som brukes for å skrive programmer [WebM].
Filter	Et *filter* er en serie av handlinger/modifikasjoner som kjøres på bildet.
Fossefallsmodellen	En modell som beskriver gangen i et software prosjekt.
Free software	Gratis programvare, det kan også, hvis man bruker *GNU* sin definisjon som er friheten til å velge.
FSF	*Free software* Foundation. En organisasjon som lager *free software* [WebM].
GNU	*GNU* er et rekursivt akronym for *GNU* is Not *UNIX*. Et prosjekt av *FSF* [WEB].
GIMP	*GNU* Image Manipulation Program [WebA]..
GPL	General Public License, Se Vedlegg A.
GUI	Graphical User Interface. Et grafisk brukergrensesnitt.
HiG	Høgskolen i Gjøvik.

Java	Et programmeringsspråk utviklet for å generere applikasjoner som kan kjøres på alle hardware plattformer, uten noen form for modifikasjoner/porting. *Java* er basert på C++ og er utviklet av Sun [WebM].
JAVA applet	Et *Java* program som lastes inn og kjøres på *web browser*en hos klienten [WebM].
JAVA servlet	En *Java* applikasjon som kjøres på en Web server som gir serverside processing.
JNI	*Java* Native Interface [WebM].
Linux	Et gratis operativsystem som er basert på *UNIX*.
LISP	LISt Prosessing, et programmeringsspråk [WebM].
Open source	Vi benytter definisjonene til Bruce Perens. Det innebærer blant annet at ”*open source*” betyr at alle har tilgang til kildekoden til et program.
Output	Data som sendes fra en modul eller et program. For eksempel så er meldinger til skjermen en *Output*.
Perl	Practical Extraction Report Language, er et programmerings språk skrevet av Larry Wall som kombinerer syntaks fra flere *UNIX* verktøy og språk. *Perl* er designet for å takle system administrator funksjoner og gir muligheten til avansert string behandlings funksjoner. Er også brukt til å Web server programmer [WebM].
PHP	PHP(Personal Home Page) Hypertext Preprocessor, er et script språk som brukes til å lage dynamiske web sider [WebM]..
Scheme	En *LISP* ”dialekt” utviklet hos MIT og Indiana University [WebM].
Swing	Er et *Java* verktøy for utvikling av *GUI*s. Det Inneholder elementer slik som menyer, verktøylinjer og dialogbokser [WebM].
Root	Brukeren på et *UNIX* som har alle rettigheter.
UNIX	Et flerbruker operativsystem.
Web browser	Et program som er din front end mot WWW. For eksempel Netscape, Internet Explorer, Opera mm.

X	X Windows System. Et vindubasert brukergrensesnitt som kan benyttes under *UNIX* og de fleste moderne operativsystemer.

1.2 Definisjon av oppgaven/Problemområde og avgrensning

Oppgaven går ut på å tøye grenser ved web kommunikasjon. Ønsket fra oppdragsgiver var å få implementert et web basert grensesnitt for *GIMP*. *GIMP* er et kraftig bildebehandlingsprogram for *Linux*, og kan sammenlignes med f.eks. Adobe Photoshop.

Webklienten skulle først og fremst være plattform uavhengig, slik at man kunne kjøre den under de fleste operativsystemer som f.eks. Mac, Windows, *Linux*, BeOS, etc.

Det viktigste vi skulle kunne gjøre var at klienten skulle kunne

1. Laste opp et bilde
2. Utføre modifikasjoner på bilde
3. Få et preview av bilde
4. Laste ned et endelig resultat

Koden til programmet skal lisensieres som for *GIMP*, dvs. *open source*. Målet med oppgaven er å prøve å utvide bruksområdene for WWW.

1.3 Målgruppen

Målgruppen for programmet er alle som bruker Internet og som er interessert i å manipulere bilder uten å måtte installere et bildebehandlingsprogram.

1.4 Formålet

Når vi valgte denne oppgaven var det fordi vi syntes den virket spennende og fordi den ga oss muligheten til å :

1. Utvikle et program der vi kunne sette oss dypere inn i *Linux* og *Open source* programmering

2. Bidra til at *GNU*/*Linux*/*GIMP* får et større brukerområdet

1.5 Egen bakgrunn og kompetanse

Av formell utdannelse har Alexander Kvam tre årig allmennfaglig utdanning fra videregående. Bjørn Nyland har tre årig allmennfaglig utdanning fra videregående. Bjørn Reiten har tre årig elektronikk utdanning fra videregående, han har dessuten et ett årig forkurs for ingeniørhøgskolen. Øystein Huse har tre årig allmennfaglig utdanning fra videregående, i tillegg har han gått ett år på programmeringslinja på Bjerkely Folkehøyskole.

Alexander Kvam, Bjørn Reiten og Bjørn Nyland er i sitt siste semester av en tre årig utdannelse til høgskoleingenør datateknikk. Øystein Huse er i sitt siste semester av en to årig utdannelse til høgskolekandidat multimedieteknikk.

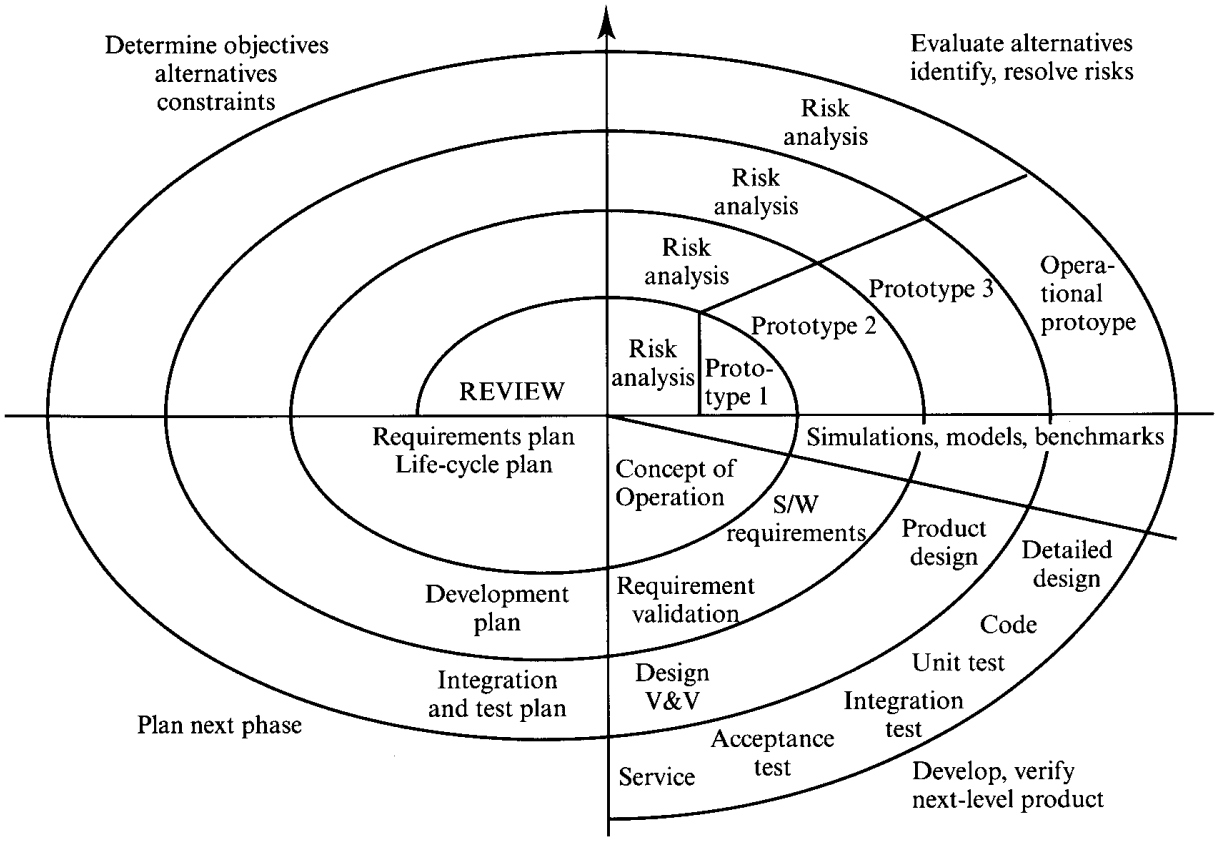
1.6 Arbeidsformer

Vi valgte å ha en prosjektleder fordi vi var en stor gruppe og trengte en person som kunne koordinere samarbeid/møte med oppdragsgiver/veileder. Slik at gruppa fikk en kontaktperson som knutepunkt.

Gruppeleder fikk ansvar for å organisere møter og holde oversikt over hva som skulle skje, og eventuelt mekle hvis konflikter oppsto. Mens gruppemedlemmene fikk ansvar for selv sørge for å overholde tidsfrister og jobbe for fellesskapet.

Gjennom prosjektperioden har vi jobbet individuelt, parvis og som gruppe når dette har vært naturlig. Vi delte opp oppgavene mellom oss. Vi bestemte oss for at Alexander Kvam skulle jobbe med applet og *GUI*, Bjørn Nyland med applet, servlets og kommunikasjon mot C, Bjørn Reiten med koden mot *GIMP*, og kommunikasjon mot servlet mens Øystein Huse skulle jobbe med å kommunikasjon mot *GIMP* og bruk/tilkalling av plugins/funksjoner.

Vi har gjennom hele utviklingsfasen brukt spiralmodellen, denne er illustrert i figur 1 på side. Figuren er hentet fra Ian Sommervilles[Som95] bok om system utvikling.
Foruten å beskrive hvordan vi har utviklet net*GIMP*, er dette også en modell som kan beskrive utviklingen av et *open source* system. Fordelen med å bruke en denne utviklingsmodellen er at brukerne kan ta del i utviklingen, og faktisk kan ta del av videreutviklingen av prosjektet. Da vil programmet være gjenstand for oppdateringer og utvidelser. Feil og mangler oppdages tidlig i utviklingen av programmet og man oppdager ofte nye elementer som bør være med på et tidlig tidspunkt.



Hver loop i spiralen representerer en fase i prosjektet, for vårt vedkommende vi den første fasen tilsvare planleggingsfasen, den andre programmerings fasen osv.

1.7 Organisering av rapporten

Rapporten følger en mal som er gitt av avdelingen for elekro- og allmennfag. Det er gjort enkelte avvik ifra den.

1. Innledning	Her forsøker vi å gi litt bakgrunn for prosjektoppgaven vi har hatt, vi gir dessuten en oversikt over rapportens oppbygning.
2. Kravspesifikasjon	Her beskriver vi hva som skal lages sett utfra brukernes synspunkter.
3. Design	Her skisserer vi hvordan programmet skal lages. Her finner en også de tekniske detaljene.
4. Koding	Dette kapittelet gir en gjennomgang av implementeringen av programmet. Beskrivelse av hvordan kodingen har foregått.
5. Testing	Her beskrives hvordan opplegget rundt testing av net*GIMP* har vært.
6. Installasjon	Her er forklaringen på hvordan man installerer programmet samt en liten innføring til hva man kan gjøre i net*GIMP* og hvordan man kan gjøre det.
7. Konklusjon	I dette kapittelet diskuterer vi resultatene av prosjektet og evaluerer arbeidet vi har gjort.
8. Litteraturliste	Her finnes alle referansene som er brukt i prosjektet
9. Vedlegg	I dette kapittelet finner man en nummerert oversikt over alle vedlegg.

1.8 Terminologi

De ord og utrykk i rapporten som blir brukt i denne rapporten har vi beskrevet i kapittel 1.1 på side.

2 Kravspesifikasjon

2.1 Bakgrunn

GIMP er *Linux* sitt svar på Adobe Photoshop, i motsetning til Photoshop er *GIMP* gratis og *open source* (se 2.2.2). Selv om *Linux* streber for å bli mer brukervennlig kan det fortsatt, for mange, være til dels forvirrende og noe vanskelig å installere og sette seg inn i. Det eksisterer en versjon av *GIMP* for Microsoft Windows, men den lider av ustabilitet. Ettersom internett båndbredde bare blir bedre og bedre er muligheten for å kunne tilby bildebehandling på høyt nivå over internett meget reel.

Kombinasjonen av *GIMP*'s tilgjengelighet og utviklingen på internett, har åpnet for at man kan lage et program som kombinerer dette.

2.2 Grunnleggende krav til systemet

2.2.1 Generelt

netGIMP skal gjøre bildemanipulering lettere tilgjengelig for den vanlige bruker. Programmet skal kjøres på en maskin som benytter GNU/ Linux operativsystemet som plattform. Maskinen må i tillegg ha GIMP av en senere utgave. Klientene som knytter seg opp mot programmet vil være plattform uavhengige, slik at en kan bruke netGIMP fra den plattformen man måtte ønske. netGIMP vil bli utviklet med verktøy som er godt kjent og fungerer med GNU/ Linux.

2.2.2 Open source

Det var et krav fra oppdragsgiver, og noe vi ønsket selv, at prosjektet ble utviklet under prinsippene for *open source*. Det er mange forskjellige tolkninger rundt teamet *open source*, vi vil basere oss på *GNU* sitt lisens system. All kode og dokumentasjon skal lisensieres med *GNU* sin General Public License. "*GNU* General Public License" sier at alle har rett til både innsyn og å kunne forandre på kildekoden, i tillegg til å kunne benytte programmet uten noen form for betaling. Alle programmer som lages utfra kode som er *GPL* lisensiert, må selv være lisensiert med *GNU* sin *GPL*. Denne lisensen er vedlagt rapporten i sin helhet som vedlegg A.

Open source er et omstridt tema, det strider sterke debatter både for og imot. Man kan ta utgangspunkt i markedsledende bedrifter som Microsoft som baserer seg på et lisenssystem, som overhodet ikke ville fungert under samme regler som *GNU* programvare. Mange har et slikt forhold til *open source* at de rett og slett nekter å bruke programvare som ikke faller innenfor denne kategorien. For at dagens marked skal fungere i et *open source* miljø krever enorme omveltninger, og totalt forandring av strategier. I det siste har *open source* fått mer og mer oppmerksomhet, en av grunnene at *Linux* har tatt over store markedsandeler fra Microsoft. Det skal bemerkes at Microsoft har vist interesse for *open source*, og har planer om å lage en *open source* versjon av sitt operativ system Windows.

Vi har brukt ordet *open source* gjennom rapporten, men noen velger å kalle det "*free software*". Det kan lett oppstå mistolkninger rundt dette ordet, og mange vil tenke gratis når man hører det. Det er heller friheten til å kunne velge som ligger bak ordet *free software*.

"*Free software*" is a matter of liberty, not price. To understand the concept, you should think of "free speech", not "free beer".
- Richard Stallman

Vi skal ikke drive noen debatt rundt temaet *open source*, eller teoritisere på meningen bak. Vi vil se på fordeler og ulemper som dette vil medbringe for utviklingen. Det skal legges til grunne at vi er studenter som gjennomfører et hovedprosjekt og dermed ikke tenker profitt på noe av arbeidet, men som en læringsprosess. Erfaring fra *Linux* og en generell forståelse rundt temaet *open source* var med på å gi interesse til prosjektet. Det var et selvfølgelig krav at alt vi gjorde under prosjektperioden skulle være tilgjengelig etter reglene beskrevet i *GPL*. Alle dokumenter og kode var tilgjengelig for den som måtte ønske det.

2.2.3 Sikkerhet

Vi vil prøve, så godt det lar seg gjennomføre, å ikke åpne for noen nye sikkerhetshull. Vi kan likevel ikke stilles til ansvar for hull i eksisterende systemer og protokoller som vi tar i bruk. Det vil bli påpekt eventuelle problemer, og systemene må være godt uttestet.

Da net*GIMP* skal lisensieres i henhold til *GPL*, så garanterer vi heller ikke imot feil som måtte oppstå i programmet. Se vedlegg A for nærmere beskrivelse. Dette er ikke et punkt som gjør at man kan ta lett på saken og se gjennom fingrene på ting som måtte oppstå. Det er viktig at alt fungerer så feilfritt som mulig og dermed gir brukerne en behagelig og opplevelse, som fører til ønsket gjentakelse. Hvis sikkerhet og brukervennlighet blir nedprioritert vil interessen bli likeledes. Populariteten vil ha mye å si for videreutviklingen.

2.3 Omgivelsene

Det er en forutsetning at net*GIMP* serveren kjøres på et *GNU*/*Linux* operativsystem av nyere dato. Programmet er testet på Red Hat distribusjonen av *Linux*, og må minst være på versjons nummer 7.0. Klientene er plattform uavhengige, som vil si at sluttbruker kan ta i bruk net*GIMP* fra hvilket som helst operativsystem den eller de måtte ønske.

2.4 Systemets brukere

Systemets brukere kan hovedsakelig deles inn i to hovedgrupper :

Den ene er en eller flere administratorer. Disse har *root* tilgang på net*GIMP* serveren, og vil ha tilgang til alt som har med drift og vedlikehold av programmet, dette inkluderer også oppgraderinger og å legge til nye funksjoner. Den andre vil bli klientene som via en web side får tilgang til programmet og der kan kommunisere med det.

Ettersom programmet er fritt tilgjengelig, kan enhver som har tilgang til å få satt opp en *GNU*/*Linux* basert web server sette opp net*GIMP*, og i effekt bli administrator. Det kan likevel antas at administrator er en datakyndig person, da det må til en del kunnskap for å få satt systemet i drift. På figur kan man se hvordan systemet fungerer i utgangspunktet. En maskin som fungerer som server inneholder net*GIMP* og andre essensielle programmer. Serveren tar seg av alt kommunikasjon mellom klient og server.

2.5 Brukergrensesnitt

netGIMP vil basere seg på et godt grafisk bruker grensesnitt. Brukergrensesnitt må være enkelt og effektivt, utseende vil minne om andre bildemanipuleringsprogrammer. En person som har prøvd bildebehandlings program før skal raskt kjenne seg igjen. Likeledes vil det for en ny bruker være intuitivt, slik at det er mulig å raskt sett seg inn i og føle seg komfortabel med programmet. Det vil ikke bli laget noe grensesnitt for administratorer da programmet er vedlikeholds fritt. Implementering av nye filtere og lignende må kodes og legges inn for hånd, et brukergrensesnitt for dette hadde vært et prosjekt i seg selv

2.6 Operasjon

2.6.1 Serveren

netGIMP vil bli utviklet og testet på maskiner med Red Hat 7.x. For at netGIMP skal fungere kreves det at noen pakker er installert. Systemet testes med oppgitte versjoner av pakkene, det vil derfor være nødvendig å installere pakker av samme versjon eller av nyere dato. Det kan ikke sies om netGIMP vil fungere i eldre versjoner.

- GIMP versjon 1.2.1
- Webserver - Apache versjon 1.3.19
- Xserver - XFree86-4.0.1-1

netGIMP på server siden vil ikke kreve noe vedlikehold, alt arbeid er automatisert og blir tatt av de nødvendige programmene.

2.6.2 Klienten

Da meningen med *netGIMP* er at det skal kunne kjøres på alle plattformer og gjennom web, så er det eneste kravet at klienten har en nyere *web browser* som vil ha støtte for netGIMP, dette via diverse plug-ins/direkte støtte.

2.6.3 Sikkerhet

Alle kommandoer som skal kjøres vil kjøres mot en temp katalog som er definert i en egen konfigurasjonsfil og som bare administrator har tilgang til å editere. Alle kommandoer kjøres som bruker nobody, og har da heller ikke tilgang/rettigheter til å endre på viktige filer i et UNIX-system.

Sletting av filer fungerer ved at i en konfigurasjonsfil vil det bli definert en path og filene vil dermed slettes derfra og nedover i filstrukturen. Dette gjøres når man finner en katalog som ikke er aksessert på over to timer (denne tiden kan settes i konfigfilen for netGIMP), så vil alle filer og kataloger som finnes i denne katalogen slettes.

Systemkallene som kjøres, generes ut i fra filer som ligger statisk på harddisken som bare Administrator har tilgang til. Disse filene vil bli lagret under en egen katalog (som er satt i konfig filen). En slik fil vil inneholde skjelettet til en kommando i GIMP.

For at brukeren via web ikke skal kunne få tilgang til å se filen, som skal tilkalle GIMP, vil den da legges et sted som ikke vil bli synlig for webserveren. (eks under /tmp/netgimp).
Denne filen vil det settes in path til gimp og til de bildene som det skal jobbes med ut i fra hva som blir sent i netGIMP og hva som står i konfigfilen.

For at ikke brukerene som kopler seg til webserveren ikke skal kunne jobbe på samme fil og eventuelt overskrive andres filer, vil hver enkelt bruker tildeles en ID når de kommer inn på websiden. Denne IDen vil genereres når du kopler deg på websiden og blir tatt vare på hele tiden mens brukeren surfer på websiden i samme browser vindu. Det vil da lages en egen katalog med denne session IDen som navn. Alle filer som brukeren kommer til å generere via netGIMP og laste opp vil da komme under denne katalogen.

2.6.4 Kritisk oppetid

Det er en forutsetning at serveren er oppe for at netGIMP skal kunne fungere. Alt av arbeid blir gjort på server siden. Klienten sitter kun med en GUI som sender data til serveren, går serveren ned, vil klienten sitte med en GUI, men ikke få foretatt noen operasjoner.

2.7 Programmets Livssyklus

net*GIMP* lisensieres under *GNU GPL*. Dette betyr at hvis produktet vårt blir godt mottatt i *open source* miljøet, er det mange som vil være interessert i å gjøre forbedringer og legge til funksjonalitet. Det vil aldri i teorien være et ferdig produkt. Driftsoperatører vil etter all sannsynlighet gå inn i koden å gjøre sine egne tilpasninger.

Produkter som ikke er *open source* har som oftest en gratis test versjon, men med begrenset funksjonalitet. Dette kan gi et vrang bilde av produktet, da brukeren aldri får utprøvd det skikkelig før innkjøp. Her vil ethvert *open source* prosjekt få et fortrinn ved testing og implementering av nye features. Vårt program er helt gratis uten noen heftelser, hvilket gjør det mulig for mange flere å hente det ned og prøve det. Produktet vil da få flere tilbakemeldinger om eventuelle feil og ønskede features. I og med at kildekoden er fritt tilgjengelig er at det ikke bare opp til eierne av programmet til å finne og utbedre feil, alle som ønsker det kan være med på å bidra. Helhetlig sett skal alt dette skal dette føre til et bedre og mer fullverdig produkt som tilfredsstiller brukernes behov.

Det er en rekke punkter som er viktige for at et *open source* prosjekt lykkes. Her er basispunktene i nøkkelord format:

1. Behov
2. Orden
3. Tilgjengelighet

Første punkt kan sies å være dekket, da dette tross alt var en etterspørsel fra en oppdragsgiver. For det andre finnes det ingenting der ute som kan sammenlignes med net*GIMP* blant *free software* eller vanlig software for den saks skyld, og interessen for hva som er mulig på internett er alltid stor.

Det andre punktet, orden, tilsier at koden må være strukturert og dokumentert på en oversiktlig og forståelig måte. Hvis koden er lett å sette seg inn i, vil det være lettere for folk å bidra. Det er også viktig at minst mulig *bugs* følger med produktet som blir gitt ut, en mindre "humpete tur" gir en bedre opplevelse og følelsen av at det ikke er noe halvhjertet prosjekt. Under dette punktet går også installasjon. Den bør være forholdsvis enkel. Ved å pakke programmet inn i *tar* baller, et pakke system for *GNU*/*Linux*, vil alt legge seg på riktig plass når en bruker vil installere det.

Punkt tre har vi et enormt hjelpemiddel til å dekke oss; internett. Det finnes sider som er spesielt dedikert til å melde fra om nye programmer innenfor *open source* og *GNU*/*Linux* miljøet. Vi ser blant annet for oss å få lagt ut programmet på *GNU* sine software sider og freshmeat.net.

Ved noen tilfeller ville det vært et fjerde punkt, portabilitet. Vi har jo dekket dette ved sluttbrukers side, men siden net*GIMP* baserer seg så kraftig på *GIMP* så vil selve programmet net*GIMP* kun være portabelt innenfor *Linux* systemet.

Dersom alle disse punktene oppfylles så er net*GIMP* godt på vei.

2.8 Ytelse

net*GIMP* stiller ikke store krav til klienten, det eneste forutsetningene en har tilgang på intranett/internett og en datamaskin med *web browser* installert.

På serversiden er det selvfølgelig kjekt å ha en kraftig prosessor og stor båndbredde, men det er ikke noen forutsetning for å kunne kjøre net*GIMP* serveren. Minstekravet til net*GIMP* serveren er en i80368 prosessor med 64 MB RAM , men vi anbefaler at den iallefall kjøres på en 500MHz med 256 MB RAM.

2.9 Analyse

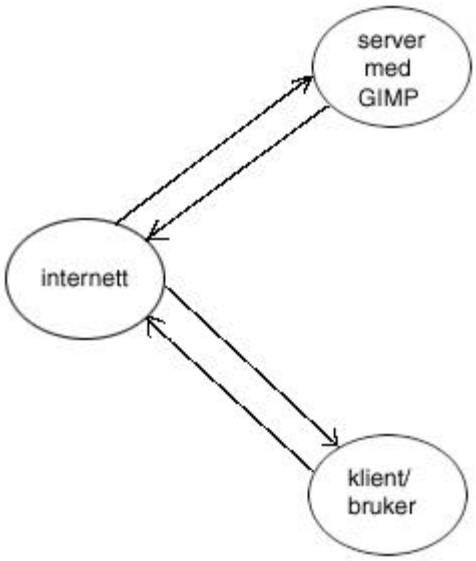
2.9.1 eksisterende systemer

Det nærmeste en kommer et netGIMP lignende program er tjenesten man finner på www.cooltext.com. Denne siden bruker i likhet med netGIMP GIMP til å utføre sine operasjoner. Man har muligheten til å velge blant en rekke predefinerte *styles* og andre *templates*, disse blir så lagt på en tekst med bruker valgte parametre. Det er også mulighet for å lage knapper med tekst på. Det er et veldig fint tilbud for å lage en enkel tekst med noen intersessante effekter, men kommer litt kort i det lange løp.

Problemet med cooltext er at den gir for lite valgmuligheter, du kan for eksempel ikke manipulere et eget valgt bilde frem og tilbake. Cooltext er heller ikke en open source tjeneste, så det er ingen tilgang på kildekoden. En interessant løsning, men desverre inget hjelpemiddel for oss, da vi ikke har noen tilgang på hverken kildekode eller fremgangsmåte.

www.cooltext.com er ikke i nærheten med hva visjonen om hva netGIMP skal bli. Med grunnlag i dette og den manglende sammenlignings materialet kan det sies at netGIMP er et lite pioneer prosjekt

2.9.2 funksjonell spesifikasjon



Den funksjonelle strukturen kan sees på modellen. Dette er et *dataflytdiagram* nivå 0, eller dfd0. Dataflytdiagramer er til stor hjelp for å få oversikten over komponentene og se sammenhengen de imellom. Det er ingen navnsetting på dataflyten på dfd0 da det forsåvidt bare foregår kommunikasjon på et eller annet nivå mellom komponentene, og en videre spesifikasjon ikke egner seg på dette tidspunkt. Prosjektet baserer seg som sagt på spiral modellen, og man ser først helheten etter en del testing og løsningsforslag.

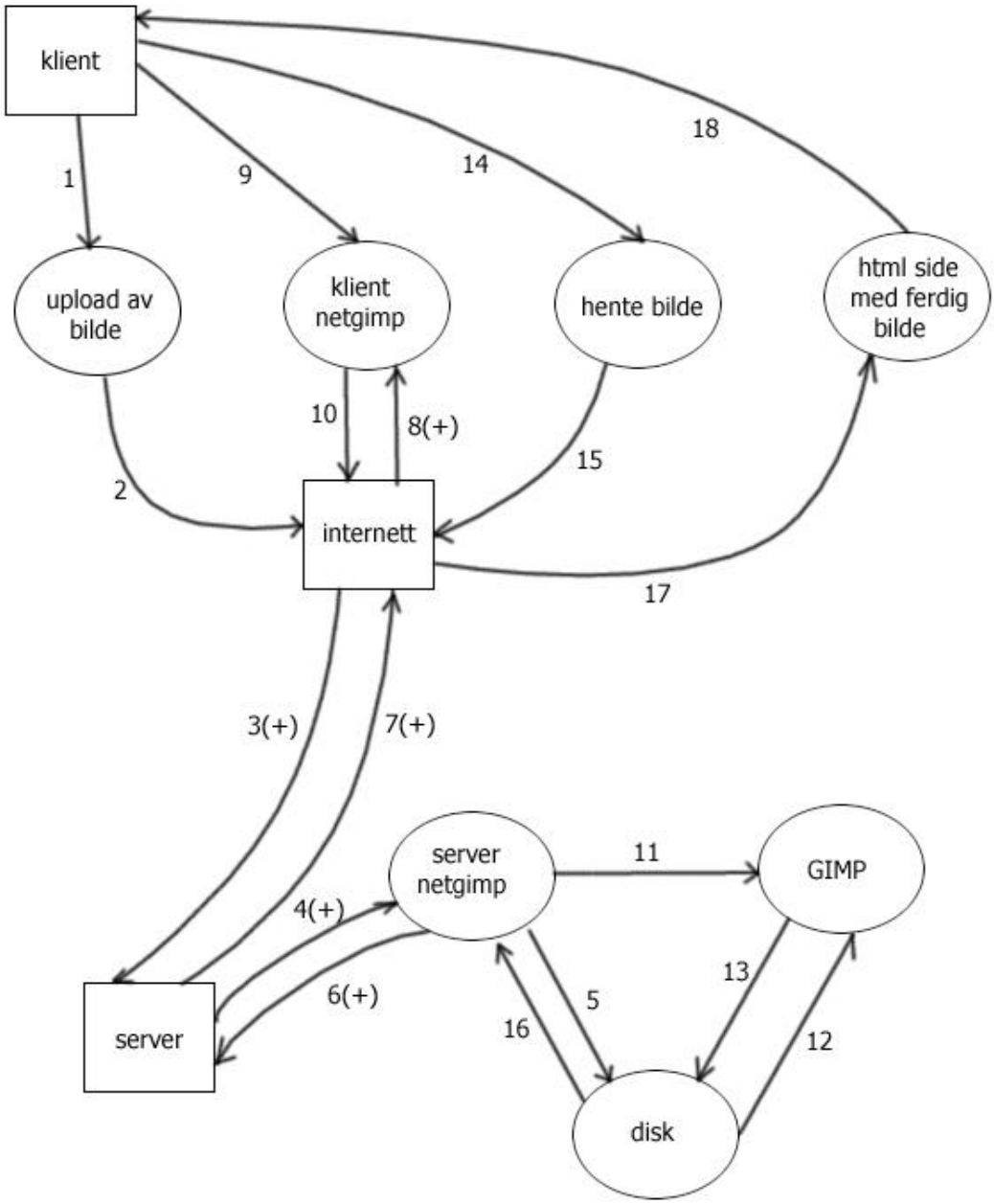
dfd0 forklaring:

-server med gimp
Her befinner selve programmet seg. Det er her alle bilde operasjoner blir utført, brukeren ser aldri noe til det som skjer. GIMP, som ligger på serveren, mottar funksjonskall fra netGIMP og utførerer ønsket operasjon på bildet. Resultatet blir så sendt til bruker.

- internett
Internett blir det store mediumet for kommunikasjonen mellom bruker og program. netGIMP blir selvfølgelig avhengig av brukers internett tilgang og kvaliteten på denne.

- klient/bruker
Ved hjelp av en web-browser vil brukeren kunne aksessere en netGIMP side på internett og sende et bilde det er ønsket å manipulere. Deretter få opp klient-netGIMP med bildet og utføre diverse filtere og lignende, som for eksempel blur eller dynamisk tekst. Når bruker er fornøyd kan bildet hentes ned.

Neste nivå vil gå mer inn i detalj på strukturen og gi et klarere bilde av de forskjellige komponentene. Illustrert ved et *dataflytdiagram* nivå 1, dfd1. Da vi hadde kommet så langt at vi kunne sette opp et dfd1 hadde vi allerede gått igjennom en del prototyper og hadde en mye klarere oversikt over dataflyten.



Dataflyt forklaring

1. klient går inn på upload side.
2. klient har valgt bilde og sender det til server (*).
3.1. bilde blir tatt imot av server.
4.1. server-netGIMP tar imot bilde.
5.1. bilde lagres på disk.
6.1. server-netGIMP sender data til klient, til server.
7.1. server sender videre til klient(*).
8.1. klient-netGIMP startes hos klient med bilde.
9. klient foretar bildemanipulasjon.
10. forandringer sendes til server(*).
  3.2(+). server tar imot.
  4.2(+). server sender til server-netGIMP.
11. server-netGIMP ber GIMP om å utføre manipulasjon
12. GIMP henter bilde fra disk.
13. etter at manipulasjon er ferdig lagres bilde på disk.
  6.2(+). server-netGIMP sender bilde til server
  7.2(+). server sender bilde til klient-netGIMP(*)
  8.2(+). klient-netGIMP tar imot bilde. (#)
14. førespørsel om å hente ferdig bilde.
15. førespørsel om å hente ferdig bilde(*).
  3.3(+). server tar imot.
  4.3(+). forespørsel sendes til server-netGIMP.
16. server-netGIMP hetner bilde fra disk
  6.3(+). server-netGIMP tar og sender ferdig bilde til server.
  7.3(+). server sender ferdig bilde til en html side(*).
17. ferdig bilde lagt på hmtl side.
18. klient lagrer bilde på lokal disk

* via internett.
# denne prosessen vil bli kjørt i en løkke så lenge klient ønsker å foreta manipulasjon på et bilde.

Klient

U**pload av bilde**

Ved å gå inn på en side på internet som tilbyr netGIMP tjenesten vil denne siden komme opp. Det vises et felt hvor brukeren kan skrive direkte inn *path* eller bla gjennom et lokalt lagringsmedium etter et bilde. Når brukeren er ferdig og trykker på ok, vil han få opp neste side.

klient-netGIMP

Dette vil være et grafisk brukergrensesnitt som vil minne noe om et bildebehandlingsprogram (se GIMP, Adobe Photoshop). Her vil klienten kunne utføre diverse operasjoner på bilde ved hjelp av et intuitivt navigerings system. Klient-netGIMP tar imot bilde fra server-netGIMP.

hente bilde

Når klienten er føler seg ferdig med bildet kan han sende en forespørsel om å få det ferdige bildet.

html side med ferdig bilde

netGIMP vil generere en html side med det ferdige bildet på som det kan høyreklikkes på for å lagres.

Internett(/intranett)

Medium for all kommunikasjon mellom klient og server.

Server

Server-netGIMP

Programmet som vil ta imot alle operasjoner fra klienten og gjennomføre disse. Bilder blir lagret og hentet fra disk. De nødvendige parametre for å starte opp klient-netGIMP lir sendt. Her vil alle filtere og andre operasjoner som skal gjøres av GIMP bli tilpasset og sendt til GIMP for manipulasjon. Server-netGIMP vil vite når GIMP er ferdig med bildet og sende det til klient-netGIMP

Disk

Lagrings medium på serveren hvor bilder blir lagret og hentet.

GIMP

Dette er GNU/ Linux sitt bildemanipuleringsprogram. GIMP vil bli brukt for å kunne utføre bildemanipulasjoner. GIMP vil få beskjed om hva den skal gjøre og hvilket bilde den skal operere på fra server-netGIMP. GIMP vil så hente bilde fra disk, utføre operasjon og lagre det igjen.

3 Design

3.1 Design perioden

Etter at kravspesifikasjonen var ferdig var det fortsatt en del som var uklart. Vi hadde ingen tidligere prosjekter som kunne være retningsledende for oss, og det var diverse problemer med å spore opp metoder og dokumentasjon på hvordan vi skulle bruke GIMP. Disse problemene fulgte oss i lang tid utover design perioden.

3.2 Eksisterende Systemer

Som det ble gjort klart tidligere i kapittel 2.9.1 finnes det ingen lignende systemer ved dags dato. Det finnes et par eksempler på bildemanipulasjon på internett, men disse er statiske med dårlige valgmuligheter, og liten mulighet for videreutvikling.

Det ble tatt en titt på www.cooltext.com sine sider., Desverre var ikke mye i hverken inspirasjon eller kunnskap å trekke fra sidene. "Cooltext" er for det første veldig statisk, og for det andre heller ikke er lisensiert under noen form for open source, hvilket gjør at vi ikke kan trekke noen lærdom fra hvordan det er bygd opp.

3.3 Valg av metoder

3.3.1 Bruker interface

Brukeren skal ha mulighet til stor interaksjon, hvilket krever fleksibilitet. Vi lekte med tanken på å bruke *Flash*, men det ble fort klart at Java var den eneste gjennomførbare løsningen. Da vi startet med prosjektet var Java det beste web-baserte programmerings språket til vårt bruk. Java er også plattform uavhengig, som var et viktig krav ved utviklingen Vi trengte et språk som kunne tilby en brukbar GUI. Vi startet med AWT, som er Java's første GUI implentasjon. Vi gjorde et raskt utkast til GUI, men under forskningen oppdaget vi Java's nye og mye bedre GUI implementasjon, Swing.

Swing er arvtakeren til AWT. Desverre er det ennå dårlig med støtte for Swing i dagens web browsere. Neste generasjon web browsere skal støtte Swing, Netscape er en av de få som allerede gjør det. Til tross for svak støtte, måte Swing bli valget. AWT er ikke bare stygt, men mangler en rekke funksjoner som Swing har. Et viktig punkt var videreutvikling, og da ville ikke AWT vært et særlig framtidsrettet valg, hvilket gjorde Swing til en logisk løsning.

Da vi hadde kommet frem til at en Java *applet* var den mest fornuftige løsningen måtte vi også ha en *servlet* del som kunne kommunisere med *applet'en.* Servleten(e) vil fungere som ryggraden i systemet, med appleten som den visuelle delen som brukeren jobber mot

3.3.2 Kommunikasjon mot GIMP

Det første vi måtte finne ut var hvordan vi skulle kommunisere med *GIMP*. Vi gikk grundig til verks for å finne de forskjellige metodene som kunne brukes, og kom opp med en rekke forskjellige løsninger. Disse var :

1. Bruk av scripting

1. Bash og *GUI*le *Scheme* script. Dette vil føre til kommandobasert kjøring av GIMP.

2. Bruk av en kjerne i C opp mot:

    1. *JNI - Java* Native Interface
    2. *CORBA*
    3. Sockets

Vi måtte sette oss inn i hvordan de forskjellige metodene fungerte, og vurdere de i forhold til brukbarhet, hvor raskt det var mulig å sette seg tilstrekkelig inn i de, og tilgjengelig tid. Det skal nevnes at vi ikke hadde noen tidligere kunnskap innenfor dette området.

1.
En metode for å få *GIMP* til å utføre en operasjon er å bruke den innebygde støtten for script ekstensjoner. Ekstensjoner kommuniserer med *GIMP* på samme måte som plug-ins

    1.1
    Guile Scheme, som er en dialekt av *LISP*. Dette er et språk bygd opp av paranteser. For hver ny blokk vil man sette i en ny parantes. Mot GIMP kan man her tilkalle GIMP sine funksjoner for å utføre oppgaver. Problemet her vil være at en ikke kan jobbe direkte mot pixler.

2
En annen metode for å kommunisere med *GIMP*, er direkte med C. Man blir da nødt til å definere en API som kommuniserer mot GIMP.

2.1

JNI er en del av *Java*'s JDK. Ved å skrive programmer som bruker *JNI* blir koden plattform uavhengig. For å bruke JNI må man lage funksjoner på både C siden og java mot JNI, slik at Java og C kan kommunisere med hverandre.

2.2

CORBA gir deg programmeringsspråk uavhengighet ved å bruke Interface Description Language. IDL lar *CORBA* objekter bli tolket på samme måte. Det eneste som trenges er en "bro" mellom språkene, i vårt tilfelle *java* & C, og IDL. *CORBA* objekter kommuniserer med hverandre ved å bruke en Object Request Broker (ORB) som mellomledd. *CORBA* har en rekke fordelaktige attributter deriblant:

- skalerbar for å håndtere et stort antall brukere
- støtte over en rekke plattformer
- bra sikkerhet
- bred støtte i markedet

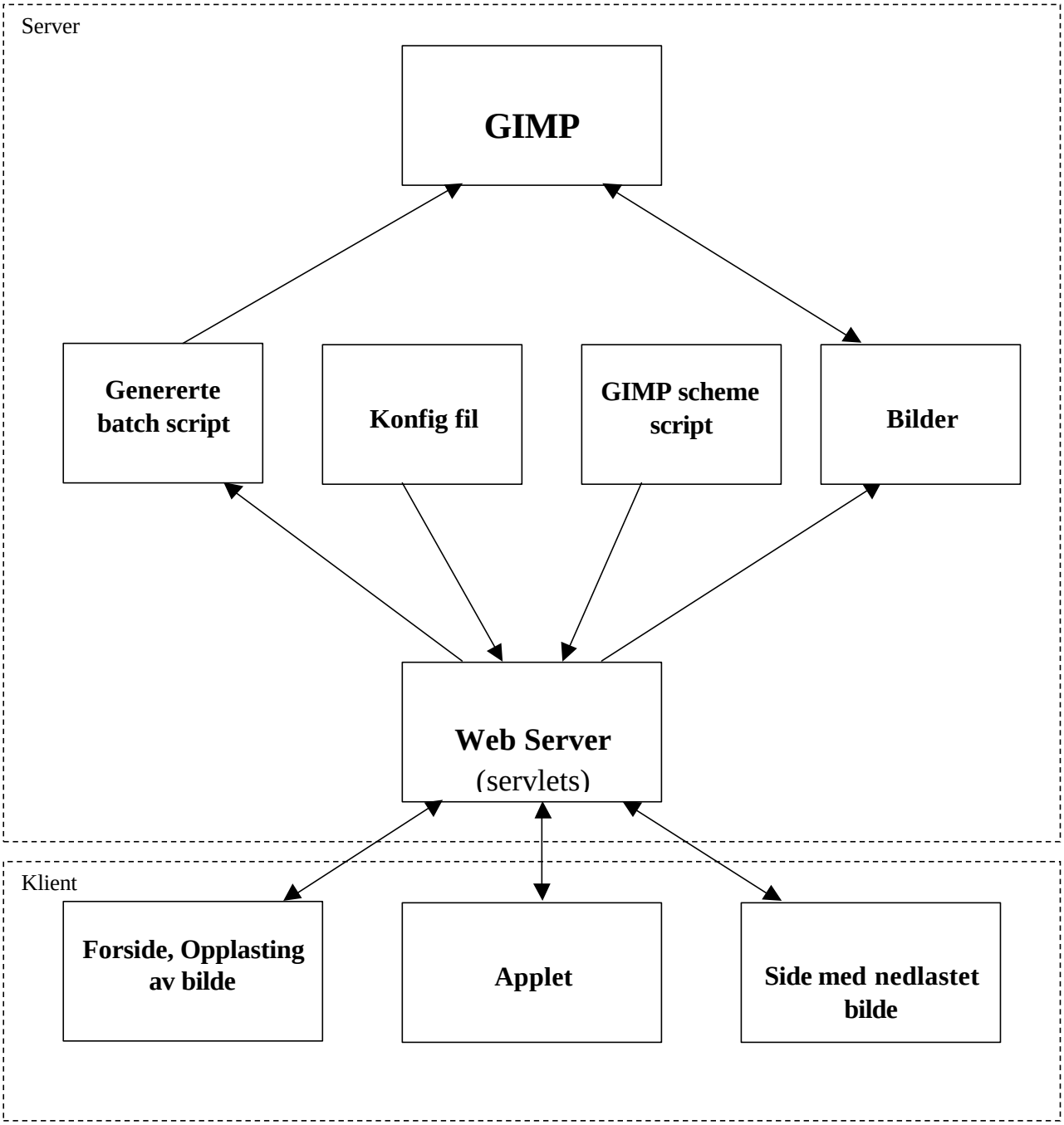
Ulempen var at vi hadde knapt hørt om *CORBA* før og måtte sette oss inn i det på relativt kort tid.

2.3

Sockets: her vil man kommunisere mot en socket for å få kommunikasjon mellom C og Java. Java vil da skrive til en socket, for å sende info til C, og C vil lytte på en port (socket) for å motta meldinger. Problemet vil her å få C til å kommunisere med GIMP.

Det hadde vært ønskelig å kunne valgt en av løsningene som brukte en C-kjerne da dette ville ha gitt produktet en mye større fleksibilitet, og bredere apell til videre utvikling. Likevel, etter en lengre prosess hvor vi mer eller mindre hadde testet de forskjellige metodene, ble vi klar over at vi hverken hadde tid til å lage en C kjerne som kommuniserte opp mot GIMP og i tillegg sette oss tilstrekkelig inn i de metodene som denne løsningen krevde. For å få gjennomført denne løsningen måtte vi ha satt oss grundig inn i kildekoden til GIMP, noe som var et altfor omfattende prosjekt for oss på det tidspunktet. Vi måtte dermed se oss nødt til å velge den første løsningen, scripting. Ulempen med scripting er som sagt at man ikke kan jobbe direkte på pixler, dette er likevel et kraftig verktøy, og ville gi oss den beste muligheten til å rekke målene vi hadde satt oss.

3.4 Grov design



Dette er hovedgangen i hvordan webserveren som kjører netGIMP vil fungere. Klienten starter med å laste opp et bilde fra en startside med en uploadform. På serveren vil dette lagres som et tif bilde, for å ikke miste kvalitet på bildet underveis på grunn av komprimering. Så sendes en applet tilbake til klienten, som vil være en fast side resten av tiden.

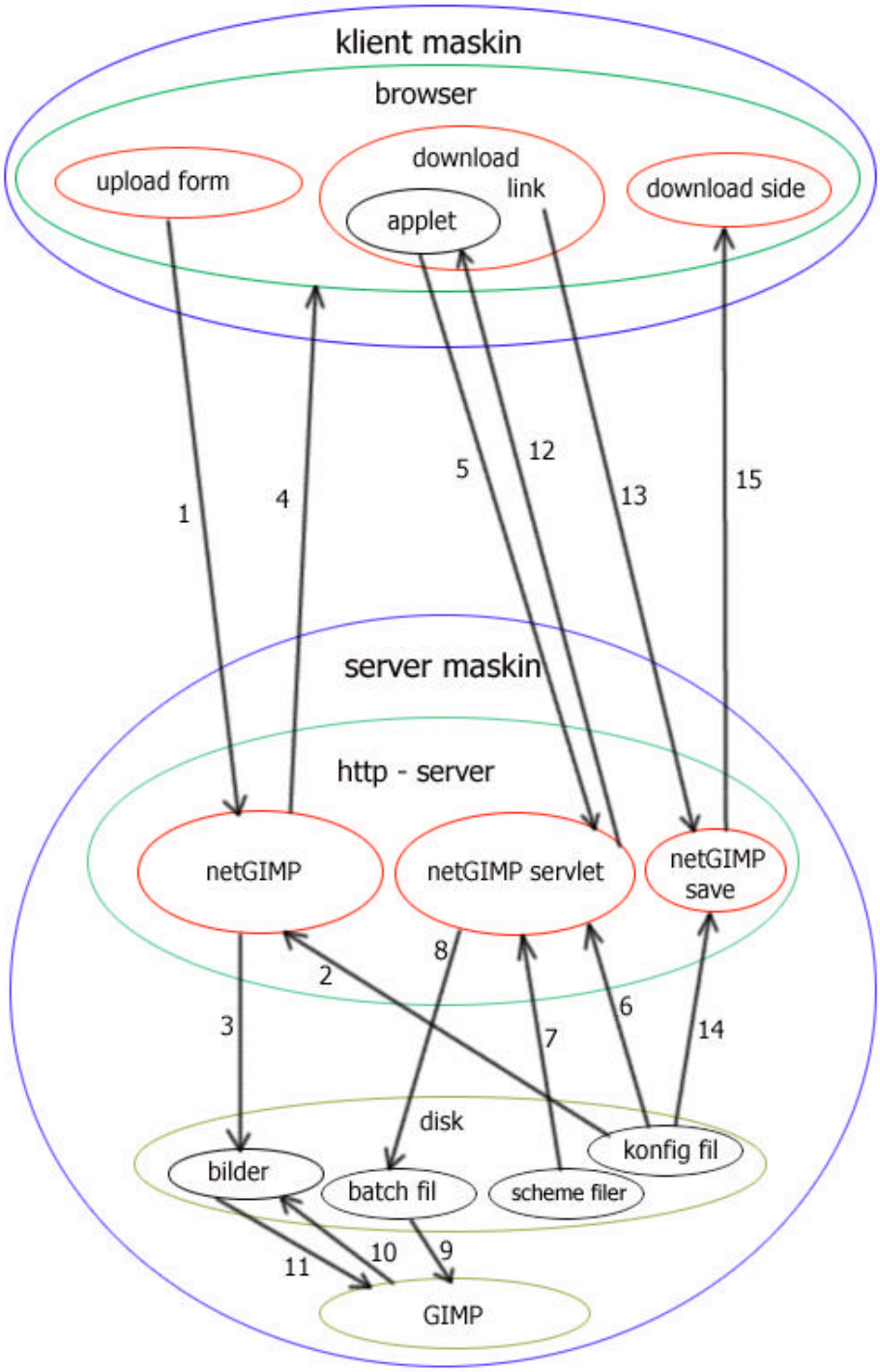
I denne appleten vil brukeren kunne se det bildet som er lastet opp. Her kan man da kjøre forskjellige filtere på det bildet som er lastet opp via GIMP. Appleten vil da sende forskjellig informasjon til webserveren, ut i fra hva klienten har valg av filtere. Hvert filter vil kjøres på det generte tif bildet, og samtidig vil det lagres som jpeg, dermed vil man alltid kunne beholde kvaliteten på bildet.

Den informasjon som blir tatt i mot styres av Java servlets. Servleten vil hente informasjon som er lagret i en egen konfigurasjons fil (info om hvor programmer og filer ligger på harddisken). Så vil den kunne lese ferdigdefinerte scheme script, og ut i fra hva som er sendt av parametere fra appleten, generere et batchscript. Dette batch scriptet vil så eksekvere GIMP som i sin tur vil generere en ny fil med resultatet.

Når brukeren er fornøyd med det bildet som er redigert, kan han velge en egen download link, som vil føre til at et nytt webbrowser vindu vil komme opp med det ønskede bildet. Et nytt vindu er nødvendig fordi brukeren skal slippe å laste ned applet på nytt. Hvis bildet hadde kommet på samme side, så ville appleten henge seg, blant annet fordi den ikke får de riktige parametrene inn (som ble sendt fra forsiden).

3.5 Detaljert design/Komponentene

3.5.1 Detaljert diagram



Forklaring på diagram:

1. Brukeren uploader et bilde.
2. Leser fra konfigfil.
3. Lagrer bilde på disk
4. Servlet sender en HTML kode til klienten med applet-tag. Appleten startes i browseren og laster inn bildet (via HTTP serveren) som ble lagret på punkt 3
5. Når brukeren kjører et filter, sendes en request med de aktuelle parametre til netgimp servleten.
6. Leser fra konfigfil.
7. Henter aktuell scheme fil.
8. netGIMP servleten genererer en batch-fil og kjører bash via et systemkall.
9. Bash kjører batch-filen som starter GIMP.
10. GIMP åpner et bilde og utfører filteret.
11. GIMP lagrer bildet.
12. Appleten laster ned det nye bildet og viser det.
13. Når brukeren vil lagre bildet og trykker på downloadlinken så sendes det en request save servleten.
14. Save servlet leser fra konfig fil
15. Save servlet sender html side med bilde tilbake til bruker

Punkt 5-12 gjentas flere ganger helt til brukeren velger å avslutte.

3.5.2 Servlet - netgimp.java

Netgimp.java har to metoder. En doGet og en doPost metode. Når man skriver URLen til servleten i browseren blir netgimp.java kjørt og da vil doGet metoden i den kjøres. Denne viser en form der brukeren kan velge et bilde og laste det opp.

Når brukeren trykker OK, blir det sendt en POST forespørsel til netgimp.java. Denne kjøres og doPost-metoden kjøres. I doPost lagrer bildet på disk og skriver ut en ny HTML side med applet tag, slik at browseren starter appleten.

Dette skjer i doGet:

Les config fil fra disk.
Sett globale variable fra config.
Skriver HTML side som inneholder upload form.

Dette skjer i doPost:

Les config fil fra disk.
Sett globale variable fra config
Henter parametre fra upload form.
Sjekker om opplastet fil er større enn en viss grense, i såfall så vises ei side med feilmelding og servleten avsluttes.

Finner fil ekstensjon til opplastet fil og sjekker om det støttes av netGIMP. Hvis filen ikke støttes, så vises ei side med feilmelding og servleten avsluttes.
Lager midlertidige kataloger for lagring av bilde og batch-fil.
Hvis alt har gått bra, så sendes ei side med applet-tag. I applet-tagen sendes det tre parametre til appleten:

Session ID: Et unikt 10 bytes ID som brukes til å skille mellom brukere. Katalognavnet til brukerens midlertidige kataloger er session ID.
Filnavn: Navnet på opplastet fil.
URL til servlet: Adressen til servleten som appleten skal sende bl.a. batch-kommandoer til.

Det er nødvendig for servleten å sende disse parametrene til appleten fordi en applet ikke håndterer en HTTP session riktig. Hver gang appleten skal sende en beskjed til servleten, så må den i tillegg til aktuell beskjed sende med session ID og eventuelt filnavn. Hvis HTTP session hadde fungert, så kunne session ID og filnavn blitt lagret som session variable hos serveren.

Utenfor applet-tagen blir det laget en link til netgimpSave.java og inneholder session ID som parameter til netgimpSave.java.

3.5.3 Servlet - netgimpSave.java

NetgimpSave er en servlet som skal vise en HTML side med link til sist modifiserte bilde slik at brukeren kan lagre det. Man må ha en slik servlet fordi en applet ikke har lov til å skrive til disk. Dette betyr med andre ord at vi ikke kan bygge inn lagre-funksjon direkte i applet'en.

For å komme rundt problemet, må vi legge en link utenfor applet-tag'en som peker på netgimpSave og som sender med session ID som parameter.
Dette er det den gjør:

Les config fil fra disk.
Sett globale variable fra config fil.
Finner sist modifiserte fil i temp-katalogen.
Skriver ut HTML kode med URL til bildet.

Man kan når som helst lagre bildet, selv mens appleten jobber. Dette er fordi den er utenfor appleten. Preview bilder vil ikke komme opp her, fordi de lagres under en egen preview-katalog. Hvis man tar undo/redo, så vil appleten sende en touch-beskjed til netgimpServlet.java slik at ny dato til aktuell fil blir satt, slik at riktig bilde kommer opp.

3.5.4 Servlet - netgimpServlet.java

Filen netgimpServlet er en servlet som vil gjøre alle systemkall for å kjøre gimp og kommunisere med appleten. Denne servleten har bare en doPost metode. Denne vil bli tilkalt ved en forespørsel fra appleten.
Dette er hovedoppbygningen Servleten:

1. Leser en konfigfil for å laste inn globale faste variable. Eksempel på dette kan være en variabel med path til hvor eksekveringsfilen til GIMP ligger lagret.

2. Åpner en forbindelse for å kunne motta meldinger fra appleten.

3. sjekker hva slags type som sendes fra applet og tilkaller riktig funksjon ut i fra dette.

4. Sender informasjon tilbake til applet.

For å få system på hva slags meldinger som er sendt fra appleten, bruker vi en Hashtable. Denne fungerer slik at man har en array med elementer, hvor hvert element har en tekststreng som index.

For hvert kall fra appleten vil det alltid sendes med en index som kalles ”type”. Denne bestemmer hva servleten skal gjøre videre.
index ”type” kan ha inneholde følgende strenger:

”batch” - Det vil nå komme et filter som skal kjøres. Her vil man også sjekke om det finnes temp kataloger som er for gamle. Disse vil da automatisk bli slettet.

”init” - Variabler som ligger i konfigfilen skal leses og sendes tilbake til appleten.

”touch” - Appleten vil sende med path til en fil, som man skal endre attributtene til filens sist modifiserte tid på. Denne vil endres til nåværende tidspunkt.

”rename” - Appleten vil sende med path til en fil som skal ha ett nytt navn. Her vil servleten gå inn og forandre navnet på denne fila, til ønsket filnavn, som også er sendt med.

En annen index som også vil bli send med hver gang er ”session_id”. Denne inneholder navnet på underkatalogen som brukeren vil legge filer under. Katalogstrukturen vil derfor bli slik:

/Tempkatalog / session_id / brukerens tempfiler

Hvorav ”session_id” vil være forskjellig ut i fra hvilken bruker som sender en forespørsel.

Vi vil nå gå nærmere inn på hver enkelt hoveddel.

Ved type lik ”batch” vil en funksjon filter bli tilkalt. Denne vil returnere en Hashtable med info om system kallet for å kjøre GIMP var vellykket eller ikke.

Først vil en hente en index lik ”filter” fra Hashtabelen som beskriver hvilken fil som skal lese inn. Dette vil være et ferdiglaget gimpscript som vil kjøre et spesielt filter på et bilde.

Den aktuelle filen vil bli lest inn i en string. Så vil denne stringen bli splittet på spesialtegnet i filen som er lik @. Først vil man finne hvor mange felter som skal settes inn, og så vil man sette inn så mange verdier som man fant forekomster av @ i strengen. Parameterene som skal settes inn i strengen vil hentes fra index lik ”0” og andre parameter vil være index lik ”1” osv. Det vil da bli sendt med riktig antall parametere fra appleten ut i fra hvilket filter som skal kjøres. Dette vil si at man for hver @ i strengen vil bytte ut denne med en variabel sendt fra appleten.

Den strengen som nå er bygd opp vil nå skrives til en fil, som lagres under en egen temp katalog. Denne katalogen er også bygd opp med session_id som siste del av katalogstruktur. Scriptet vil da laste et bilde som ligger under denne katalogen og generere en ny fil som legges i samme katalog med et annet navn.

Grunnen til at scriptet må skrives til fil, er at java sitt exec kall bare takler en parameter. Siden GIMP krever mange parametere, så må denne stringen først skrives til fil, og så eksekveres dette scriptet som ett shellscript. Path til filen som skal kjøre dette scriptet sendes til funksjonen runBatch(). Her vil da shellscriptet eksekveres. Hvis en feil oppstår under eksekveringen, vil det komme en melding ”*batch command: experienced an execution error*.”. Hvis første linje inneholder dette, vil denne prosessen bli slettet, og servleten vil returnere en feilmelding til appleten.

Når man er ferdig med å eksekvere scriptet, vil man så gå inn og sjekke i de katalogene som inneholder tempfiler. Funksjonen *deleteOldDirectories* vil da tilkalles med temp katalog som parameter. Denne vil da sjekke alle kataloger under denne for å finne noen som er for gamle. Disse katalogene vil være session_id kataloger. Hvis en fil under denne katalogen, blir forandret, vil også sist modifisering på denne katalogen være lik.

/Tempkatalog / session_id / brukerens tempfiler

Tiden før en katalog skal slettes er definert i konfigfilen. Denne er satt til default to timer. Hvis en finner en katalog som er eldre en denne tiden vil funksjonen *deleteDir* tilkalles med aktuelt katalognavn med full path som parameter. Her vil da alle filer og kataloger slettes rekursivt. Denne vil bli tilkalt for temp kataloger for både bilder og temp katalog for script.

Ved type lik ”init” vil konfig filen bli lest, legges inn i parametere og sendes til appleten i en Hashtable. Denne starter med å tilkalle funksjonen *readConfig*. Denne Starter så med å åpne konfigurasjonsfilen, og leser linje for linje. Hvis linjen starter med et kommentar tegn vil linjen hoppes over. Hvis linjen starter slik vil det tolkes som en kommetarlinje:

;
#
//

Resten av linjene vil leses inn i en streng, som splittes på likhetstegn (=). Den første delen av linja blir da satt til key (index) og den siste delen blir satt til value (parameter verdi) inn i en Hashtable. Når filen er lest ferdig vil denne Hashtablen returneres til appleten.

Ved type lik ”touch” vil man få en filpath som parameter, hvor man skal endre attributtene til filens sist modifiserte tid på. Denne vil endres til nåværende tidspunkt. Den starter da ved at man mottar et filnavn fra appleten, hvor man legger til katalognavn til denne filen. Så tilkalles funksjonen *touchFile*, som henter tiden akkurat ved det tidspunktet. Hvis filen finnes vil den prøve å sette sist modifiserte tid på denne filen til dette tidspunktet. Returnerer true hvis dette går bra, og man sender så en respons tilbake til appleten.

Ved type lik ”rename” vil sevleten rename en fil, som er gitt ved parameter. Funksjonen vil sjekke om den filen som skal få et nytt navn finnes. Så vil den gi denne filen et nytt ønsket navn. Returnerer true hvis alt gikk bra og false ellers.
Til slutt vil det returneres en melding til appleten om det gikk bra eller ikke, sendt i en Hashtable

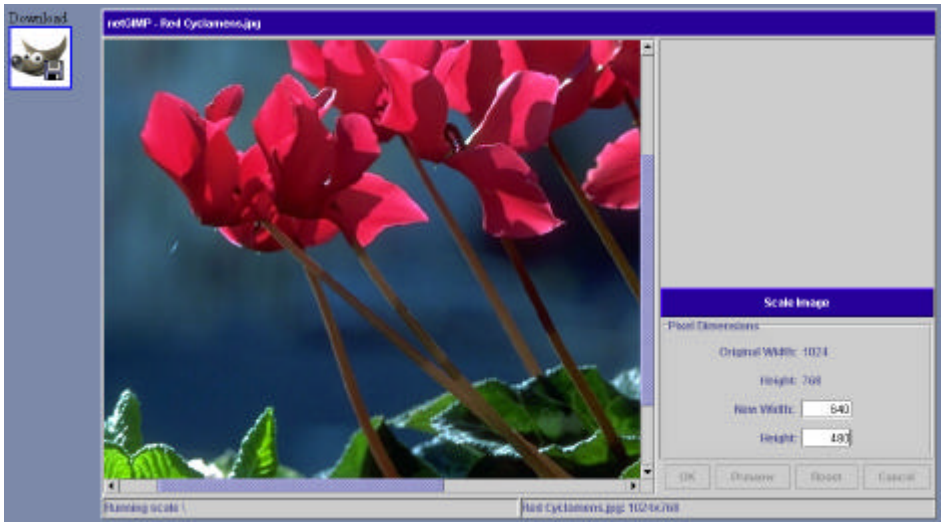
3.5.5 Applet - netgimpApplet.java

.

Appleten er hovedvinduet for netGIMP. Alt brukeren vil gjøre med bildet som er lastet opp vil skje gjennom appleten. Det brukeren kan gjøre er å kjøre forskjellige filtere. Disse deles opp i to grupper:

- Parameterløse filtere
- Filtere med parametere.

Parameterløse filtere kjøres uten at noe tekst felt med valg kommer opp. Filteret vil da kjøres ferdig, og en vil være klar til å kjøre nytt filter. Under kjøring av det aktuelle filteret er det ikke noen mulighet til å velge eller kjøre noen andre filtere.



Ved valg av filtere som krever parametere vil man få opp en meny på venstre side. Her kan man skrive inn de ønskede parameterene som skal sendes med. Her har man fire hovedvalg:

- Man kan sette de parameterene en ønsker og så velge **OK**. Man vil da kjøre filteret med de medsendte parameterene, og vente til filteret er ferdigkjørt. Når da filteret har kjørt ferdig, vil menyen av valg fjernes, og det forandrede bildet vil bli satt som originalbilde, og man er klar til å velge et nytt filter.

- Man kan sette de parameterene en ønsker og så velge **Preview**. Man vil da også kjøre filteret med de parameterne som er satt inn av brukeren. Når Filteret er kjørt ferdig vil ikke denne menyen fjernes. Her kan man da kjøre det samme filteret flere ganger på det aktuelle bildet som en hadde før en valgte dette filteret.

- Man kan velge **Reset**. Her vil de standard parameterene som er satt i de forskjellige tekstfeltene bli satt til standard verdier.

- Valg av **Cancel**. Her vil man hoppe ut av den menyen en er inne i, og en vil få tilbake det bildet som man hadde før man valgte dette filteret.

Disse forskjellige filterene er implementert.

- **Color Enhance**: Meny: Image – Colors – Auto
  Denne pluginen kjører automatisk metningsstrekking på hver fargekanal i bildet. Dette gjøres ved at en finner maks og min punkts verdier for hvert histogram, og strekker så histogrammet til full bredde.

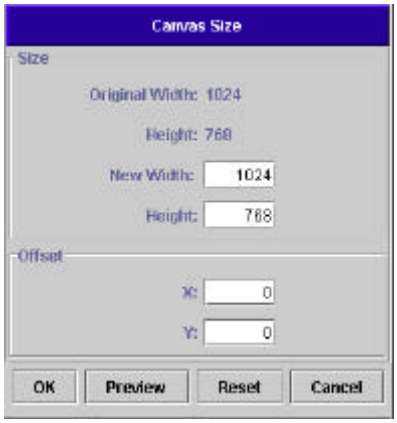
- **90 degrees**: Meny: Image – Transforms – Rotate
  Denne pluginen roterer bildet 90 grader.

- **180 degrees**: Meny: Image – Transforms – Rotate
  Denne pluginen roterer bildet 180 grader.

- **270 degrees**: Meny: Image – Transforms – Rotate
  Denne pluginen roterer bildet 270 grader.

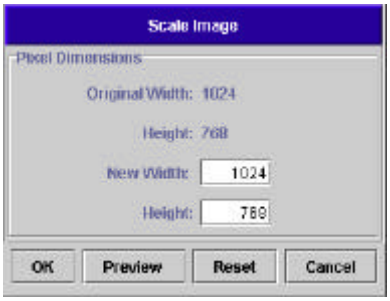
- **Canvas Size...**: Meny: Image
  Endrer størrelsen på canvas på et bilde. Dette vil si at det originale bildet ikke vil bli strekt eller krympet, mens størrelsen på bildeattributtene vil bli endret. Hvis man forstørrer bildet vil fargen rundt være hvit.
  Denne tar følgende parametere:

  Width – ny bredde på bildet.
  Height – ny høyde på bildet.
  Offset x – hvor langt i x retning bildet skal forflytte seg.
  Offset y – hvor langt i y retning bildet skal forflytte seg.



- **Scale Image...**: Meny: Image
  Skalerer bildet til en ny størrelse. Her vil bildet strekkes eller krympes til den nye størrelsen.
  Den tar følgende parametere:

  Width – ny bredde på bildet.
  Height – ny høyde på bildet.



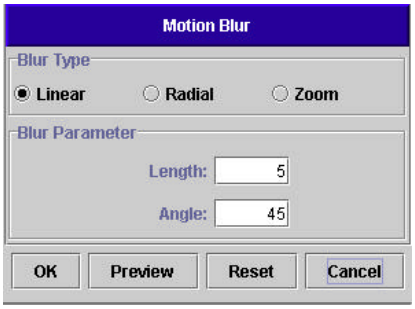
- **Flip horizontal**: Meny: Tools – Transform Tools
  Flipper bildet horisontalt.

- **Flip vertical**: Meny: Tools – Transform Tools
  Flipper bildet vertikalt.

- **Blur**: Meny: Filters – Blur
  Vil kjøre pluginen blur på hele bildet. Dette vil føre til en effekt slik at bildet blir mer utydelig.

- **Motion Blur...**: Meny: Filters – Blur
  Dettefilteret vil simulere effekten sett når man fotograferer bevegende objekt ved sakte lukketid på kameraet. Dette gjøres ved å legge til flere feilplasserte kopier av samme bildet.
  Den tar følgende parametere:

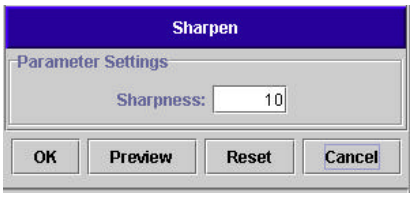
  Valg mellom Linear, Radial og Zoom.
  Length – Lengen på bluringen.
  Angle – Antall grader blur skal være på bildet.



  Linear jobber i en rett retning, Radial jobber i sirkler mot midten og Zoom vil strekke mer ved sidene for å skape en zoom effekt.

- **Sharpen**: Meny: Filters – Enhance
  Vil gjøre bildet skarpere.
  Den tar følgende parameter:

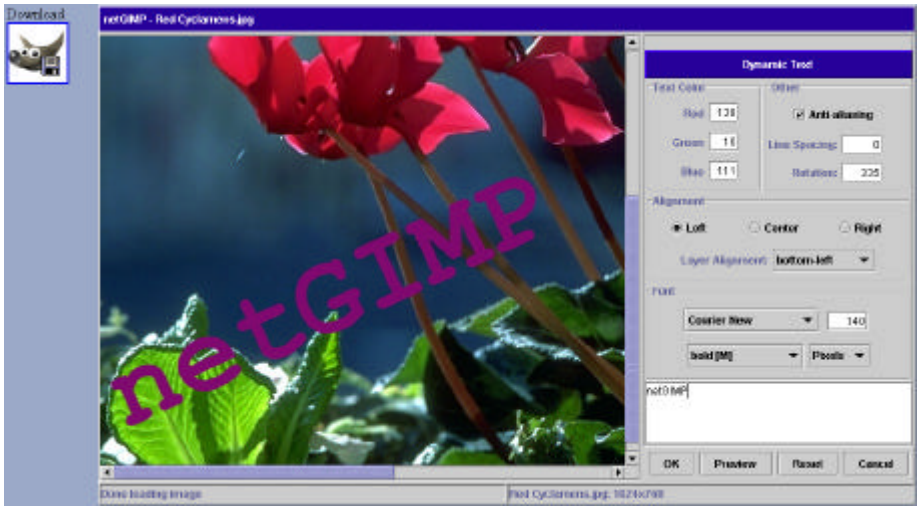
  Sharpeness – Hvor mange prosent sharpen som skal kjøres.



- **Apply Lens**: Meny: Filters – Glass Effects
  Denne vil bruke Snell's lov for å tegne en elipse formet linse på bildet.
  Denne tar følgende parametere:

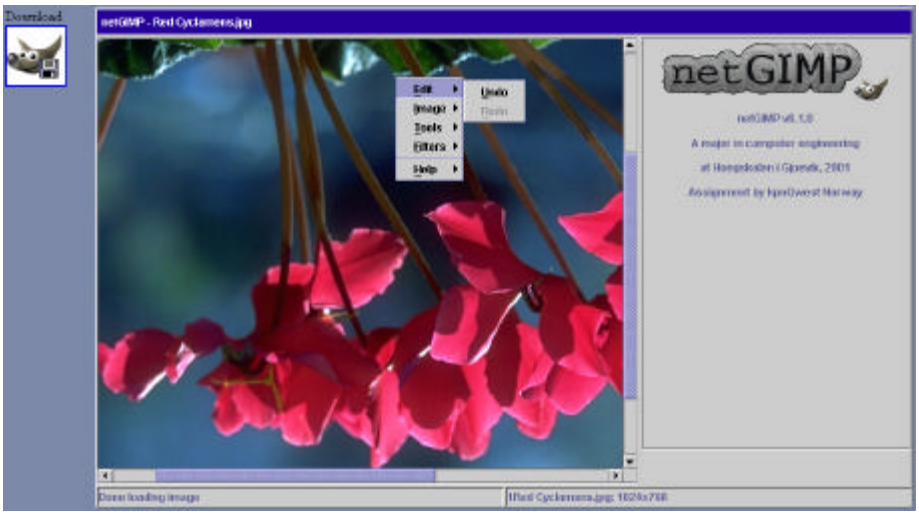
Valg mellom ”Keep Original Surroundings” og ”Set Surroundings to Background Color”

Lens Refraction index – Setter lysbrytnings indeksen, som kan settes fra 1.00 til 100.00



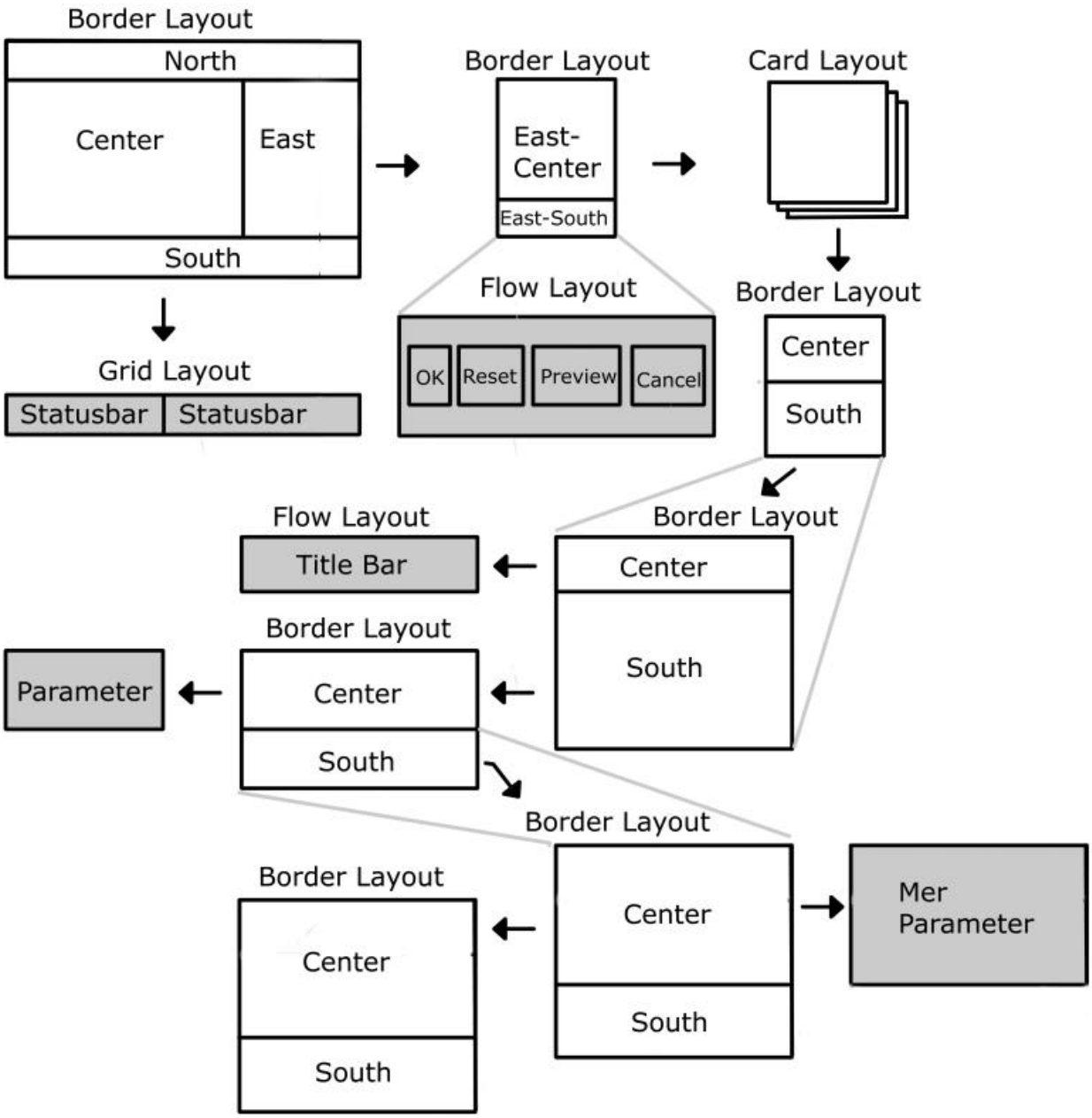
- **Dynamic Text**: Meny: Filters – Render
  Plaserer en tekst på bildet. Her kan man velge fonter, størrelse (kan settes i pixler eller points) og rotasjon på teksten.
  Den tar følgende parametere:

Text Color	– Har her tre tekstfelter (Red, Green, Blue). Disse verdiene i hvert av tekstfeltene kan variere fra 0 til 255.
Anti-aliasing	– Velger om en vil ha med antialiasing på fonter eller ikke.
Line Spacing	– Setter linje mellomrom.
Alignment	– Velger om teksten skal justeres til venstre, midjusteres eller høyrejusteres.
Layer Alignment	– Setter hvor på bildet teksten skal komme. Denne settes til topp venstre, topp midten, topp høyre, midten venstre, midten, midten høyre, bunn venstre, bunn midten eller bunn høyre.
Font	– kan velge font, fontstørrelse, som kan være i enten pixler eller points. For hver font kan man velge en style. Denne vil kunne forandre seg i forhold til hvilken font du velger.



Undo og redo muligheter i netGIMP er fullt implementert. Man kan til enhver tid angre alt det man har gjordt, og komme tilbake til det bildet en har lastet opp. Redo vil også fungere på samme måte ved at man kan angre alt man har undoet.

3.5.6 Visuel oppbygning av applet



4 Koding

4.1 Generelt

Når vi koder er det viktig å kunne få jobbet effektivt. Her har vi valgt emacs som editor for å skrive inn kode. Denne har vist seg effektivt for å holde greie på paranteser og blokker.

4.2 Kodestandarden

På Java har vi satt egne standarder for kommentering.

- Før hver funksjon kommer det en blokk som sier kort hva denne funksjonen gjør og lister opp hva slags parametere som vil bli sendt med, og hva som returneres.

- I funksjoner vil lokale variable vil ha tegnet ”_” for å binde sammen variable om går over flere ord.
  Eks: en_lang_variabel

- Globale variabler vil alltid skrives med store bokstaver.
  Eks: GLOBAL_VARIABEL

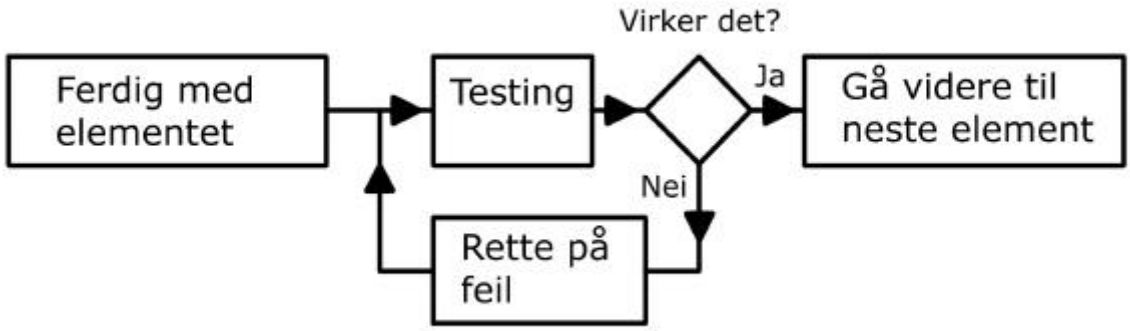
- Kommunikasjon mellom applet og servlet benyttes det en Hasharray for å sende over parametere. Dette kan sammenlignes med en assosiativ array hvor hver nøkkel består av en streng.

- Vi har valgt å holde oss til et språk, Java, ved koding på webserver.

5 Testing og Kvalitetssikring

5.1 Testing

Hver enkelt utvikler har foretatt testingen av elementene av programmet selv, testingen har skjedd når vi har gjort oss ferdig med hver enkelt modul. Hvis vi oppdaget bugs i programmet har vi rettet på dette med en gang. Vi brukte den samme metoden når det gjelder implementeringen av elementene i selve programmet.



5.2 Kvalitetssikring

Da net*GIMP* er et program lisensert gjennom *GPL*, er en av fordelene at mange kan komme med tilbakemeldinger eller egne oppdateringer og modifikasjoner. Dette har vi desverre ikke fått utprøvd da vi ikke har kommet så langt i utviklingen.

6 Installasjon

For at net*GIMP* skal fungerer slik det skal, er det nødvendig at disse pakkene er installert på serveren :

Apache-1.3.19
http://www.apache.org/
ApacheJServ-1_1-2
http://www.apache.org/
j2sdk-1_3_0
http://www.sun.com/
j2ee
http://www.sun.com/
j2sdk-1_3-beta
http://www.sun.com/

Serveren må også ha X-Server installert eller ha tilgang til en. Denne må støtte de fontene som GIMP skal generere.

Når disse pakkene er nedlastet, installert og konfigurert kan man installere net*GIMP*, det gjøres ved at man pakker ut netgimp.tar.gz. Legger html filene til html katalog, servlets til en katalog som ligger under en servlet zone. Editere konfigurasjons filen slik att dette passer med serverens instillinger.

7 Konklusjon

7.1 Prosessen

Vi startet prosjektet med å planlegge tidsforbruk, etablere arbeidsrutiner, definerer grupperegler og å avtale møtetidspunkter med veileder.

I starten av prosjektperioden var på besøk hos KpnQWest , der hadde vi et møte der vi gikk igjennom hva de egentlig forventet ut av prosjektet. Vi fikk også en liten omvisning, dog veldig inspirerende omvising hos dem.

Planleggingsfasen tok lengere tid enn det vi hadde forventet da det var vanskelig å finne ut hvordan vi egentlig skulle bygge opp systemet. Tidlig i programmerings fasen hadde vi en del problemer men ettervert som vi kom lengere ut i fasen følte vi at ting begynte å løsne og at vi så et lys i enden av tunnelen.

Vi brukte lang tid til å finne ut hvordan vi skulle kommunisere direkte med *GIMP*, løste dette problemet med å kjøre *GIMP* kommando basert istedet. Det å gjøre det på denne måten er ikke den beste løsningen men den fungerer.

I løpet av prosjekt perioden leverte vi tre statusrapporter til veileder.
Lå etter i forhold tidsplanen under hele prosjektet, grunnene til dette har vi omtalt i kapittel 8.6.1 på side. Vi også førte dagbok for hver dag, skrev også referat fra alle møter med veileder og oppdragsgiver. Vi har vedlagt malen for møtereferaten dette i Vedlegg J, mens dagboken er levert til Veileder.

7.2 Hva har vi oppnådd

Vi har klart å målet med prosjektet, vi har via web klart å bringe *GIMP* ut til den vanlige bruker, og dessuten tøyd grensene for hva man kan lage på web. Vi har også oppnådd de fleste læringsmålene vi satte oss i forprosjektrapporten, disse var :

1. Få større innblikk i *Linux* og *Open source* programmering. Lære å ta i bruk utviklingsverktøy for det vi skal drive med.

2. Vi har klart å sette oss dypere inn i klient- og serversideprogrammering (*Java*).

3. Målrettet gruppearbeid over lengre tid.

4. Lære å dokumentere arbeidet. Planlegge fremtidig arbeid. Sette fremtidige mål. Møtevirksomhet og rapportering.

5. Sette seg inn i *GIMP* og dets funksjonalitet.

Det var noen læremål vi ikke klarte å oppnå :

1. Programmere/jobbe på samme kode (*CVS*).

2. Overholde tidsfrister.

7.3 Hva har vi lært

Vi har lært mye i løpet av dette prosjektet. Det har vært mye nytt som vi har måtte lære oss. Det har vært en spennende periode med mange interessante emner å sette seg inn i. Vi vil her i dette avsnittet prøve å gi en oversikt over hva vi har vært igjennom i løpet av prosjektperioden.

Planleggingsperioden tok lengere tid en det vi hadde planlagt da vi hadde problemer med å finne ut hvilke programmerings språk/verktøy og oppbygning vi skulle bruke. Vi vurderte mellom å bruke *Corba*, *Perl*, *PHP*, *Java*, *JNI* men landet, etter mye prøving og feiling, på å bruke *Java* (*Java* Applets & *Java* Servlets) og *scheme*.

Vi har gjennom prosjektperioden vært innom noe som har vært helt nytt for oss, dette gjelder spesielt for GIMP og scheme. Men vi har også fått dypere innsikt i hvordan *Java* er bygd opp, både når det gjelder applets og servlets, og hvordan disse fungerer seg imellom.

Vi har også lært mye om GNU/Linux plattformen, og ikke minst mye om hvordan det er å drive utvikling innenfor dette miljøet.

7.4 Videre arbeid

Slik som net*GIMP* fungerer nå, startes *GIMP* for hver gang en skal kjøre et *filter* eller gjøre endringer på et bilde. Man må lagre bildet for hver gang et *filter* er kjørt, og ved å benytte *scheme* har men heller ikke tilgang til å jobbe direkte på pixler. En mulig utvidelse av net*GIMP* vil derfor kunne være:

1. Tegne pixler direkte på bilde.
2. Implementasjon av alle plugins for *GIMP* i net*GIMP*.
3. Implementasjon av *Perl*- og *Scheme* script som er registrert i *GIMP*.
4. Støtte for layers.
5. Direkte kommunisering med *GIMP*.

7.5 Evaluering

7.5.1 Gruppas arbeid

Gruppa har hatt noen problemer gjennom hovedprosjektperioden, vi har hatt problemer med å koordinere arbeide og vi har kommet frem til at vi burde ha hatt et grupperom. Den generelle følelsen er at vi da ville ha fått utrettet mer og konsentrert arbeidet. Vi har søkt mange løsninger på en rekke problemer, og har her stanget hodet i veggen, noe som igjen har medført et snev av oppgitthet i gruppen.

Vi har også opplevd at vi har hatt problemer med å finne dokumentasjon, eksempler og lignende. Det sier seg jo selv at dette ikke virker oppmuntrende i noen større grad. Til tider har man fått en viss følelse av å være pionerer, og skulle ha ønsket at det hadde vært noe mer håndgripelig å støtte seg til. Motivasjonen må kunne sies å ha vært som en berg og dalbane.

7.5.2 Gruppa om veileder

Halgeir Leiknes har til tross for at han ikke har noe særlig kunnskap om *Java*/*Linux*/*GIMP* vært til stor hjelp til de prosjekttekniske spørsmålene.

7.5.3 Gruppa om oppdragsgiver

Jon Tingvold har foruten å være vår kontakt hos oppdragsgiver også har fungert som vår veileder når det gjelder den tekniske biten. Vi er svært fornøyde med hvordan han har taklet dette. Vi må rose Jon for at han har svart på de spørsmålene han har fått, og for et svært hurtig svar på disse. Jon er en ressurs person som vi kanskje kunne ha brukt mere enn det vi har gjort.

7.5.4 Oppsummering

Vi er i den store helet fornøyd med prosjekter. Vi har klart det vi satte oss fore når vi begynte på det. Det er selvfølgelig mye vi kunne gjort annerledes og vi har hatt en del vanskeligheter i løpet av prosjektperioden, men det var forventet når vi startet opp med prosjektet.

Kapittel 8 Litteraturliste

Referanser

[A&S96] Structure and Interpretation of Computer Programs, Second Edition; Harold Ableson and Gerald Jay Sussman with Julie Sussmam.

[Som95] System Engineering, Fifth Edition; Ian Sommerville.

[WebA] The *GIMP*
http://www.*GIMP*.org/

[WebB] *GIMP* User Group
http://gug.sunsite.dk/

[WebC] Java 2 dokumentasjon
http://java.sun.com/j2ee/tutorial/api/index.html

[WebD] Introduksjon av Swing
http://developer.java.sun.com/developer/onlineTraining/GUI/Swing1/shortcourse.html

[WebE] Kode eksempler på bruk av Swing komponenter
http://java.sun.com/docs/books/tutorial/uiswing/components/example-swing/index.html

[WebF] Design av ikoner og about boks
http://java.sun.com/products/jlf/dg/high.htm

[WebG] Svar på diverse Java spørsmål
http://www.jguru.com

[WebH] MultipartRequest pakken for opplasting av filer til en servlet
http://www.servlets.com/cos/

[WebI] Kommentering av Java kode
http://www.hta-bi.bfh.ch/~daepm/javaCodingStd.html

[WebJ] Dokumentasjon og alt om PHP
http://www.php.net

[WebK] Beskrivelse av CORBA
http://www.javaworld.com/javaworld/jw-10-1997/jw-10-corbajava.html

[WebL] Høgskolen i Gjøvik
http://www.*HiG*.no/studietilbud/index.phtml

[WebM] Tech encylopedia
http://www.techweb.com/encyclopedia/

Kapittel 9 Vedlegg

Vedlegg

A	*GNU* General Public License
B	Kode fra *Java* appleten
C	Kode fra *Java* servleten
D	Kode for *scheme* script
E	Komplett oversikt over dataflyten i net*GIMP* appleten
F	Fremdriftsplanen
G	Statusrapport 1
H	Statusrapport 2
I	Statusrapport 3
J	Mal for møtereferat
K	CD med program og en Elektonisk versjon av hovedprosjektrapporten