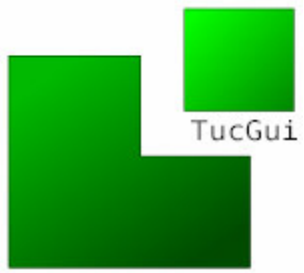
TucGui

En bit av puslespillet



Forord

Da jeg startet siste semester på datateknikk var det uvisst hva jeg kom til å skrive om. Selv om prosjektet, med tittel "Elektroniske Emosjoner", nest siste semester var interessant ønsket jeg å skrive om noe annet. Etter mye frem og tilbake endte jeg opp med en oppgave jeg var fornøyd med og som jeg føler jeg har fått mye igjen for.

Jeg har møtt på tekniske utfordringer og lest en anselig mengde litteratur. Jeg har kjempet mot Word, men heldigvis ikke så ille at jeg har lengtet etter Latex. Stressnivået har til tider vært over sunnhetsterskelen og det har vært mer enn en natt med dårlig søvnkvalitet. Men noe har jeg fått tilbake; foruten kunnskapen jeg har ervervet gjennom den litteraturen jeg har lest, har jeg gjort meg erfaringer på personlig plan som vil komme godt med videre, bl.a. hvordan jeg takler stress og hvordan jeg må strukturere dagene og legge planer for å oppnå det jeg skal.

Det du skal i gang med å lese nå er manifestasjonen av mitt siste semesters stress, svette, nedturer og oppturer. Jeg har strebet etter å skrive en oppgave som er interessant og lett å lese.

I løpet av siste semester har jeg vært i kontakt med forskjellige mennesker. Jeg vil takke Amund Tveit for veiledning i starten av semesteret, Rune Sætre for velvillig støtte mhp. GeneTuc og min familie for støtte i de mest slitsomme tidene. Stor takk til Astrid Lægreid for inspirerende, visjonære og motiverende innspill. Sist men ikke minst stor takk til Tore Amble for at han tok over veileder-rollen på strak arm og ga grundige og gode tilbakemeldinger.

Trondheim, 15. juni 2005.

Sammendrag

Tuc er et system med mange samarbeidende deler og prosesser, for eksempel; forstå setninger brukere trykker inn, lete fram svar, gi tilbake svar på forståelig form, øke kunnskapsbasen og plugge inn nye domene-modeller. Som TucGui-logoen symboliserer utgjør TucGui en del av dette systemet, mer bestemt i kunnskapsakkvisisjons-fasen.

GeneTUC er en versjon av TUC hvor domenet grovt er biologi og mer bestemt protein-interaksjoner. Systemet er utviklet ved gruppe for kunnskapssystemer ved IDI. NTNU har et modent forskningsmiljø innen biologi. Potensialet for synergieffekter er store, men til nå er ikke dette potensialet utnyttet fullt ut. Blant annet er det fordelaktig å bruke biologene til å kvalitetssikre og utvide kunnskapsbasen som ligger til grunn for GeneTUC. Til nå har denne prosessen skjedd ved hjelp av e-post og Excel regneark fra GeneTUC-utviklere på IDI til biologi-eksperter og tilbake. Dette medfører unødvendig mye ekstra arbeid.

TucGui gir, gjennom et lettfattelig visuelt grensesnitt, biologi-ekspertene direkte tilgang til GeneTUC sin kunnskapsbase og gjør dermed deler av kunnskapsakkvisisjonsprosessen raskere og med høyere kvalitet enn prosessen har tillatt tidligere.

Innholdsfortegnelse

Liste over tabeller

Liste over figurer

Innledning

Dette dokumentet er delt i tre, ekskludert appendikser, som er plassert sist. Først gjennomgår jeg det teoretiske grunnlaget hvor jeg blandt annet ser på hvilke systemer som allerede finnes for visualisering av ontologier, deretter blir TucGui systemet beskrevet mer eller mindre på samme måte som et vanlig systemutviklingsprosjekt. Avsluttningsvis trekker jeg linjene videre samt reflekterer over prosessen jeg har vært igjennom og implementasjonen.

Mitt personlige mål med diplomen-oppgaven har vært å få opp et system som skal kunne brukes og jobbes videre på, dernest har det vært å grave meg ned i materien for å faktisk få en kunnskapsmessig ballast. Med hensyn på selve diplomen-teksten har det vært et mål å ikke gjøre den for kjedelig ved å bryte opp tekst med bilder og plassere mindre vesentlige ting i fotnoter. Enkelte steder har jeg plassert et mer konkret og belysende eksempel i en boks ved siden av teksten. Jeg refererer mye, også til nettsider, siden det kan være av interesse for leseren å se nærmere på det som er omtalt.

Bakgrunnsteori

I denne delen av oppgaven redegjør jeg for det teoretiske grunnlaget for TucGui. Først rettferdiggjøres et system for samhandling i kunnskapsakkvisisjonsprosessen for GeneTUC samt utfordringer et slikt system gir. Deretter gjennomgåes hva semantisk nett, TUC/GeneTUC og ontologier, etterfulgt av hva som tidligere er gjort for å visualisere ontologier. Hvordan GeneTUC kan visualiseres gjennomgåes i kapittel "Visualisere GeneTUC". Avslutningsvis, redegjør jeg for utfordringer i forhold til graf-utlegging og metodikk i systemutviklingsprosessen.

Enorme datamengder

Med nye metoder i biologisk forskning, f.eks. genomskala screeningsteknologi, mikromatrise (se [53]) og høyhastighets DNA sekvensiering (se [52]) og det faktum at biologi er forsket på i lang tid gjør at biologi er et stort domene med mye kunnskap. Bare på en uke i mai ble det på www.nature.com publisert rundt 80 artikler innen molekylærbiologi.

Det har vært sånn at forskere ofte har hatt sitt gen eller sin organisme som de har konsentrert seg om og blitt eksperter på. Sånn må det nødvendigvis være i starten. Men nå er biologi som forskningsfelt så modent at man kan begynne å sette det hele i sammenheng. Systembiologi er navnet på fagfeltet som går ett hakk opp og ser f.eks. hvilke proteiner et sett gener resulterer i og hvilken oppførsel disse proteinene til slutt ender opp med å gi organismen de er del av[1].

ERA-NET (se [54]) er et EU initiativ som skal koordinere og samordne forskning innenfor EU. Noe som trengs skal man greie å sette sammen brikkene og se det store bildet.

GeneTUC (se kapittel "TUC og GeneTUC"), når systemet blir ferdig, vil være til hjelp for å navigere i kunnskap fra biologi-domenet.

Konsolidere kunnskap

Som nevnt er det mange som er eksperter på sine avgrensede områder. Det er ikke mulig[2] for en enkelt person å ha full detaljkunnskap og holde seg oppdatert på alt som skjer innen biologi.

En datamaskin har verken plassbegrensninger (som muligens ikke mennesket har heller) eller retensjonsproblemer. Kunnskap fra en datamaskin er alltid tilgjengelig og kan aksesseres og brukes samtidig, av mennesker og maskiner, uavhengig av geografisk lokasjon.

Det er en fordel å konsolidere kunnskap til en datamaskin.

---

[1] "Emergent behaviour", det at summen er mer enn delene.

[2] Mest pga. tid; det tar tid å lese artikler/bøker og kunnskapen må vedlikeholdes.

Kunnskapsakkvisisjon

Domeneeksperter er de som har kunnskapen. Hvordan skal man få den kunnskapen ut av dem og inn i en datamaskin slik at vi kan bruke datamaskinens fordeler til å f.eks. resonnere rundt den kunnskapen?

Selv om kunnskapskildene til GeneTUC hovedsakelig skal være artikler som automatisk er tolket av GeneTUC selv, har det vist seg (iallefall i en oppstartsperiode) at det må det noe manuelt arbeide til for at setningene skal gå igjennom. Data er matet inn manuelt, hentet automatisk fra f.eks. Gene Ontology (se [8]) eller automatisk fra artikkel-abstrakter (se [6]). Spesielt data fra [6] har måttet bli validert manuelt av domene-eksperter som har vært en pragmatisk og treg prosess med e-post og excel-ark[3].

Til syvende og sist bør kunnskapen på en eller annen måte valideres av domene-eksperter. For å få så oppdatert og så komplett kunnskap som mulig bør eksperter gies uhindret tilgang til ontologien.

Kunnskapsakkvisisjon er et fagfelt som dukket opp i kjølevannet av kunstig intelligens sin glanstid på 70-tallet (se [46]). Viktigheten av å trekke ut kunnskap fra eksperter fikk fokus. Forskjellige metoder for å trekke ut forskjellig kunnskap ble, og blir, utviklet (se "KA Techniques" i [46]).

For TUC og GeneTUC er "laddering" en teknikk som kan brukes. Domene-ekspert sitter sammen med data-ekspert[4] hvor data-eksperten stiller spørsmål på formen "Har denne klassen noen sub-klasse?".

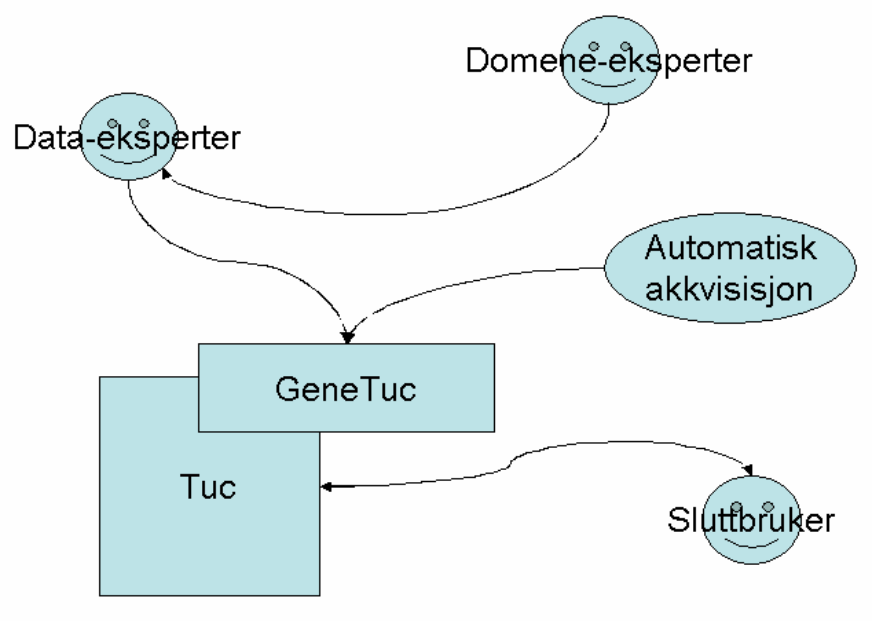
Poenget med laddering er å hjelpe domene-eksperter som ikke er vant med å tenke "klassehierarki". Biologer, som er GeneTUC sine domene-eksperter, er i høyeste grad kjent med hvordan ting kan ordnes i klasser, hierarkier og ontologier[5]. Så at kunnskapsbasen til GeneTUC blir eksponert direkte mot domene-eksperter er ikke noe problem, men man må uansett ha en felles forståelse av hvordan data skal mates inn og hva de betyr. GeneTUC sin kunnskapsbase består bl.a. av et klassehierarki (semantisk nett). Dette nettet består bl.a. av isa (eng; is a) og ako (eng; a kind of) relasjoner, som begge har en tydelig semantikk[6].

[3] Excel ark med data fra [6] ble kvalitetskontrollert av domene-eksperter, og måtte derfra manuelt legges inn i Prolog-filene til GeneTuc for å settes i produksjon.

[4] I denne kontekst kalt "knowledge engineer".

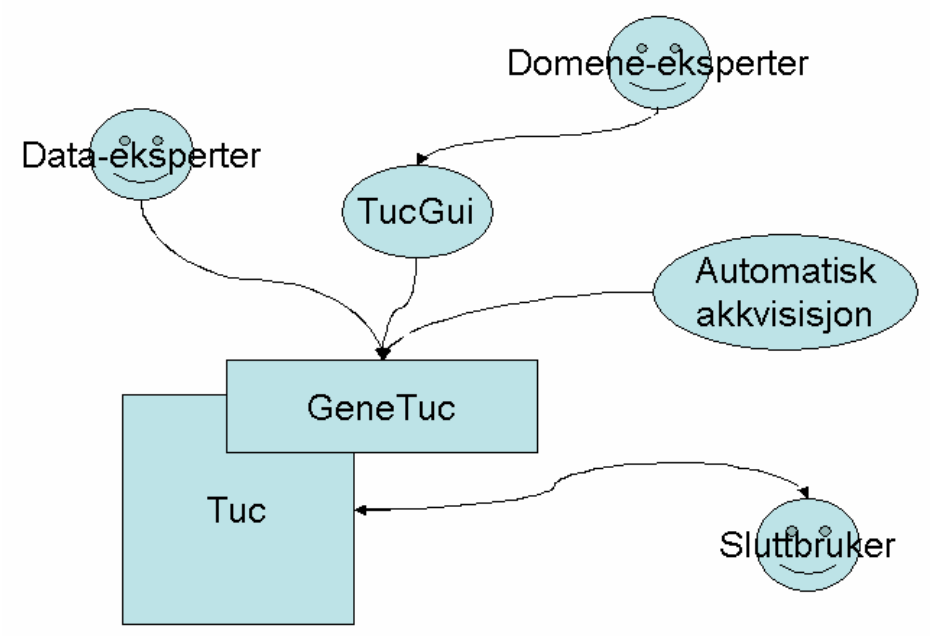
[5] Blandt annet fordi alle dyrearter klassifiseres i hierarkier og fordi biologer nå samler data i ontologier (f.eks. Gene Ontology).

[6] Semantikken til den kunnskap som er interessant er beskrevet i kapittel "Kunnskapsrepresentasjon i GeneTUC".



Figur 1 - kunnskapsakkvisisjon for GeneTUC før

For GeneTUC sin del har akkvisisjonsprosessen fra domene-eksperter til nå vært som i figur 1, mens situasjonen ideelt sett er som i figur 2. Det er denne delen av problematikken rundt kunnskapsakkvisisjon fra biologier denne diplomen omhandler.



Figur 2 - hvordan domene-eksperter kan kobles rett inn på domenemodellen til GeneTUC.

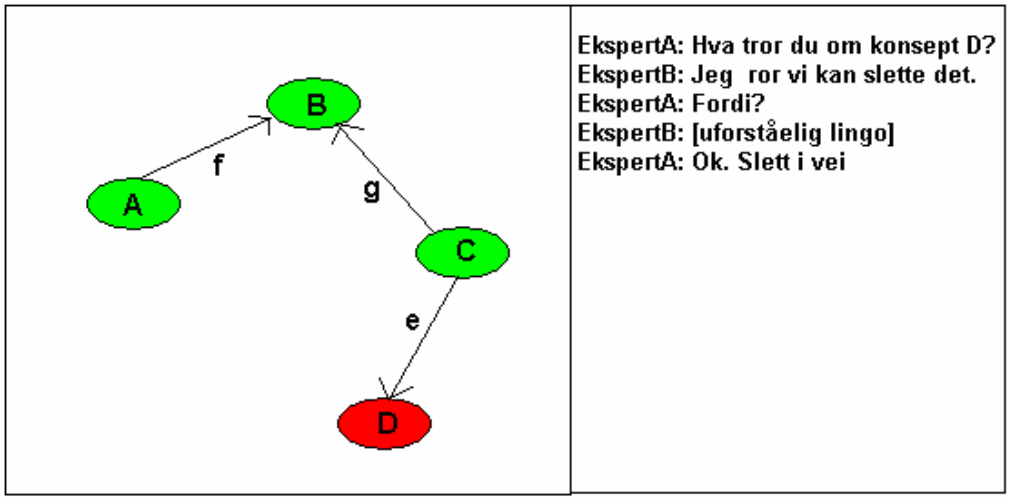
Utfordringer

Når domene-eksperter gies direkte tilgang til kunnskapsbasen oppstår det endel utfordringer. Hvordan kan man motivere biologer til å gjøre denne jobben? Hvordan skal den koordineres? Hvordan kan man mest mulig unngå data av dårlig kvalitet?

Motivasjon

Et grafisk grensesnitt er raskere (færre operasjoner trengs for for eksempel å opprette en relasjon), gir mer oversikt (man ser med en gang hva som er relatert til hva) og er morsommere å bruke enn å redigere en tekstfil.

Ved å ha funksjonalitet for sanntid tekstsamtale (chat), diskusjonsgrupper, og kunne hekte på annoteringer på noder/relasjoner (f.eks. å kunne markere noder som må sjekkes med en annen farge og relasjoner som er usikre med en prosent på hvor sikre de er. Se eksempel i figur 3) legger man til et sosialt aspekt som nok mange ville satt pris på. Ontologien ville vokst dynamisk og på en god måte siden biologer kunne testet ut sine påstander der og da. Et godt interaksjonsverktøy ville gjort prosessen å populere en ontologi morsommere, raskere og resultert i en ontologi med færre feil.



Figur 3 - konseptskisse, grensesnitt med tekst-samtale

Metadata

I et miljø hvor mange har tilgang til å legge til og endre data er det viktig å kunne spore endringer. Metadata (Dublin Core er en velkjent "minste felles multiplum"-liste med interessante metadata-felter. Se [48]) som f.eks. hvem som endret, når det ble gjort, hva det ble endret fra er viktig informasjon i tilfelle en kritisk feil-endring skjer. Andre interessante metadata er hvor sikker vedkommende som la inn en relasjon er på at den relasjonen er korrekt. Dette kunne, som nevnt, for eksempel vist seg i form av en prosent ved siden av

relasjons-navnet (for eksempel kunne relasjon E i figur 3 annoteres med "Sikkerhet: 75%"). Man kunne også hatt kommentarer fra den som la inn en relasjon/et konsept samt referanse til artikkel/kilde som støtter relasjonen/konseptet.

Minimere feil

Når flere slipper til vil volum øke, men kontroll og kvalitet muligens minke. For å minimere potensielle feil kan man kreve at de som skal kunne endre/legge til i kunnskapsbasen skal være eksperter man stoler på, og som må logge inn for å få tilgang.

Selv om eksperter legger inn data i beste mening kan det skje at kunnskapen invalideres av f.eks. nye funn fra forskningen eller at eksperten gjorde en feil da han la inn data. Man trenger en måte å gå over ontologien regelmessig for å detektere inkonsistens. For eksempel (hypotetisk) hvis regel "protein kan ikke være ligand"[7] og ny kunnskap "proteinX er_ligand_for proteinY" vil regelen kunne bryte inn å si ifra.

Man trenger også å være enige om hvilke prinsipper for innlegging av data man skal følge. Et opplagt eksempel er at et nytt konsept bør legges så langt ned i hierarkiet som mulig, dvs. gi det nye konseptet den mest spesifikke[8] forelder.

Domene-eksperter og utviklere av ekspert-system (i dette tilfellet GeneTUC) må være enige om hvilken semantikk konstruksjonene i kunnskapsrepresentasjonen har. Hva betyr egentlig "person ako agent"? Semantisk beskrivelse kan gjerne være fritekst og fyldig, men må være utvetydig og komplett. Verste-tilfelle-scenario er at domene-eksperter og data-eksperter har hver sin like, men ikke identiske, forståelse av hva som er i kunnskapsbasen.

Spesifikt for GeneTUC hadde det vært interessant at dersom en bruker (av naturlig-språk grensesnittet) fikk tilbake et svar han mente var galt, kunne han få se bevistre for hvordan GeneTUC kom frem til svaret. Dermed kunne han se hva som var feil og endre i ontologien for å få det riktig, på en måte debugging.

Ontologi

Fra filosofien er ontologi den delen av metafysikken som omhandler hva som er og hvordan det som er er relatert (se [1] for en grundig gjennomgang). I AI-verdenen er en ontologi det som eksisterer og relasjonene mellom dem; en kunnskapsrepresentasjon. I utgangspunktet det som faktisk er, men konsepter som "tid", "hendelse" og "tall" (eksisterer et tall?) kan også være representert i ontologien. I forhold til databaser kan man si at databaseskjemaet er en del av en ontologi. Jeg ser senere på hvordan en enkel biologi-ontologi kan representeres i UML. Men også data lagret i selve databasen (instanser av klasser) kan være med i ontologien.

[7] Ligand er stoff som er signalbærende.

[8] Det er mer kunnskap å vite at "P1 er en P2" og "P2 er en ting" enn at begge er ting

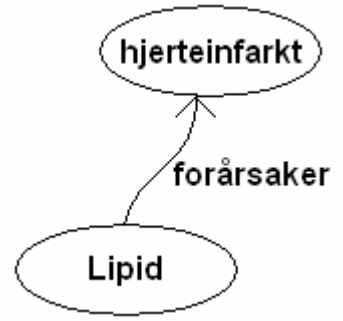
Ontologier er formalisert kunnskap. Det skal ikke være tvil om hvilken kunnskap det er snakk om og hva det betyr. Derfor er ontologier en bra ting for datamaskiner å ha.

Når kunnskap (f.eks. fra en ekspert eller hentet ut fra fri-tekst) er representert i en ontologi kan den brukes til f.eks. å:

1. Gi nye ansatte en oversikt over konsepter relevant for en bedrift..
2. Forbedre objekt-deteksjon i bilder (se [17]).
3. Gjøre at maskiner kan "forstå" relasjoner mellom ting og dermed resonnere rundt dem, som bl.a. er målet for Berner Lee sitt Semantic Web.
4. Forbedre søk, for eksempel term utvidelse; at søk etter kjøretøy også impliserer søk bil, båt, fly etc.
5. I datasikkerhet (se [16]).
6. ...

Det både eksisterer og kommer til å dukke opp massevis av ontologier for forskjellige domener, f.eks. for olje-sektoren, finans, tid, forskjellige prosesser. De er brukbare alene, men (som nevnt) får man kombinert dem og standardisert syntaks og semantikk, har man plutselig en kjempe kunnskapskilde som et dataprogram kan bruke for å resonnere i.

Semantisk nett



Figur 4 - trivielt eksempel på semantisk nett.

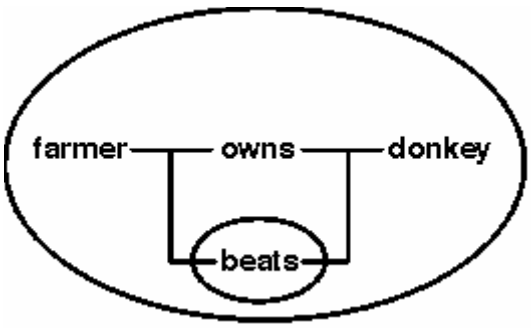
Quillian regnes for å være den første som lanserte konseptet "semantisk nett" innen AI (se [65]) og implementerte ideen i et leksikon-system; noder var ord og relasjoner knytter lignende ord sammen (kan minne om dagens WordNet). Semantisk nett ble adoptert av datidens kognitive psykologer (kognitiv psykologi[9] var veldig i vinden i 70-årene) som en interresant analogi på hvordan minnet fungerer (at kunnskap ligger i relasjoner).

[9] Kognitiv psykologi går ut på at man prøver lage abstrakte modeller for hvordan psyken fungerer, og da ofte med inspirasjon fra data-verdenen. "Korttidsminne", "langtidsminne" og "sentral prosesseringsenhet" var begreper psykologene ofte brukte.

Semantisk nett i sin tidligste form ble anklaget for ikke å ha semantikk. F.eks. påpekte Brachman (se [64]) at det ikke er entydig hva som ligger i en "x isa y" relasjon, som han mente kunne bety at x er medlem av y, at x er subklasse til y, at x er en type y eller at x er en mer spesifikk beskrivelse av y[10].

Semantiske nett er i utgangspunktet noder (ofte vist som ellipser) med piler mellom dem. Semantikken til pilene og ellipsene bestemmer man selv (f.eks. si at det kun finnes 2 typer piler: ako og isa og at alle ellipser tilsvarer unike, eksisterende entiteter). Sowa (se [40]) definerer noen typer semantiske nett, f.eks. definerende (eng; "definitional". Klassehierarki med "isa"-relasjoner og støtte for arv) og impliserende nettverk (eng; "implicational". Noder impliserer andre noder. Kan inneholde kunnskap om antakelser og årsaks-sammenheng. Eks: Bayesiansk nett). Fellesnevner er at de har enheter som er koblet sammen med streker i et nettverk og at kunnskapen ligger i relasjonene. Nettverket og delene av det er meningsbærende (har semantikk). Dette er i tråd med konneksjonismen (eng; "connectionism") (se [41]). Selv om den visuelle representasjonen kan være divers, regnes navngitte ellipser med navngitte relasjoner som den enkleste og mest vanlige form for semantisk nett (se Figur 4).

Skop (eng; "scope") er en ting semantiske nett takler dårlig. Et eksempel er at det er tungvindt å representere logiske kvantifikatorer eller negasjoner (som begge danner et skop) med mindre man introduserer nye symboler som markerer skopets grenser. Pierce (se [40]) sin Exsistential Graph løser problemet med skop for negasjoner ved å tegne en ellipse rundt de enheter som er inkludert av negasjonen (se figur 5). [42] beskriver hvordan man kan utvide semantisk nett med symbol for logiske kvantifikatorer.



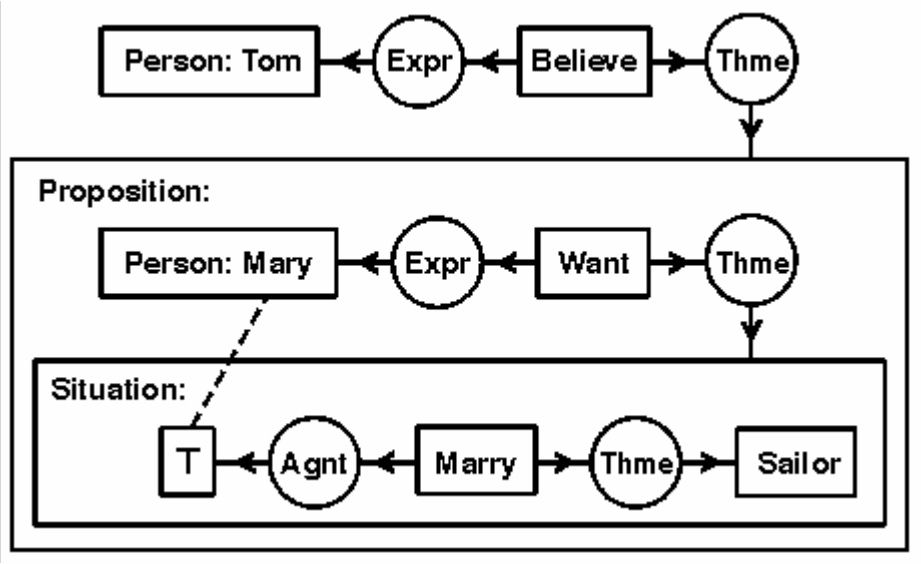
Figur 5 - viser hvordan Pierce's Existential Graph løser problemet med skop for negasjon.

Man kan utvide semantisk nett med mer eller mindre pragmatiske konstruksjoner for å lettere kunne representere noe kunnskap, men personlig er jeg mot å kludre til et så rent utseende som semantiske nett i utgangspunktet har. For TucGui er basal formen av semantisk nett tilstrekkelig.

[10] Personlig mener jeg løsningen er gode og dekkende eksempler, siden eksempler er det som til syvende og sist definerer et konsept.

To eksempler

Som nevnt er et semantisk nett det man vil det skal være. Forskjellige strukturell notasjon og semantikk til relasjoner man velger gjør at det finnes myriader av variasjoner hvorav mange blir glemt og forsvinner i tiden. J.F. Sowa har utviklet et semantisk nett som er av de mer suksessrike jeg har sett. Konseptuelle grafer (eng; "Coceptual Graphs", CG) er en notasjon (tekstlig og grafisk) for første ordens predikatlogikk. CG er bl.a. brukt til klustring av data (se [59]) og kildekodeforståelse (se [60]). Det finnes verktøy for å tegne grafene (f.eks. [61]) og CG blir kanskje en ISO standard i framtiden (for foreløpig kladd, se [63]).

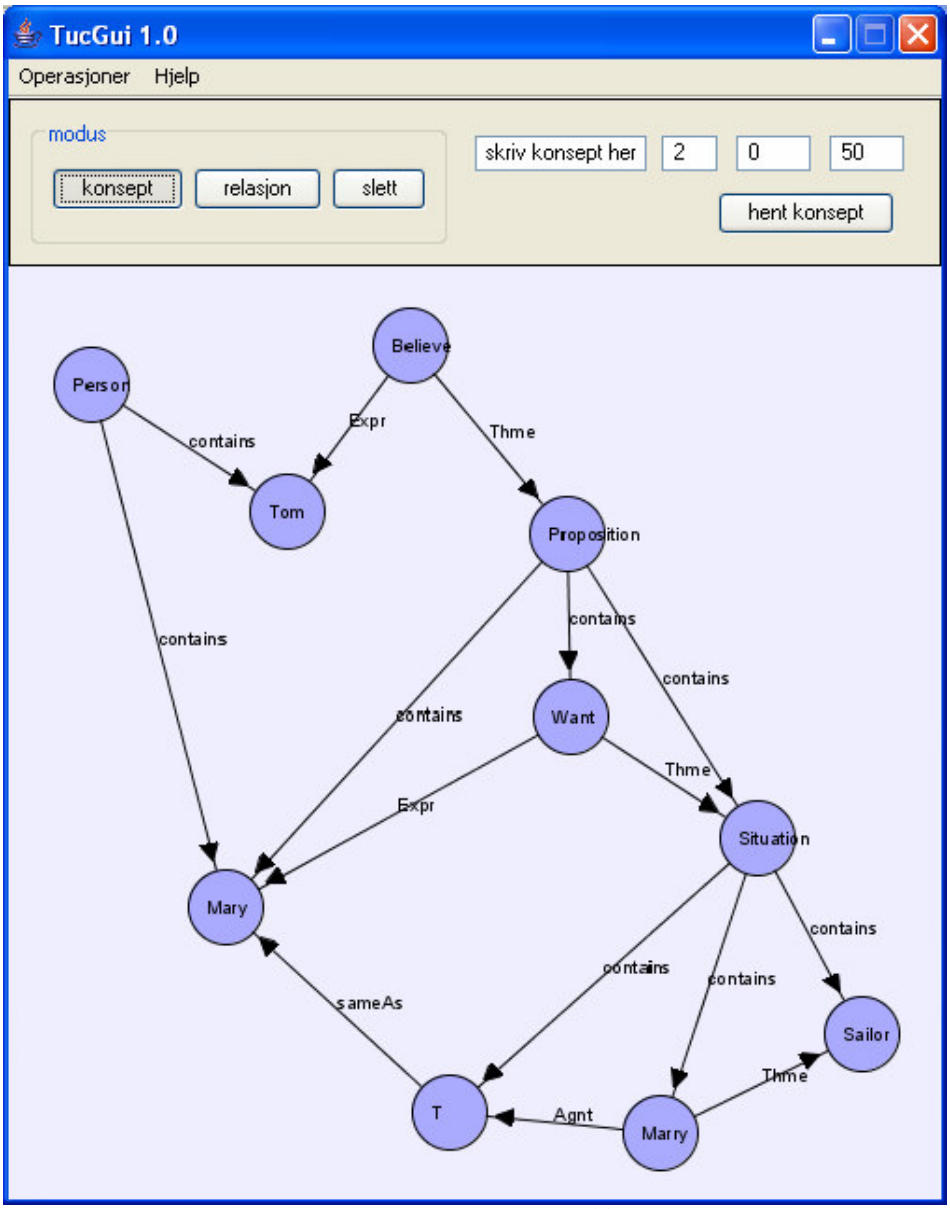


Figur 6 - eksempel på konseptuell graf[11].

Figur 6 viser et eksempel på en konseptuell graf. Meningen denne grafen formidler er "Tom tror at Mary vil gifte seg med en seiler.". Firkanter representerer konsepter og sirkler representerer relasjoner. Hvis to konsepter refererer til det samme knyttes de sammen med en stiplet linje for å vise at de er det samme konseptet. Et konsept (F.eks. "Proposisjon:..." i Figur 6) kan være sammensatt av relasjoner og konsepter, da tegnes konseptet som en stor boks rundt relasjonene og konseptene. Hvis man ser øverst i figur 6 så kan det leses sånn at "Believe" er et konsept som har en som opplever det, nemlig Tom, og som har som tema alt det som står i boksene under.

TucGui skal hovedsakelig representere binære relasjoner fra GeneTUC sin kunnskapsbase, og bruker derfor en enklere notasjon enn CG gjør. Men ekspressiviteten er ikke dårligere fordet; kunnskap fra figur 6 kan i TucGui representeres som i figur 7.

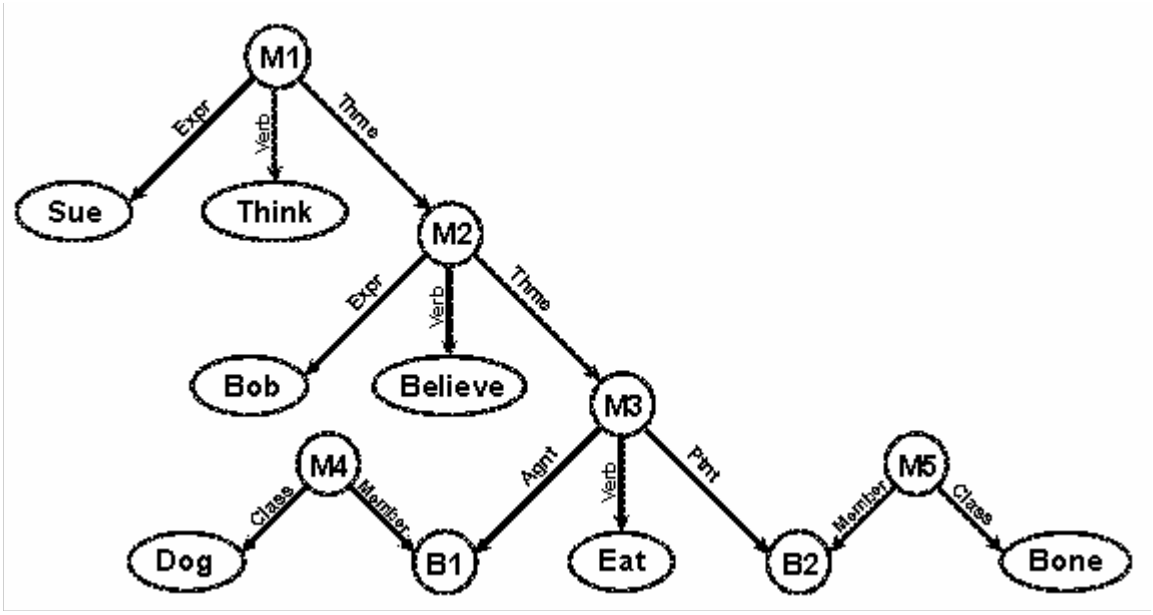
[11] Hentet fra http://www.jfsowa.com/cg/cgexamp.htm



Figur 7 - ”konseptuelt nettverk” i TucGui.

Teksten som på serveren blir generert på bakgrunn av nettverket i figur 7 kan lett transformeres til den samme tekstlige første ordens predikatlogikken som kan genereres fra nettverket i figur 6.

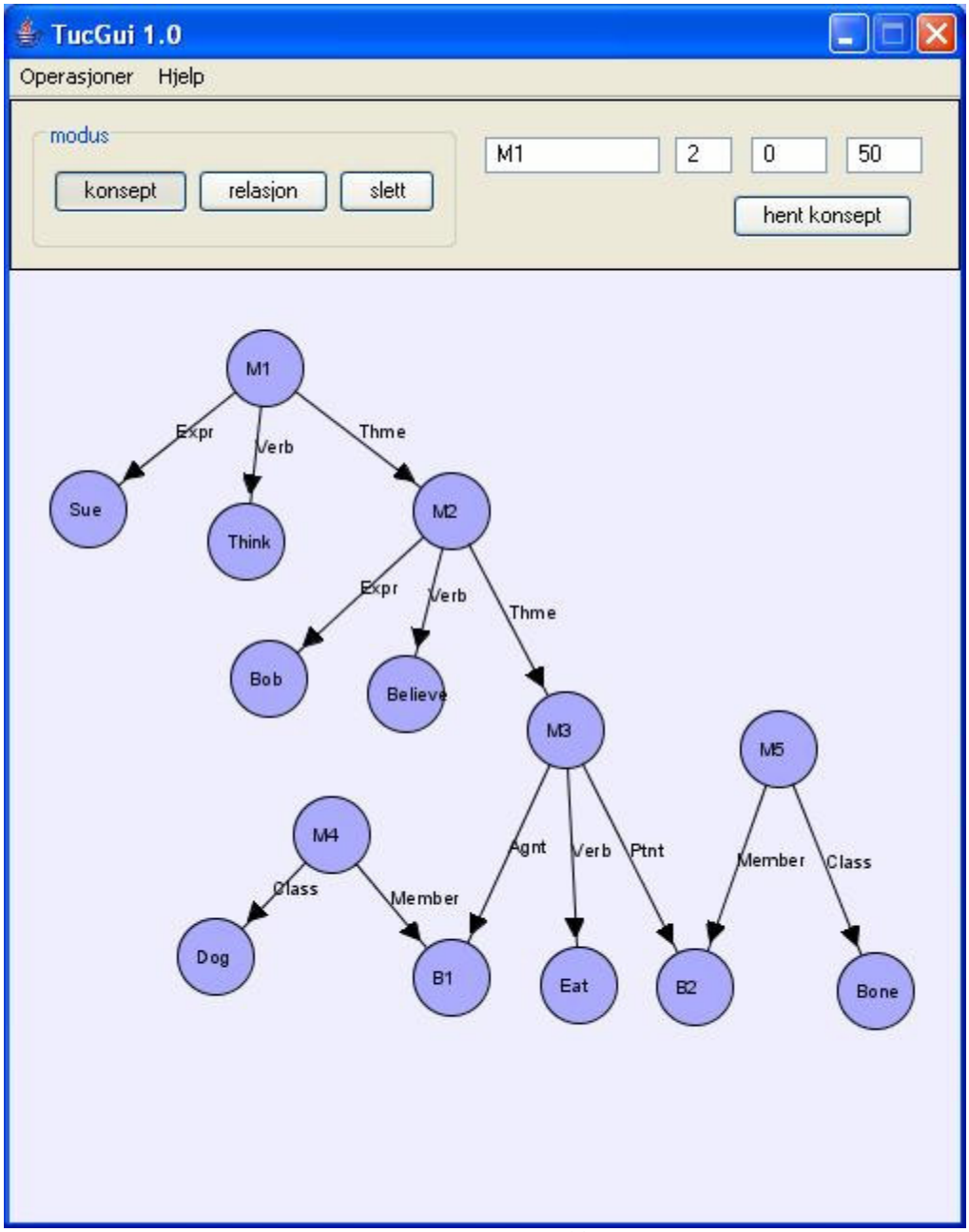
En annen formalisme er SNePS (se [70]) sin kunnskapsrepresentasjon, som er en modifisert utgave av første ordens predikatlogikk spesielt rettet mot å representere de kognitive endringer en agent opplever som følge av å ha tolket og forstått en naturlig språk setning. De ønsker altså ikke representere setningen per se. SNePS har et lite sett relasjoner som binder konsepter og relasjoner sammen (konseptuelle relasjoner er også representert som noder), for eksempel ”Expr” og ”Theme” (se figur 8) som henholdsvis betyr Experiencer (nor; ”opplever”, den som opplever verbet) og Theme (nor; ”tema”, verbets objekt).



Figur 8 - visuell fremstilling av SNePS sin representasjon av setningen "Sue thinks that Bob believes that a dog is eating a bone"[12].

Figur 8 viser hvordan setningen "Sue thinks that Bob believes that a dog is eating a bone" vil se ut internt i SNePS, mens figur 9 viser hvordan dette kan implementeres i TucGui. Teksten TucGui genererer kan lett transformeres til SNePS sin modifiserte utgave av første ordens predikat logikk.

[12] Figuren er hentet fra [40].



Figur 9 - slik kunnskapen i Figur 8 ser ut i TucGui (skjermdump).

TUC og GeneTUC

I dette kapitlet gir en oversikt over TUC generelt og GeneTUC spesielt (litt om hva systemene er i dag og hva de kan bli i framtiden). Deretter ser jeg på hvordan GeneTUC representerer sin kunnskap, både hvordan systemet faktisk gjør det (i Prolog filer og hvordan de henger sammen) og konseptuelt (klassehierarki, instanser, regler og hjelpekonstruksjoner).

The Understanding Computer

TUC (The Understanding Computer) er et system for naturlig språk tolkning og resonnering rundt det som tolkes og er laget av Tore Amble (se [4]). Man kan stille spørsmål slik man vanligvis skriver spørsmål, og får tilbake svar i naturlig språk. For brukere som verken har tid eller lyst å sette seg inn i et avansert spørrespråk, for eksempel SQL, er TUC et bra alternativ. TUC brukes i form av BussTUC daglig av Trondheims befolkning. TUC deler prosessen å skjønne hva mennesket skriver i 2 hoved-deler: tolke tekst til TQL (TUC Query Language[13]) og inferere i et domenespesifikt semantisk nett. Selv om tolking av tekst skal være domeneuavhengig har det vist seg at man for GeneTUC og BusTuc har måttet legge til spesielle verb templater, dvs. regler for hvilke subjekt og objekt et verb har lov til å ta (se kapittel "Kunnskapsrepresentasjon i GeneTUC" for mer om hvordan verb templater ser ut) for å få setningene fra hhv. biologi- og buss-domenet til å gå igjennom systemet. Men etterhvert, når TUC blir så robust at den takler alle utfordringer med referanser i setninger, metaforer og kontekstavhengighet etc. skal det bare være å plugge inn domene-kunnskap i form av semantiske nett og sette systemet rett i drift. Andre spennende ting vi kanskje vil se i TUC i framtiden:

Eva som er forsker på MTFS har funnet ut at det er mye av protein VEGF (eng; vascular endothelial growth factor) i en kreft-klump. Det er veldig interessant for Eva å vite hva dette proteinet kan forårsake; er den starten, eller er med i en lang rekke interaksjoner som forårsaker flere kreft-celler? Med GeneTuc kan hun spørre "hva påvirkes av VEGF?" og hun vil få svar "VEGF aktiverer angiogenesis". Eva tenker seg om å innser at kreftceller er avhengig av blodtilførsel for å formere seg. Angiogenesis er dannelsen av blodårer. GeneTuc har gitt henne pålitelig svar raskt.

figur 10 - case: forsker Eva

- Å be brukere forklare noe når den ikke forstår. F.eks. hvis GeneTUC ikke vet hva en (jeg bruker engelske termer her) erythrocyte er og bruker spør "Is cytochrome c present in an erythrocyte?", kunne TUC svart med "What is an erythrocyte?" hvorpå bruker svarer "Erythrocyte is a cell". GeneTUC parser informasjonen og oppdaterer det semantiske nettet (legger til klassen Erythrocyte som subklasse av klassen Cell". Dermed kan GeneTUC svare "Yes" siden alle celler har cytochrome c[14].
- Forklaring på resonnering. Hvordan kom TUC fram til akkurat det svaret den ga? Spesielt for spørsmål av type "Hva er en celles ytre effekter dersom den blir utsatt for TGFb?" (TGFb er et signalstoff involvert i vekst). Hvis GeneTUC hadde kunnet forklare hvilke stoffer som var med i signal-veien ville det vært til stor hjelp for forskere.

GeneTUC (se [5]) er et TUC-system som hadde som mål å få matet inn artikkel-abstrakter og deretter svare på spørsmål fra en bruker slik at vedkommende kan finne ut om noen av artiklene inneholder ting han lurer på og dermed er verdt å lese i sin helhet. Da blir GeneTUC brukt som en søkemotor; man vet hva man er ute etter og lurer på om det man lurer på er omtalt i et sett artikler man har.

[13] TQL er en form for predikatlogikk som er tilpasset naturlig språk.

[14] Cytochrome c er et molekyl som kan holde på elektroner og er viktig for å produsere energi-bærende molekyler (ATP, adenosine tri-phosphate) som celler bruker til å drive prosesser.

Variasjonen av informasjon man finner i abstrakter er enorm, så GeneTUC har valgt sitt fokus til å i hovedsak være protein-interaksjoner (proteomics), dvs. hvilke proteiner som påvirker hverandre (se figur 10 for en praktisk applikasjon).

Å få oversikt over hvilke proteiner som påvirker hverandre er uhyre komplekst og det forskes mye på det. All denne forskningen publiseres på løpende bånd, og er ofte kun tilgjengelig i selve publikasjonene. Det finnes databaser som inneholder protein-interaksjoner, men de oppdateres først mye senere og da gjerne manuelt. GeneTUC skal kunne få matet inn artikkel-abstrakter, oppdatere sin egen kunnskapsdatabase, og deretter kunne svare på spørsmål relatert til artikkelabstraktene den nettopp lastet inn. Ved å akkvirere abstrakter over tid øker kunnskapsbasen kontinuerlig, og GeneTUC kan svare på fler og fler spørsmål.

GeneTUC vil være en nyttig veiviser etter hvert som at kunnskapen om proteininteraksjoner øker i mengde og kompleksitet.

Men som jeg kommer tilbake til, og som er motivasjonen for denne oppgaven; TUC kan bomme i prosessen fra artikkelabstrakt til semantisk nett. Derfor må eksperter kvalitetssikre data og her kommer TucGui inn.

Kunnskapsrepresentasjon i GeneTUC

GeneTUC sin kunnskap består av et klassehierarki, en liste med instanser av klassene, en regeldatabase[15] og predikater for å håndtere setningstolking. TucGui skal kun beskjeftiges med å visualisere klassehierarki og utvalgte relasjoner (verb og adjektiv), så disse delene er fokus her.

Konseptuelt kan man dele inn GeneTUC sin kunnskap i to deler; et klassehierarki og templater for hvilke argumenter de forskjellige gramatiske konstruksjonene tar (f.eks. at verbet skrive ikke kan ta instans av klassen fugl som subjekt). Av templatene har vi kommet frem til at det er adjektiv og verb som er interessante å visualisere.

I klassehierarki er klasser er knyttet sammen med ako (eng; a kind of) relasjoner og instanser er knyttet til klasser med isa (eng; is a) relasjoner. Mer detaljert informasjon om klasser oppnåes vha. has_a (eng; has a), apo (eng; a part of).

GeneTUC har sin kunnskap lagret som predikater i Prolog filer. Når GeneTUC startes er det kun filen semantic.pl som lastes inn, derfra refereres det til andre filer[16]. Filene semantic.pl og facts.pl er av størst interesse, siden de bl.a. inneholder klassehierarkiet for GeneTUC, men det finnes andre filer som er lagt til pragmatisk. F.eks. en fil for ny kunnskap hentet av Alchymoogle (for mer om Alchymoogle, se [12]).

I produksjonsmiljø er det kunnskapsbasen i minnet (i form av TQL; TUC Query Language) som brukes. Denne trenger nødvendigvis ikke inneholde identiske data med de data som

[15] Fra setninger i abstrakter (f.eks. fra Medline) som tolkes abstraheres det ut regler, f.eks. Protein ABC aktiverer Protein 123.

[16] Dette gjør at man må passe på i hvilke filer man putter et predikat, siden dersom et predikat med samme navn finnes i flere filer, og begge filene lastes inn, blir det ene forkastet (kalles shadowing i Prolog) med mindre man tar spesielt hensyn til det.

finnes i filene, siden GeneTUC kan ha lest inn og tolket ett eller flere abstrakter, og økt sin kunnskap pga. dem[17].

Filene kan ha kommentarer, både enkel linje og flerlinjet.

Tabell 1 viser relasjoner, med faktiske eksempler hentet fra GeneTUC sin kunnskapsbase, som er interessante å visualisere. I kapittel "Visualisere GeneTUC" ser vi hvordan disse relasjonene ser ut i TucGui.

Tabell 1 - interessante relasjoner og deres semantikk.

Relasjon	Predikat	Semantikk
Is a	isa	A isa B betyr at A er instans av klassen B. F.eks. blood isa liquid.
A kind of	ako	A ako B betyr at A er subklasse av B. F.eks. forebrain ako object.
Has a	has_a	A has_a B betyr at A har egenskapen B. F.eks. cell has_a chromosome.
A part of	apo	A apo B betyr at at er en fysisk del av B. F.eks. park apo city.
Intransitive verb template	iv_templ	Tar ikke objekt. Iv_templ(verb, subjekt). F.eks. iv_templ(drive,vehicle) som sier at verbet kjøre tar subjekt av type kjøretøy.
Transitive verb template	tv_templ	Tar ett objekt. Tv_templ(verb,subjekt,objekt). F.eks. tv_templ(analyse,agent,thing) som sier at verbet analysere tar subjekt av type agent og objekt av type ting.
Verb complement	v_compl	Tar subjekt og preposisjonsobjekt. V_compl(verb, subjekt, preposisjon, objekt). F.eks. v_compl(bind,thing,to,thing). forteller at ting kan binde til ting. For eksempel setningen "protein integrin binder lipider til cellevegg". Her er "protein integrin" subjekt, "binder" verb, "til" preposisjon og "cellevegg" objekt.
Adjective template	adj_templ	Tar substantiv og adjektiv. Adj_templ(adjektiv, substantiv). F.eks. adj_templ(african,thing) som sier at ting kan være afrikanske.

[17] Det er kun kunnskap fra filer som fysisk ligger på disk som det implementeres visualisering/editering for, men med TucGui sin generelle datakilde-arkitektur er det ingenting i veien for å hente ut TQL og visualisere det.

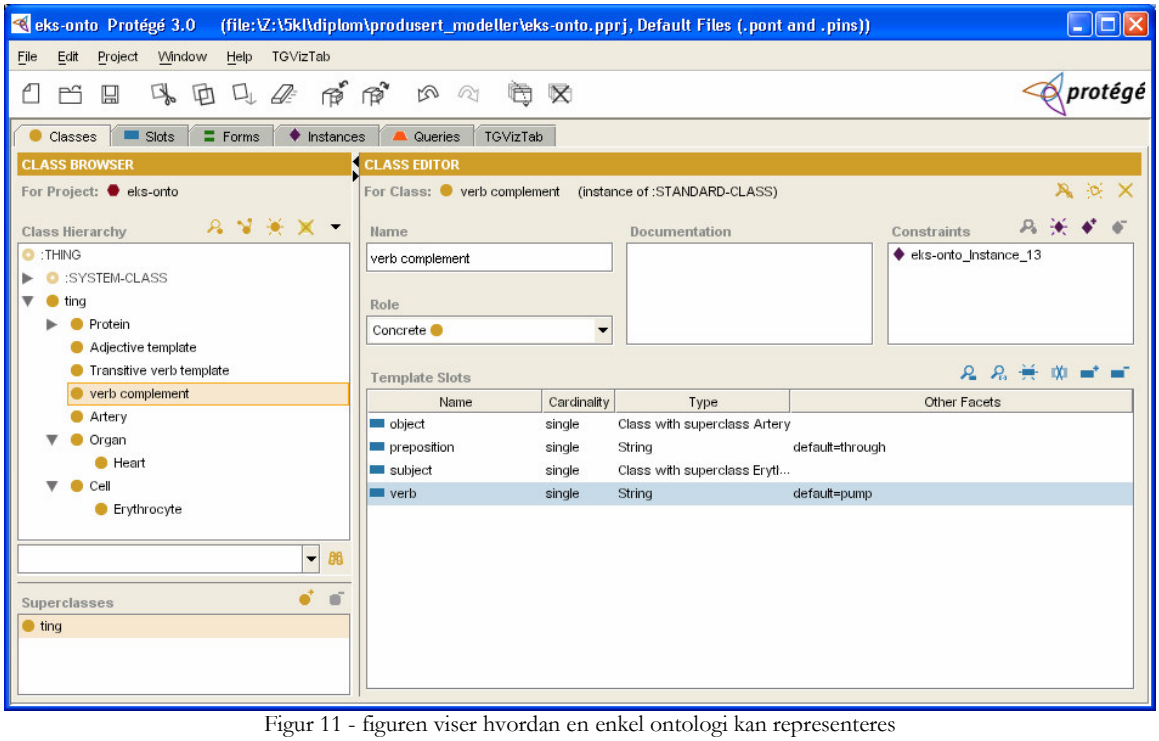
Visualisering av ontologier - en oversikt

Datasystemer som er modne kjennetegnes ofte av at de har gode verktøy, deriblandt verktøy for visualisering. I dette kapitlet ser jeg på et lite utvalg av hvilke verktøy som allerede finnes for visualisering og editering av ontologier samt hvorfor disse ikke er optimale for GeneTUC. I noen eksempler under brukes relasjoner fra GeneTUC (f.eks. ako, isa, has_a og tv_templ), se kapittel "Kunnskapsrepresentasjon i GeneTUC" for disse relasjonene sin semantikk.

For Protégé og UML lager jeg en enkel ontologi relevant for GeneTUC. Poenget med denne ontologien er at den er ment for å gi en felles plattform for å kunne sammenligne verktøyene og at den inneholder alle relasjoner fra GeneTUC som vi skal visualisere.

Protégé

Protégé (se [73] og [7]) er et verktøy for bygging av ontologier som har har blitt utviklet siden 1987. Som mange andre verktøy fra den akademiske verden var programmet lite brukervennlig, men det kommet langt og brukes nå av utallige miljøer. En ulempe med Protégé i forhold til TUC er at kunnskapsbasen til TUC blant annet skal vedlikeholdes av folk som ikke har data som primærområde og Protégé er litt å sette seg inn i og har mange funksjoner som er unødvendig for TUC.

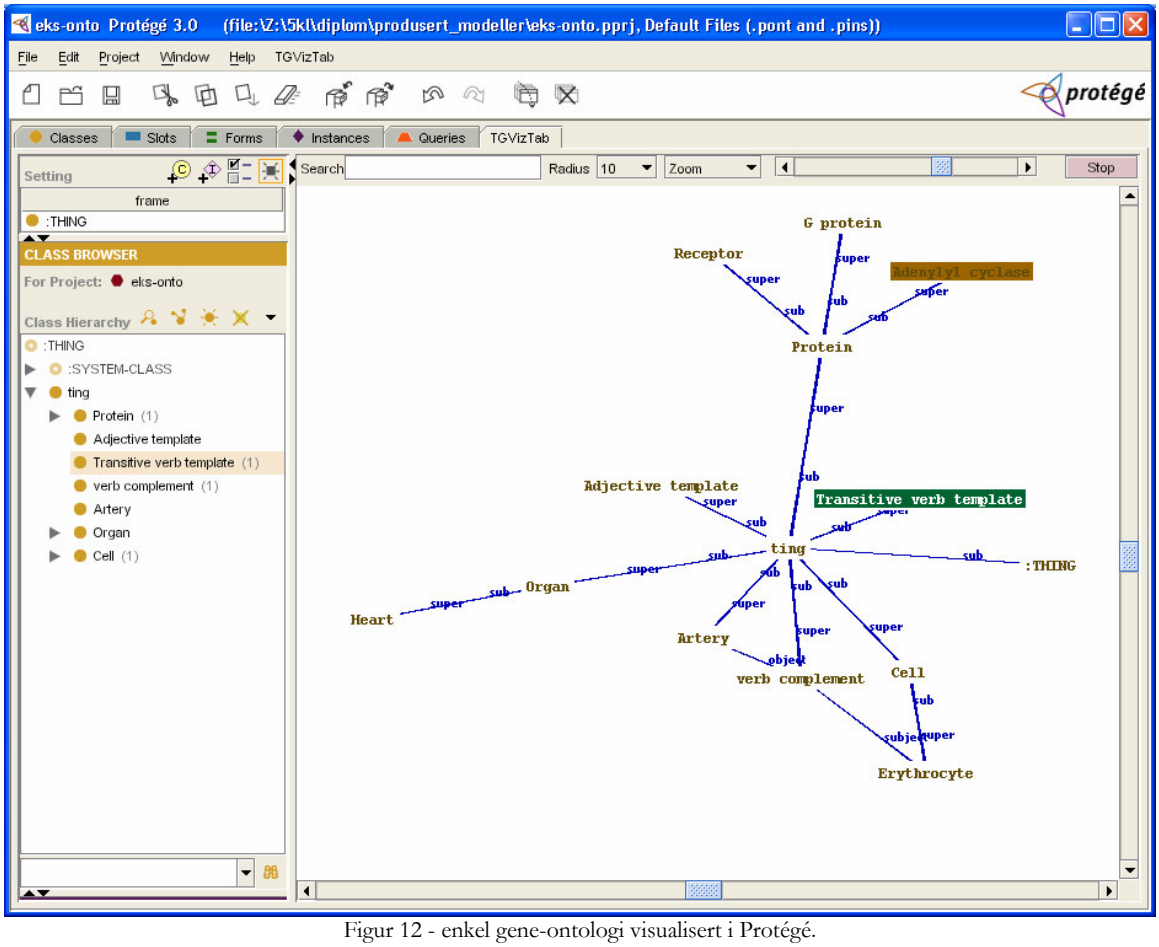


Figur 11 - figuren viser hvordan en enkel ontologi kan representeres i Protégé.

Figur 11 viser hvordan en liten ontologi kan representeres i Protégé. Klassehierarkiet (til venstre i figuren) har en egen tab separert fra instanser. Klasser kan ha slots (i blått i senter i figuren), som tilsvarer attributter i UML-klassediagram. Slottene kan ha begrensninger og standardverdier. UML-klassediagram kan har kardinalitet på relasjoner, men i tillegg til kardinalitet har Protégé begrensninger på slots, hvor man kan ha lange logikk-setninger som definerer begrensninger. Standardverdier i Protégé tilsvarer initieringsverdier i UML.

Den eneste relasjonen Protégé har er super-/sub-type relasjonen. Derfor måtte templat-relasjonene modelleres som klasser, som for så vidt er greit, men ikke optimalt. De kunne eventuelt vært subklasse av klassen "relation". For GeneTUC er det fordelaktig å kunne visualisere relasjoner som piler mellom klassene.

Det er mulig å få visualisert klassehierarkiet ved å bruke en plug-in (se [49] og [51]). Grafen blir lagt ut med spring-force algoritmen, den samme som jeg bruker i TucGui. Grafen er kun for visualisering (kan ikke editeres direkte).



Figur 12 - enkel gene-ontologi visualisert i Protégé.

Protégé er tungvindt å bruke, det er alt for mange bokser man må innom for å gjøre det man vil (i motsetning til gode uml-verktøy som f.eks. gratisprogrammet Gentleware Poseidon

UML[18], hvor man kan gjøre alt direkte i selve diagrammet). Noen av dialogboksene var til og med tomme, som tyder på at det er noe som ikke er helt som det skal være.

UML

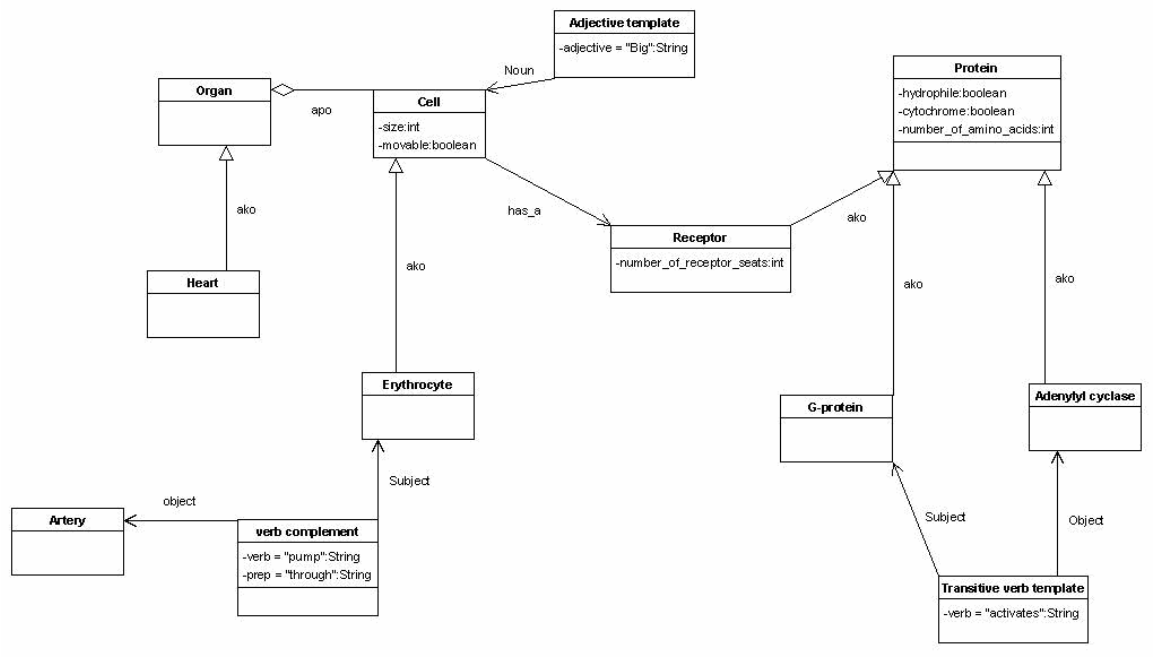
En kanskje uortodoks, men like fult brukbar måte å visuelt lage ontologier på er å bruke OMG (Object Management Group) sitt UML klassediagram (Unified Modelling Language, se [69]). UML er en åpen standard for visuell modellering av flere aspekter hovedsakelig i forbindelse med systemutvikling (bl.a. usecases, interaksjonsdiagram, sekvensdiagram og klassediagram), men som også kan brukes for å lage ontologier (se [43]). UML er velegnet for å beskrive klasse-hierarkier, og har egne symbolder for relasjoner som "komposisjon", "subklasse" og den mer generelle "assosiasjon", men igjen ingen (naturlig nok) relasjon for templater, slik at de må modelleres som i Protégé; som klasser. Det finnes utallige programmer man kan lage UML klassediagrammer med (f.eks Gentleware Poseidon UML, Rational Rose, Microsoft Visio og Borland Together).

Hver klasse kan ha attributter som beleilig vises inne i boksen som representerer en klasse. Vanligvis har kunnskapsrepresentasjoner for kunnskapssystemer blitt definert på linær form (tekstlig), f.eks. KL-ONE. UML var i utgangspunktet kun et visuelt språk, men har nå en lineær-form representasjon; XMI (XML Metadata Interchange). Figur 13 viser hvordan en enkel GeneTUC[19] ontologi kan representeres som UML klassediagram. Klassediagrammet inneholder ingen isa relasjoner (som er instanser og dermed ikke hører hjemme i et klassediagram[20]), men har egen notasjon for komposisjon ("a part of" relasjonen i GeneTUC) og subklasse ("a kind of" relasjonen i GeneTUC). For øvrige relasjoner er det benyttet "dependency" relasjoner med navn som identifiserer relasjonstypen, f.eks. "has_a".

[18] Community versjonen er gratis. http://www.gentleware.com/index.php.

[19] Se kapittel "Kunnskapsrepresentasjon i GeneTUC" for relasjonenes semantikk.

[20] Man kunne selvsagt brukt isa relasjoner som da ville vist instanser, men det mener jeg går på tvers av UML-klassediagram som **klasse**-diagram.



Figur 13 - en enkel ontologi i UML.

Det som er fint med UML-klassediagrammer er at det er et språk mange er kjent med og at det finnes utallige verktøy for å lage modeller og at det er velegnet til å lage ontologier med. På den negative siden finnes ingen veldefinert transformasjon til mer brukte kunnskapsrepresentasjons-språk, f.eks. første ordens predikatlogikk (som GeneTUC bruker i Prolog-filene) eller TQL (Tuc Query Language). TucGui er direkte koblet på GeneTUC sin kunnskapsbase og løser dermed det problemet.

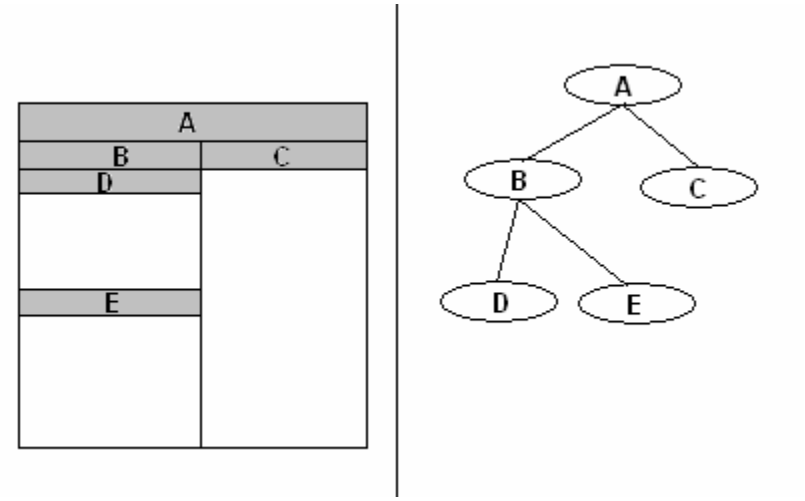
Diverse verktøy

Reactome (se [10]), som bruker Protégé for å bygge den underliggende ontologien, er en kunnskapsbase med biologiske prosesser som også er visualisert. Visualiseringen er statisk men interaktiv (man kan klikke seg rundt, men ikke legge til eller endre data).

Tanoue, Matoba, Yoshikawa, Uemura har et prosjekt (se [28]) som ligger tett opptil TucGui (visualisere Gene Ontology ved hjelp av Scalable Vector Graphics). Forskjellen er at de vil vise grafikk i browser ved å bruke ASV[21] og det er naturlig nok ikke editeringsmulighet. Jeg vurderte også ASV/browser alternativet, og startet såvidt, men fant fort ut at plugin og scripting ikke var en fleksibel og god plattform (mer om det i kapittel "Om systemutviklingen"). En annen ting de også har støtt på er problemet med at GO-data ikke har posisjonsinformasjon (altså x/y-koordinater for konsepter i grafen). De stipulerer ingen løsning, men det gjør jeg. Mer om det i kapittel "Konsept-utlegging".

[21] Adobe SVG Viewer, ASV (www.adobe.com/svg), er de facto standard svg plugin for browsere.

Ketan Babaria (se [55]) foreslår en annen metode for å visualisere Gene Ontology. Babaria bruker tree-maps[22], som er en alternativ måte å se en graf på hvor alle barna til en node X er firkanter inne i en firkant som node X (se figur 14)



Figur 14 - sammenligning mellom hierarkisk utlegg (høyre) og tre-kart utlegg (venstre).

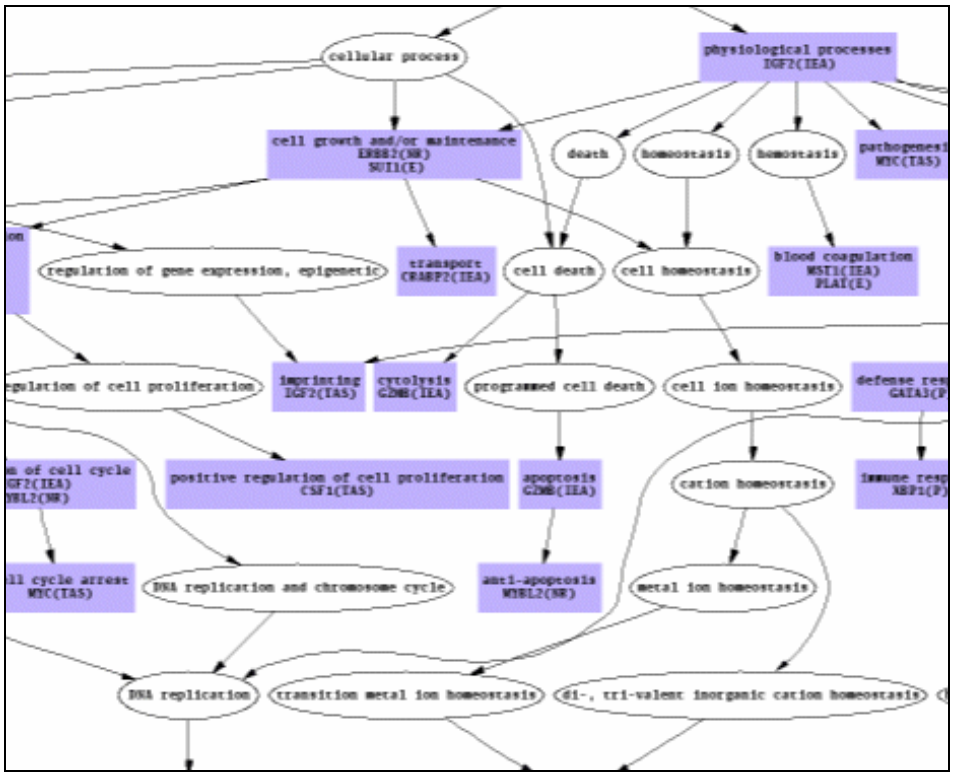
Tree-map skiller ikke tydelig mellom relasjons-typer, som hierarkisk utlegg har muligheten til i og med at relasjonene vises som streker og dermed kan ha navn..

Jaap Kemps har satt sammen et system for å visualisere Wordnet (se [58]). Wordnet består bl.a. av synword relasjoner mellom ord, dvs ord som har lik betydning. Dermed gies betydningen av et gitt ord utifra hvilke ord det har synword-relasjoner til. Det eneste Kamps implementerer er algoritmen som bestemmer hvilke ord som skal hentes[23]. Grafen er statisk og ikke editerbar.

GeneInfoViz (se [56] og [57]) er et online verktøy for å visualisere gene-interaksjoner hentet fra GeneOntology (se figur 15). Man kan få frem grafer for hver av de tre topp-nodene i Gene Ontology (cellular component, biological process og molecular function). Figur 15 viser et utsnitt av en graf hvor man har spurt etter hvilke biologiske prosesser (blå bokser) et sett gener (også skrevet i de blå boksene) er medvirkende i, og hvordan alt dette forøvrig er koblet sammen i gene ontrologien (hvite ellipser). Pilene tilsvarer "a kind of"-relasjonen i GeneTUC. Grafene som genereres ser pene ut takket være GraphViz, men de er kun for visualisering, altså statiske og kun lesbare. GeneTUC trenger et system for å også kunne endre på kunnskapen man ser.

[22] Brukes ofte når nodene har egenskaper som er sammenlignbare. F.eks. hvis nodene i figur 14 hadde hatt egenskapen "størrelse" kunne dette fysisk bli vist i tre-kart diagrammet.

[23] Kamps MPL (minimum path length) tilsvarer maxDistance i TucGui (se kapittel "Løsning på navigasjonsproblem").



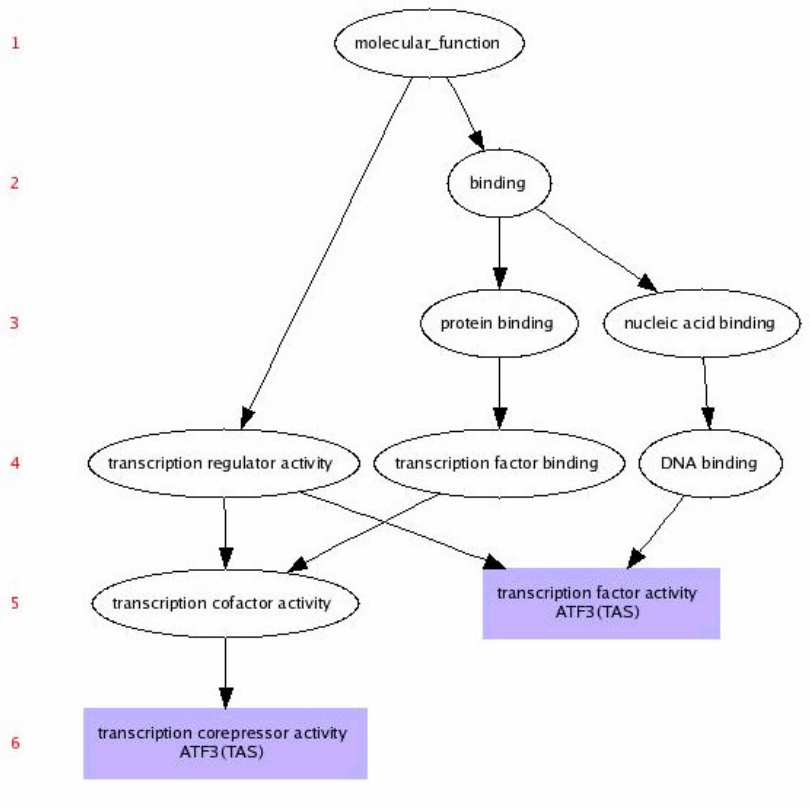
Figur 15 - utsnitt av en graf generert av GeneInfoViz.

Man kan spørre etter en eller flere gener. Som regel flere for å se om de kan ha noe med hverandre å gjøre. I figur 16 har jeg spurt etter ett spesielt gen: ATF3[24]. Av grafen kan man lese at ATF3 har to kjente molekylære funksjoner (blå bokser):

- transcription factor activity, dvs. at den er med på å legge til rette for å avlese DNA
- transcription corepressor activity, dvs. at den er en av flere komponenter som ned-regulerer avlesing av DNA

Man kan også se hvor disse molekylære funksjonene ligger i Gene Ontology. I figur 16 ser vi f.eks. at transcription factor activity er DNA binding, som er nucleic acid binding, som er binding, som er molecular function.

[24] Cyclic-amp-dependent transcript factor, en transkripsjonsfaktor (dvs. en faktor som har noe å si for avlesing av DNA) som er avhengig av et intracellulært signal; syklisk amp.



Figur 16 - graf for molekylære funksjoner ATF3 er implisert i.

Senere ser vi hvordan samme type data som er visualisert i figur 16 kan visualiseres i TucGui.

Oppsummering

Det finnes et utall forskjellige verktøy for å lage ontologier og for å se på dem. I dette kapitlet har jeg sett på et utsnitt av dem. Systemer for å lage ontologier (for eksempel Protégé og UML) er ofte overlesset med funksjonalitet som en domene-ekspert ikke har tid til å sette seg inn i. Verktøyer for å se på ontologiene (for eksempel Reactome og GeneInfoViz) er enkle og ofte pene, men mangler (opplagt nok) å kunne endre på ontologiene. Sammen med det faktum at GeneTUC sin kunnskapsbase er en kompleks struktur av Prolog filer med predikater (som naturlig nok ingen av editerings-verktøyene kan eksportere til) resulterer det i et behov for et mer tilpasset verktøy; TucGui.

Visualisere GeneTUC

GeneTUC sin kunnskapsbase inneholder forskjellig data som kan visualiseres, men ikke alt er like interessant å visualisere. I dette kapitlet ser jeg på hva i GeneTUC som er interessant å visualisere og viser noen eksempler samt ser på utfordringer i forhold til navigering i visualiserte data. Først tar jeg opp utfordringer i forbindelse med å visuelt navigere store datamengder, deretter redegjør jeg for to relevante områder hvor TucGui i fremtiden kan

brukes. Deretter ser jeg på det området TucGui er ment for og til slutt legger jeg frem et dekkende eksempel for den kunnskapen som skal representeres.

Det er mange predikater, hvorav endel er pragmatiske og ikke interessante i denne sammenheng. De relasjoner og konsepter som i vår kontekst, nemlig eksponere GeneTUC til biologer, er interessante å visualisere er de relasjoner som finnes i klasse-nettet, og de relasjoner som indikerer verb-relasjoner samt adjektiv relasjoner, siden de viser relasjoner mellom noder i klasse-nettet (f.eks. ting aktiverer ting). Isa, ako, has_a og apo er relasjoner fra klasse-nettet som er interessante. Tv_templ (eng; transitive verb template), iv_templ (eng; Intransitive verb template), v_compl (eng; verb complement) og adj_templ (eng; adjective template) er templat-relasjoner som er interessante. Se "Kunnskapsrepresentasjon i GeneTUC" for semantikk.

Alle relasjoner vi ser på her er binære, som gjør at de passer fint inn i et semantisk nett.

For å skille de forskjellige relasjonstypene brukes et prefiks etterfulgt av "##" i relasjonsnavnet. Prefikset kan være hva som helst, men er valgt slik at det best representerer det underliggende templatet[25]. Det etter "##" er relasjons-spesifikke data (for eksempel verb komplementer (v_compl) også et preposisjon).

Navigere i store datamengder

Det er ikke interessant eller overkommelig å visualisere hele GeneTUC sitt klassehierarki og verb-/abjektiv-templater. Den kunnskapen vi ønsker å visualisere kan være uhåndterlig selv om man kun ser et mindre utsnitt av den. Vi trenger lure måter å se interessante data, uten at de tåkelegges av andre data som akkurat da er mindre interessante. Man må velge fokus. I TucGui implementasjonen er dette løst vha "fokus konsept" og andre pragmatiske ideer, se kapittel "Løsning på navigasjonsproblem". Under stipuleres andre/kompletterende løsninger.

- Filtre
    - Filtre som tar vekk alle av en type relasjon (f.eks. ønsker man ikke se has_a relasjoner).
    - Filtre som tar vekk alle relasjoner bortsett fra en type (f.eks. ønsker man kun se has_a relasjoner).
- Selektiv konseptvalg – kun se konsepter (og deres relasjoner) som er subklasse til et valgt konsept.
- Se korteste vei i klassehierarkiet (via ako relasjoner) fra ett konsept til et annet.
- Se hvilke klasser som har en gitt egenskap (henter alle has_a relasjoner inn til et attributt).
- Se hvilke klasser som har en gitt verb-relasjon.

[25] Domene-eksperter må dermed læres at f.eks. tv_templ er prefiks i relasjonsnavn som viser verbrelasjon mellom subjekt og objekt.

Visualisere TQL

Figur 17 viser hvordan GeneTUC internt (i form av TQL; TUC Query language) representerer setningen ”This is a cytokine that regulates not only immune and inflammatory responses but also the growth of some tumors, including prostate carcinomas.”. I TucGui kan dette visualiseres[26] som vist i figur 18. Dette er en applikasjon som gjør at man kan se GeneTUC sin interne representasjon på en bedre måte.

```
E: this  is a cytokine that regulates not only immune and
  inflammatory responses but also the growth of some tumors , including
  prostate carcinomas.

E: E:
.....................................................................
% TQL:

sk(2)isa cytokine
sk(3)isa response
adj/immune/sk(3)/real
adj/inflammatory/sk(3)/real
regulate/sk(2)/(sk(3),sk(5))/sk(4)
event/real/sk(4)
sk(5)isa growth
sk(6)isa tumor
nrel/of/growth/thing/sk(5)/(sk(6),sk(7))
sk(7)isa carcinoma
adj/including/sk(7)/real
adj/prostate/sk(7)/real
event/real/sk(1)
be2/it/sk(2)/sk(1)
.....................................................................
```

Figur 17 - Skjermdump fra GeneTUC parsing av reell setning.

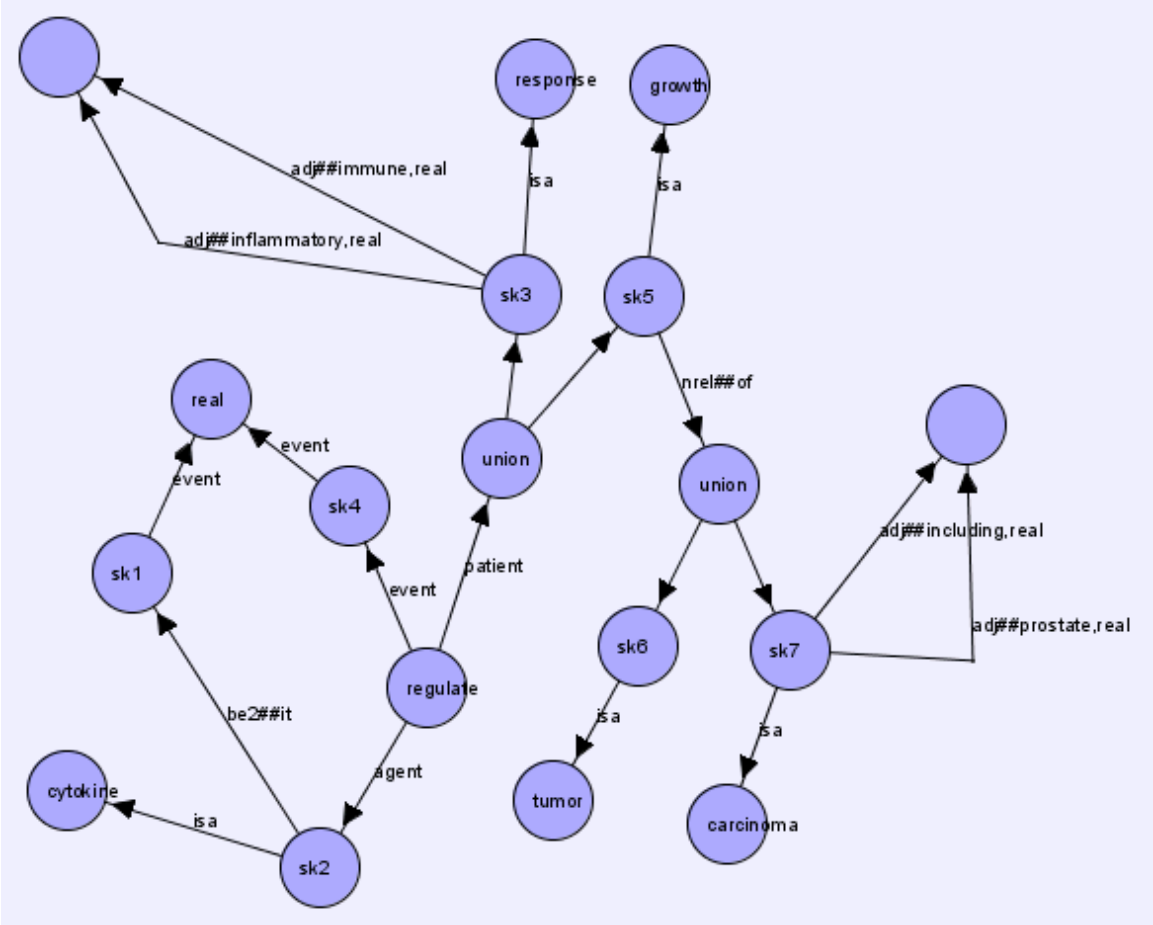
I figur 17 leses f.eks. ”adj / immune / sk(3) / real” som ”adj er et predikat med 3 argumenter hvor det første er immune, det andre er sk(3) og det tredje er real”. Det er ikke så veldig viktig hva som menes med f.eks. sk(3), bortsett fra at det er det tredje argumentet til et adj predikat.

Figur 18 viser hvordan data fra figur 17 kan visualiseres. Isa relasjonene er direkte overførbare. Adj relasjonen tar ett argument og kobles derfor til en tom node. Som nevnt har relasjonsnavn prefiks for å identifisere relasjonstype. Noen relasjonsnavn har også deletegn ”##” som etterfølges av eventuelle ekstra parametere relasjonen trenger. Nrel relasjonen har fått egen node[27] siden den relasjonen er mellom tre klasser. Nrel har tre ”argumenter”; patient, agent og

[26] TucGui er i utgangspunktet ment for å visualisere GeneTuc sin kunnskapsbase i sekundærlager (tekstfiler på harddisk). Eksempelet med TQL er tatt med for å vise en mulighet for hvordan GeneTuc sin interne kunnskapsrepresentasjon kan visualiseres.

[27] Dersom TucGui faktisk skulle vært brukt til å visualisere TQL ville det muligens vært fordelaktig med et annet utseende for noder som er relasjoner (som i utgangspunktet er piler).

event. En annen ting som er verdt å merke seg er at ”union” noden ikke er en klasse, men en måte å slå sammen klasser på[28].



Figur 18 - figuren viser hvordan kunnskap fra figur 17 kan representeres i TucGui.

I utgangspunktet skal GeneTUC greie å forstå setninger selv, så det å manuelt mate inn det som vises i figur 18 skal ikke være nødvendig (selv om det ikke er noe i veien for at man kan gjøre det). Som visualiseringsverktøy for å se sammenhenger gir TucGui her en tydelig og god oversikt over nøyaktig hvilken kunnskap som er trukket ut av en setning.

Visualisere bevistre

Når en bruker spør om noe kan det hende han ikke er enig i, eller forstår hvorfor systemet kom fram til den konklusjonen han får presentert. Da er et bevistre nyttig. figur 19 viser hvordan et bevistre kan se ut i GeneTUC[29]. Brukeren legger inn kunnskap (første linje), systemet svarer med hvilke predikater som ble tolket ut fra setningen. Deretter stiller brukeren

[28] Som for regulate-relasjonen måtte det her bli brukt et annet symbol for å tydelig vise at ”union” ikke er en klasse.

[29] Å få skrevet ut bevistre i GeneTuc er ikke implementert. Eksemplet er simulert.

spørsmål. Spørsmålet blir oversatt til intern spørring og vist fram ("<= event(X),agent...") og bevistreet blir skrevet ut. Reglene som resulterte i bevistreet under er:

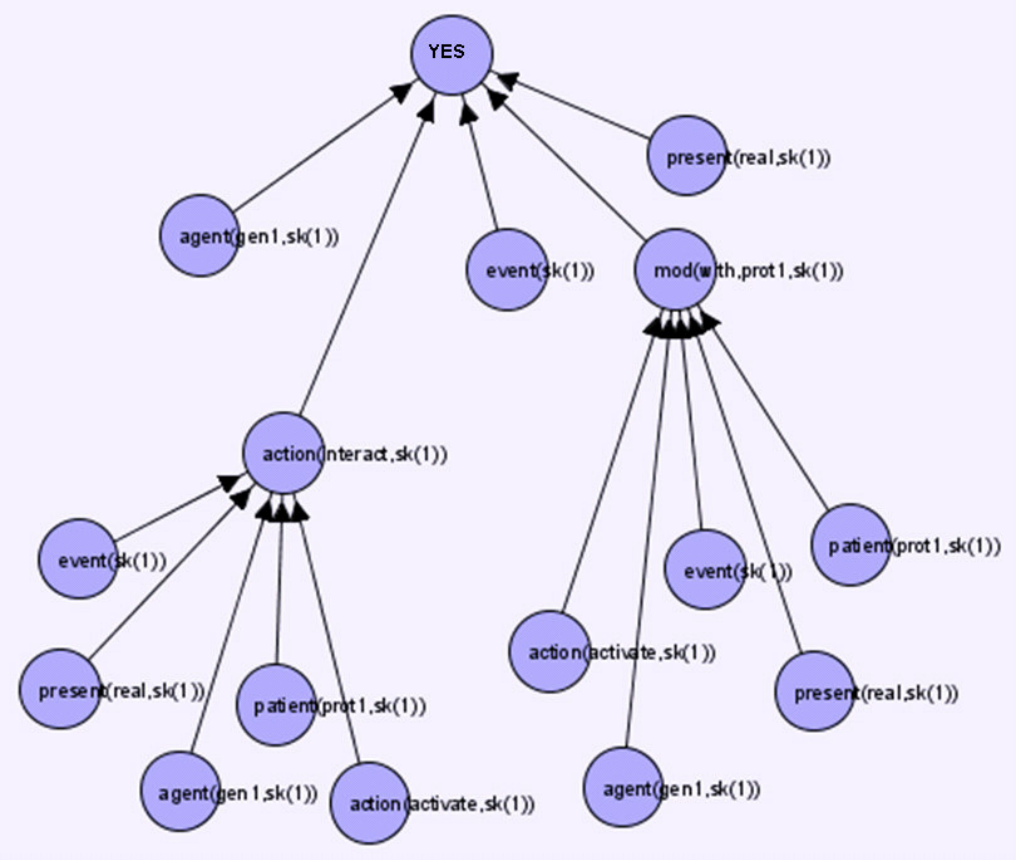
```
(activate/Gene/Protein/A,Gene isa gene,Protein isa protein)
==>
interact/Gene/A.

(activate/Gene/Protein/A,Gene isa gene,Protein isa protein)
==>
srel/with/thing/Protein/A.
```

Det som står før "==>" er premisser (kommaseparert) og det etter er konklusjon (standard implikasjon). Formatet er "relasjon/arg_1/…/arg_n" eller "arg1 relasjon arg2".

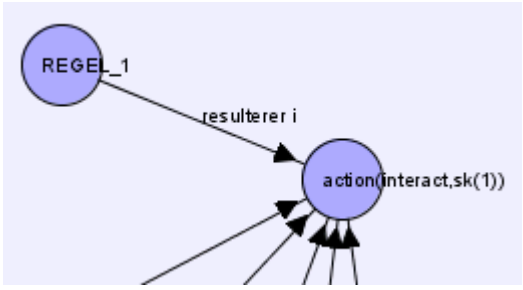
```
N: gen1 activates prot1.

event(sk(1))
agent(gen1,sk(1))
present(real,sk(1))
action(activate,sk(1))
patient(prot1,sk(1))



N: does  gen1 interact with  prot1 ?

 <= event(X),agent(gen1,X),present(real,X),action(interact,X),mod(with,prot1,X)


*** Proof Tree ***

event(sk(1))
agent(gen1,sk(1))
present(real,sk(1))
action(interact,sk(1))
    event(sk(1))
    agent(gen1,sk(1))
    present(real,sk(1))
    action(activate,sk(1))
    patient(prot1,sk(1))
mod(with,prot1,sk(1))
    event(sk(1))
    agent(gen1,sk(1))
    present(real,sk(1))
    action(activate,sk(1))
    patient(prot1,sk(1))

********************

YES
```

Figur 19 - spørring og tilhørende bevistre på tekstform.

Bevistreet på tekstlig form er nytt i seg selv, men et visualisert bevistre er mer oversiktelig. I TucGui kan bevistreet i figur 19 visualiseres som illustrert i figur 20. Relasjon fra X til Y (f.eks. fra event(sk(1)) til YES) leses "X er delmål av Y". Alle nodene (teksten som står i dem) er sanne, sånn at "YES" noden, som er endelig konklusjon, er sann fordi alle dets barn-noder er sanne. Dersom man har navn på reglene kan det også være interessant å vite hvilke regler som er brukt. Da vil det være fordelaktig å inkorporere regel-navnene i visualiseringen. Hvis vi for eksempel kaller de to reglene nevnt over for "regel_1" og "regel_2" vil resultat av applikasjon av "regel_1" være "action(interact,sk(1))" og man kunne vist dette som i figur 21.



Figur 20 - figuren viser hvordan et bevistre kan visualiseres i TucGui.



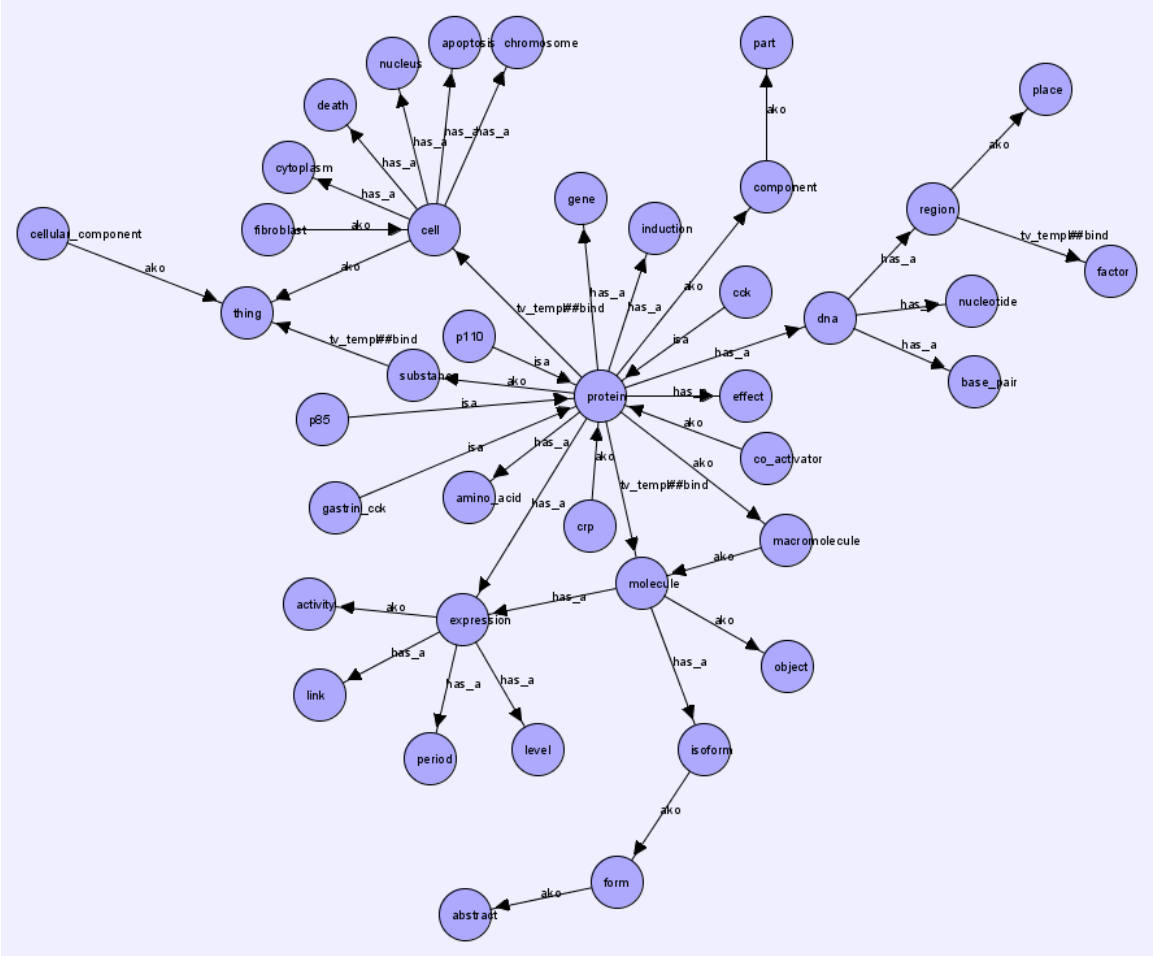
Figur 21 - hvordan man kan vise hvilke regler som er benyttet i resonneringen.

Som nevnt innledningsvis er fokus for implementasjonen av GeneTUC visualisering av kunnskapsbasen (klassehierarki og verb-/adjektiv-relasjoner). I det følgende ser vi på hvordan dette visualiseres.

Visualisere GeneTUC sin kunnskapsbase

Figur 22 viser hvordan et utsnitt[30] data fra klasse-nettet og noen transitive verb relasjoner (merket med tv_templ[31]) visualiseres. Dette er reelle data fra tekstfiler (semantic.pl) som også kan editeres.

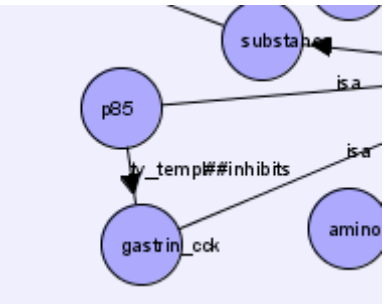
Et scenarie er at setningen "Lymphocyte has_a receptor" (som f.eks. kan opptre i et abstrakt) ikke tolkes rett av GeneTUC siden konseptet "lymphocyte" ikke eksisterer, da vil man kunne gå inn i TucGui å lage en relasjon som vist i Figur 24 og setningen vil bli tolket rett. Et annet scenarie er at en bruker spør "Does p85 inhibit the function of gastrin cck?". I utgangspunktet er det uvisst, men hvis en ekspert vet det er sånn at p85 inhiberer funksjonen til gastrin cck, kan vedkommende gå inn i TucGui og legge til relasjon som vist i figur 23. Dermed vil svaret på spørsmålet i fremtiden være "yes".



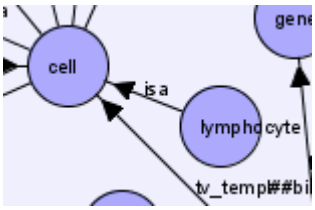
Figur 22 - visualisert kunnskap fra GeneTUC sin kunnskapsbase.

[30] Dataene er manuelt valgt for å gi et representativt og relevant utvalg fra GeneTuc sin kunnskapsbase.

[31] Tv_templ er kun en intern tag for å gjenkjenne transitive verb templater. ## skiller taggen fra det som faktisk er relasjonen, f.eks verbet "bind" i Figur 22.



Figur 23 - ny relasjon lagt til.



Figur 24 - nytt konsept lagt til og knyttet til klasse-nettet.

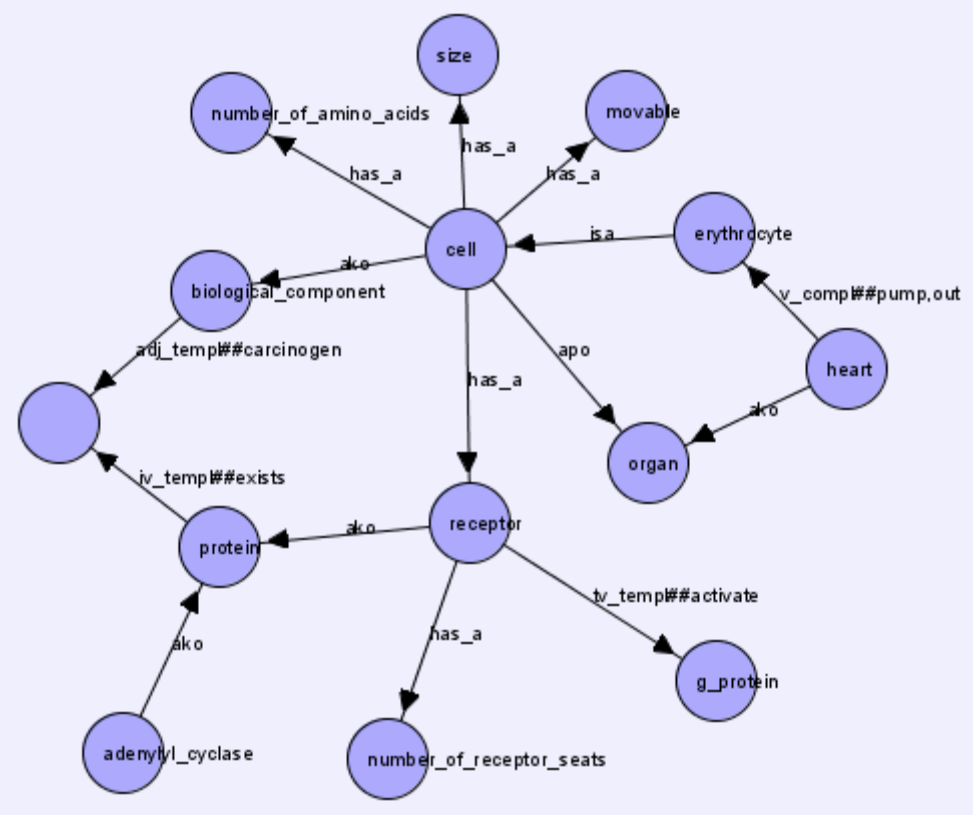
Et dekkende eksempel

Figur 25 viser alle interessante relasjoner fra GeneTUC sin kunnskapsbase. Ontologien som er vist ligner på ontologien som er bruk som eksempel for Protégé og UML-klassediagram i kapittel "Visualisering av ontologier - en oversikt".

Klassehierarkiet består av klassene (noen av dem har egenskaper, som vises med has_a relasjoner) protein, adenlyly cyclase, receptor, g-protein, organ, cell, heart og erythrocyte som er instans av cell.

De templatene som eksisterer i denne lille ontologien er et verbkomplement (for verbet pump), et intransitivt (for verbet exist), et transitivt verb templat (for verbet activate) og et adjektiv-templat (for adjektivet carcinogen[32]). Adjektiv-templater og intransitive verb templater er unære, så de er koblet til en blank node.

[32] Et carcinogent stoff er et stoff som er kreftfremkallende. Biologiske komponenter kan være mer eller mindre carsinogene.



Figur 25 - samtlige interessante relasjoner i GeneTUC sin kunnskapsbase slik de ser ut visualisert i TucGui.

Å kunne gå inn i GeneTUC sin kunnskapsbase og visuelt se og endre relasjoner umiddelbart når det er noe GeneTUC ikke forstår er en kjempefordel og letter prosessen med å heve GeneTUC sitt kunnskapsnivå betraktelig.

Konsept-utlegging

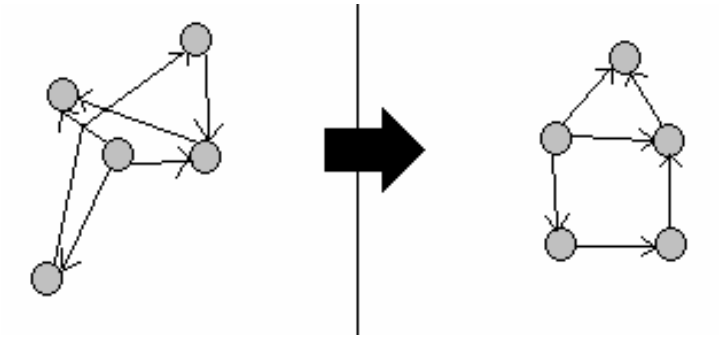
Å plassere ut en graf er et interessant algoritmisk problem; Hvordan bør den se ut til slutt? Er det mulig å legge den ut uten å få kryssende kanter? Er det en node som naturlig bør være plassert over de andre? Er det noen noder som bør klynges sammen?

I bunn og grunn må man tenke gjennom hva man vil vite av grafen man skal legge ut og hvordan den bør legges ut slik at den best mulig eksponerer akkurat det.

Kriterier for estetikk

En graf som har så få kant-kryssinger som mulig, er nogenlunde samlet, men så separert at nodene ikke overlapper og hvor nodene har sine naboer nære er noen kriterier man kan stille for en god graf (f.eks. figur 26). Har grafen et naturlig fokus-punkt (f.eks. et spesielt protein

man vil undersøke, eller en spesiell verb-relasjon) kan det gi bedre oversikt å plassere dette sentralt og mindre viktige konsepter mer perifert.

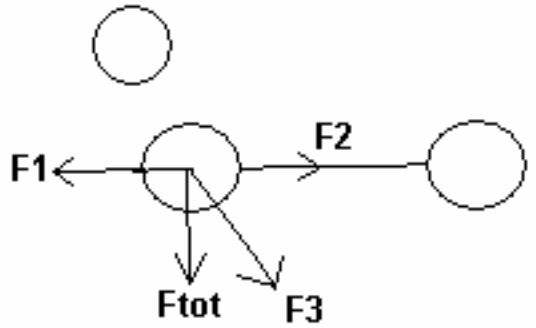


Figur 26 - eksempel på graf før og etter algoritme. Grafent til høyre er visuelt bedre.

En spennende ide, som er omtalt i [44], er å putte alle disse kriteriene inn i en fitness-funksjon og la en genetisk algoritme ta seg av problemet å søke etter den grafen som best tilfredsstiller kriteriene. Da slipper man å klekke ut avanserte algoritmer for å finne en god løsning. Problemet er selvfølgelig å finne en fitness-funksjon som inkorporerer alle ideene man har for hva som er en pen graf.

Spring-force algoritme for graf-utlegg

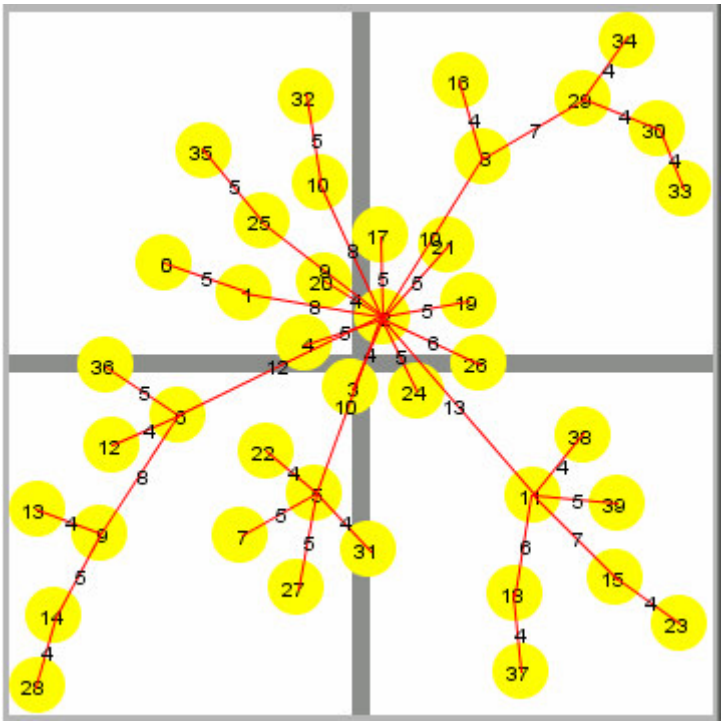
En løsning jeg falt veldig for var spring-force algoritmen (se [72] for en fin demonstrasjon). Den er enkel å implementere og resulterer i flotte grafer. Problemet er at den i sin enkleste form er treg[33], så den takler ikke for mange noder og kanter.



Figur 27 - illustrasjon på hvordan spring force algoritmen fungerer. Krefter på kun en av nodene er tegnet inn.

[33] Quigley og Eades (se [71]) har en raskere implementasjon som på forhånd deler inn grafen i klustre som fungerer som digre noder (en slags abstraksjon). Selv tenkte jeg i de baner at en litt "intelligent" preprosessering (f.eks. plassere urelaterte noder lengre vekk fra hverandre enn relaterte noder) av grafen gjør at spring force algoritmen konvergerer raskere.

Figur 27 viser hvordan spring force algoritmen fungerer. Noder betraktes som like poler som dermed frastøter hverandre og kanter betraktes som fjærer som drar tilkoblede noder sammen. F2 er fjærkraft, F1 og F3 er "nodekrefter". Hele poenget er å regne ut resultantkrefter for alle noder, flytte dem ett steg og gjenta helt til systemets totale energi (etter en eller annen målestokk, f.eks. summen av kreftene) i er minimal (her kan prinsipper som simulated annealing eller lignenede brukes for å unngå lokale optima). Når systemet etter hvert stabiliserer seg ser grafen ofte veldig fin ut (se figur 28).



Figur 28 - eksempel på resultat av spring-force algoritmen for graf-utlegg (hentet fra [72]).

Metodikk i systemutviklingsprosessen

Det var aldri aktuelt å følge noe tungvekts metodeverk, f.eks. RUP (se [13]). Det har vært kun en utvikler, dvs. at alle prosesser er i ett hode, og siden metodeverk for utviklingsprosesser vanligvis søker å få mange hoder til å fungere som ett, faller litt av vitsen bort. Men jeg prøvd å være systematisk selv om jeg har hatt mange skisser/design/krav/ideer i hodet før de har kommet ned på papiret. Den prosessen jeg kjenner meg best igjen i er Scrum (se [11]) og TDD (Test Driven Development, se [18]).

Scrum fordi man med det metodeverket har en todo-liste med prioriterte punkter hvorfra man programmerer de viktigste først og jobber i *sprinter* (pakker av punkter på listen). Jeg skriver hele tiden ned ting som må gjøres og prioriterer dem, deretter plasserer jeg de ut over 2 eller 3 dager.

TDD fordi jeg skriver tester for de mest sentrale funksjonene. Dermed oppnår TDD sitt store salgsargument: kode som er lettere å se hvor feiler når noe nytt puttes inn og kode hvor spesifikasjonen[34] av en metode skrives før selve metoden.

[34] En test er på en måte en spesifikasjon "den er riktig når dette fungerer".

TucGui

I denne delen av oppgaven beskrives teknologi-/prosess-valg for TucGui, samt selve TucGui, i form av klassekart, usecase, kravspesifikasjon etc.

Først legges grunnlag for TucGui i form av problembeskrivelse, deretter hvilke krav vi stiller til systemet og hvordan funksjonalitet systemet dermed må tilby (usecase). Så beskrives hvordan arkitekturen for systemet ser ut og til slutt hvilke teknologier som ble brukt for å få alt dette til.

Problembeskrivelse

Problembeskrivelsen er identisk med diplomens oppgavetekst, men repeteres her for flytens skyld.

Ethvert ekspertsystem er avhengig av en god kunnskapsbase i bunn. Kunnskapsbasen i GeneTUC har blitt populert inkrementelt av automatiske metoder som har blitt monitorert av dataeksperter. En flaskehals har vært at når noe har vært uklart har domene-eksperter måtte bli kontaktet via mail for avklaring, hvorpå kunnskapsbasen har blitt oppdatert av dataeksepertene.

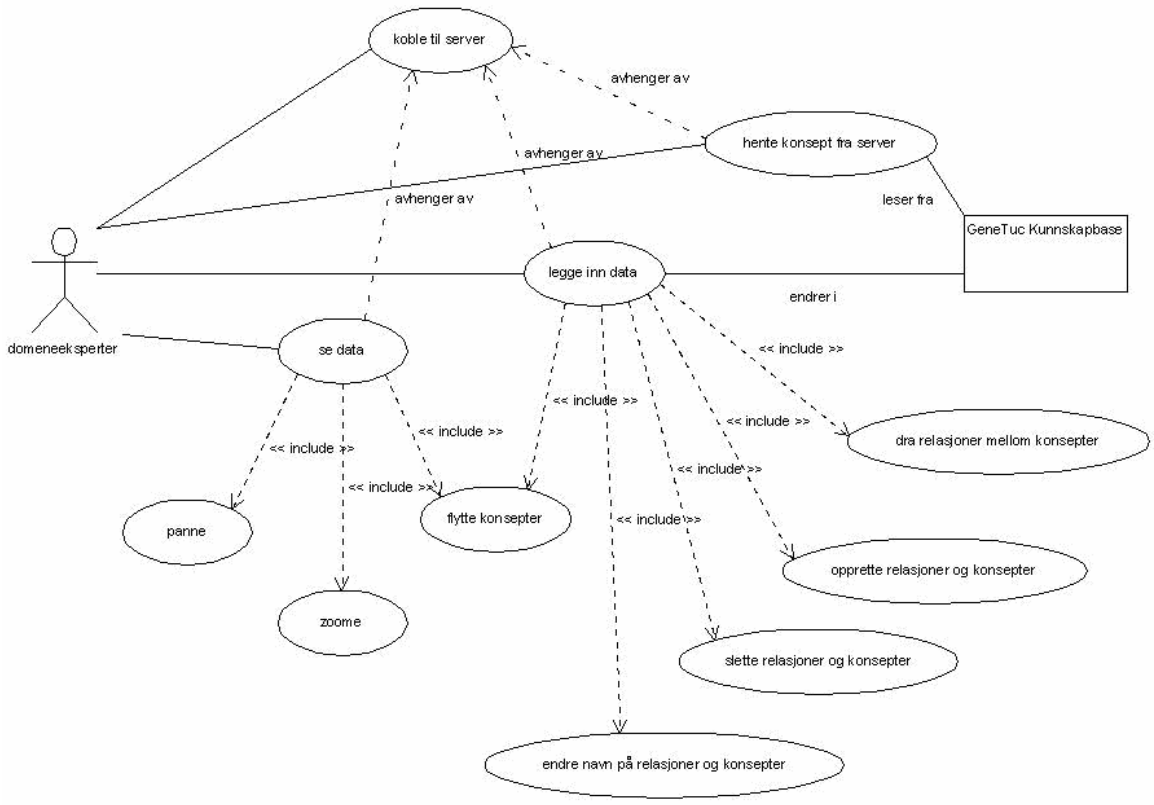
Det er et ønske å gi domene-eksperter enklere tilgang til kunnskapsbasen slik at data i kunnskapsbasen lettere kan valideres, oppdateres og legges til. Et annet viktig poeng er at GeneTUC per i dag ikke har noen mulighet for å visualisere dets data, noe som lenge har vært ønsket, men ikke prioritert.

Visualisering og mulighet for editering av GeneTUC sin kunnskapsdatabase vil blandt annet gjøre det lettere å

- holde data konsistent
- legge til nye data
- se relasjoner som ikke klart kommer fram i tekstfiler

Bruker scenario (usecase)

I figur 29 vises funksjonalitet som skal tilbys bruker.



Figur 29 - usecase for TucGui.

Kravspesifikasjon

Det skal laget et grafisk brukergrensesnitt for GeneTUC som skal være slik at brukerne lettere kan legge til, endre og slette data fra GeneTUC sitt semantiske nett. Under er kravene formalisert.

1. Brukervennlighet
    a. TucGui skal være enkelt og intuitivt å bruke.
    b. Det skal være lett å navigere og finne frem.
2. Kompletthet
    a. TucGui skal eksponere GeneTUC sin kunnskapsbase, nærmere bestemt klasse-nettet, transitive og intransitive verb templater, adjektiviske templater og verb komplementer.
3. Utrulling av TucGui skal kreve minst mulig endring i GeneTUC. Man skal kunne trekke vekk TucGui og fortsette med GeneTUC som om TucGui aldri var der.
4. Det skal benyttes teknologier som er veldokumenterte, gjennomprøvde og som er til minst mulig bry for sluttbruker.
5. TucGui skal være utvidbar med hensyn på:
    a. å plugge inn andre datakilder.

b. å legge til nye grafiske symboler.

Hvordan krav blir realisert:

Tabell 2 - realisering av krav

Krav	Realisering
1a, b, 2	Konsepter vises som runde symboler og relasjoner som piler. Dette medfører god oversikt. Det er mulighet for zoom/pan og å flytte på konsepter. Man velger hvilken modus man er i (konsept, relasjon eller slett). Man lager konsepter ved klikke i åpent område, flytte ved å dra-klikke på konsepter. Man lager relasjoner ved å klikke og dra fra kilde- til mål-konsept. Man kan endre navn på konsepter eller relasjoner ved å klikke på tekst, dialogboks popper opp.
4	Det blir brukt klient/server-arkitektur. Server jobber direkte på filer (sørger for at klient alltid ser en refleksjon av filene). Filer låses ikke (men holdes under oppsyn for endring og lastes inn på nytt dersom ekstern prosess endrer filene), som betyr at de kan endres utenom systemet (f.eks. direkte i en tekst-editor).
5	Se teknologivalg.

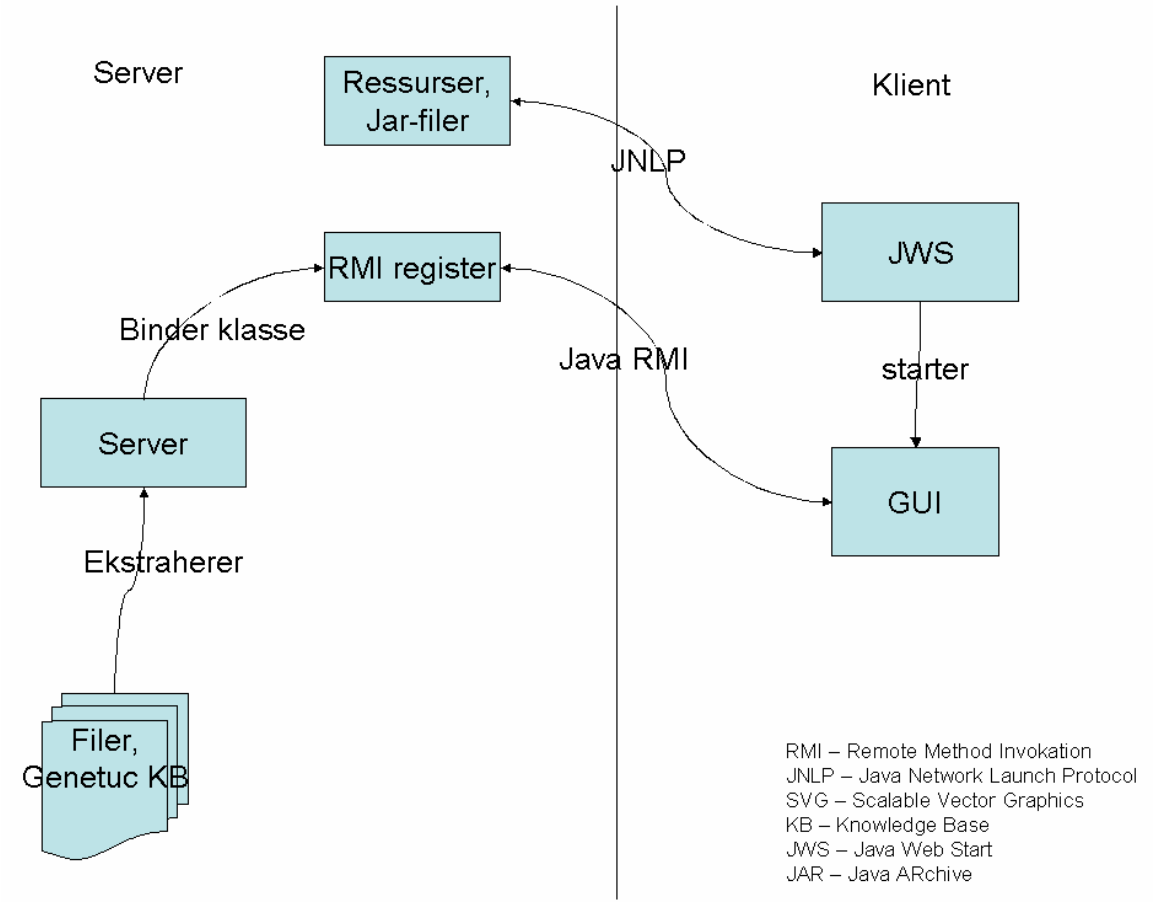
Design

Siden bruker ofte sitter et annet sted enn der kunnskapsbasen er lagret benyttes klient/tjener arkitektur. Dermed kan eksperter sitte forskjellige steder og endre på kunnskapsbasen, og veien videre til et system med funksjonalitet for samarbeid er kortere.

Serveren har datakilder som eksponeres grafisk til bruker. Datakildene kan være på hvilket som helst format men må kunne representeres som konsepter/entiteter med relasjoner mellom seg. Det er kun planlagt med GeneTUC sine filer (dvs. ikke intern TQL som ikke nødvendigvis er identisk med innholdet i filene) som datakilder, men det er ingen ting i veien for å utvide til å ha f.eks. Gene Ontology eller en database som datakilde.

Det er et enkelt meldings-system mellom server og klient (enveis fra server) slik at server kan informere om tilstand, f.eks. at et konsept man vil se ikke eksisterer eller at algoritmen for graf-utlegg bruker for lang tid.

Arkitektur



Figur 30 - overordnet arkitekur for TucGui.

For mer om de valgte teknologiene, se kapittel "Teknologier".

Under er overordnet interaksjon mellom server og klient beskrevet. På serversiden startes rmi-registeret (eget program) slik at klasser kan gjøres tilgjengelig over nettet. Serveren startes og gjør følgende:

1. Leser konfigureringsfil..
2. Tolker filer med kunnskap.
3. Binder instans av klassen som skal være tilgjengelig til rmi-registeret.
4. Venter på tilkoblinger fra klienter.

Når noen ønsker å bruke TucGui skjer dette:

1. JWS startes ved at bruker f.eks.klikker på link til JNLP fil som må ligge fritt tilgjengelig.

2. Alle data JWS trenger for å starte applikasjonen ligger i denne filen.
3. Eventuelle jar-filer etc. som trengs lastes ned (det kan være oppgradering av JVM[35] eller jar filer applikasjonen bruker).
4. Applikasjonen er startet.
5. Bruker åpner tilkobling til server. Denne kommunikasjonen skjer via RMI.
6. Bruker oppdaterer/legger til kunnskap via grensesnittet. Endringer skjer umiddelbart i de filer.
7. Klient avslutter sessjon.

Strukturen i svg-dokumentet som sendes fra server til klient er som følger (se [23] for forklaring til alle svg-dokuments elementer):

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<?xml-stylesheet href="http://www.idi.ntnu.no/~fredrhag/genetuc_gui/style.css" type="text/css"?>
<svg  xmlns:xml="http://www.w3.org/XML/1998/namespace"
xmlns:dc="http://purl.org/dc/elements/1.1/"
xmlns:cc="http://web.resource.org/cc/"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns="http://www.w3.org/2000/svg"
xmlns:svg="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink"
height="500"   width="500" viewBox="0 0 500 500"
overflow="visible"
xmlns:GenGui="http://www.idi.ntnu.no/GeneTuc/">
<defs>
<marker id="triangel" viewBox="0 0 10 10"
refX="35" refY="5" markerUnits="strokeWidth"
markerWidth="10" markerHeight="10" orient="auto">
<path d="M 0 0 L 10 5 L 0 10 z" />
</marker>
</defs>
  <g id="alt" >
    <rect  id="bg" x="-10000" y="-10000" width="20000" height="20000"/>

    <g id="relations">
      <g>
        <path d="M 222 145 L 313 474" marker-end="url(#triangel)"
        GenGui:from="CREB_binding_protein" GenGui:to="protein"/>
        <text x="267" y="309">isa</text>
      </g>
    </g>

    <g id="concepts">
      <g id="CREB_binding_protein" transform="translate(222,145)">
        <ellipse rx="20" ry="20" cx="0" cy="0"/>
        <text transform="translate(-10,4)">CREB_binding_protein</text>
      </g>
      <g id="protein" transform="translate(313,474)">
        <ellipse rx="20" ry="20" cx="0" cy="0"/>
        <text transform="translate(-10,4)">protein</text>
      </g>
    </g>

  </g>
</svg>
```

Figur 31 - forklaring til struktur på svg-dokument.

1. Plassholder for alle relasjoner.
2. Plassholder for en enkelt relasjon.
3. Attributt som angir hvor relasjonen går fra (eget navnerom for å unngå konflikt).

[35] Java Virtual Machine.

4. Navn på relasjonen.
5. Attributt som angir hvor relasjonen går til (eget navnerom for å unngå konflikt).
6. Plassholder for konsepter.
7. Plassholder for ett enkelt konsept.
8. Et konsepts interne id (trenger ikke være det samme som konseptets navn, som er lagret i 9). Brukes bare av relasjoner for å knytte konsepter sammen.
9. Navn på konsept.

Det er viktig at strukturen på dokumentet som sendes mellom server og klient er slik som beskrevet her. Se kapittel "Utvidbarhet" for hvordan man kan utvide strukturen med nye symboler.

Klassediagram

Klassediagrammer er vedlagt i Appendiks 4 – Klassediagrammer. Diagrammene er generert etter at koden ble skrevet[36].

Utvidbarhet

Er det en ting jeg har erfart er det at et systems kravspesifikasjon sjeldent er konstant. Et system som dette utvides i hele dets livsløp, slik at det tilpasses og dekker behov som dukker opp. Derfor har utvidbarhet vært noe jeg har tenkt på gjennom hele utviklingsprosessen. Mer konkret hva som kan gjøres i kapittel "Videre arbeid".

Systemet er utvidbart på flere områder:

- Nye symboler i i visualiseringen (ved hjelp av SVG).
- Bruke andre datakilder enn Prolog-filene.

Hvis det skal legges til symboler i svg-strukturen må det legges til som siste barn-node til sin forelder (da interfererer den ikke med allerede eksisterende rutiner for å hente ut data fra strukturen).

Teknologier

I denne delen av oppgaven redegjør jeg for de teknologier jeg har valgt for å implementere TucGui.

[36]Dette er kommentert i Appendiks 4 – Klassediagrammer .

Java

Valg av utviklingsplattform falt på java av flere grunner:

- Java er et portabelt, lett å distribuere og lett å vedlikeholde (med f.eks. java web start eller som applet).
- I og med at det har vært språket vi har brukt igjennom studiet, er det plattformen jeg kan best[37].
- Jeg ønsket å bli enda bedre på java.
- ECMAscript og Adobe SVG plugin komboen viste seg å ikke fungere spesielt bra.

Man kan argumentere for at java er tregt, men med introduksjonen av Hotspot[38] virtual machine og diverse andre strategier, er java i visse tilfeller raskere enn f.eks. C++ (se [66]).

SVG

SVG[39] er en XML notasjon for vektor-grafikk spesifisert av W3C (se [23]) og er ment for å være en grafikk-standard for web[40], men brukes også utenom webben, f.eks. Inkscape (se [24]), et vektorbasert tegneprogram som bruker SVG som lagringsformat (se Figur 32 for egen-laget eksempel tegnet i Inkscape). Det finnes mange grafikk-programmer som implementerer SVG-standarden, f.eks. plugins til browsere, browsere selv[41], tegne-programmer, mobile viewer (se [26] og [27]). [25] lister mange implementasjoner. Det fine med SVG er at

- Det har massevis av muligheter (alle mulige primitiver, filtre, animasjon, interaksjon og skripting).
- Det er en åpen standard, i motsetning til f.eks. Flash.
- Det finnes åpen kilde implementasjoner.
- Det kan stilsettes med CSS[42], som gjør at man kan trekke ut alt som har med utseende å gjøre til en egen fil.

[37] Målet med diplomen var ikke å lære seg et nytt programmeringsspråk.

[38] Hotspots er deler av koden som ofte brukes. Denne koden blir optimalisert og kjører derfor fort.

[39] Scalable Vector Graphics.

[40] Noe ala Macromedia Flash, bortsett fra at SVG er en åpen standard.

[41] Opera 8 implementerer TinySVG (se [61]) og Firefox 1.1 har en (per april 2005) buggy innebygd SVG-støtte.

[42] Cascading Style Sheet, se [29]

Figur 32 - eksempel på svg laget i inkscape. markup koden som utgjør dette bildet finnes i.Appendiks 3 – XML markup for smiley.

Når en SVG xml fil tolkes av et program bygges et SVG-DOM[43] tre. Denne trestrukturen gir en intuitiv intern representasjon som man kan endre på (hekte på og ta vekk noder).

Batik

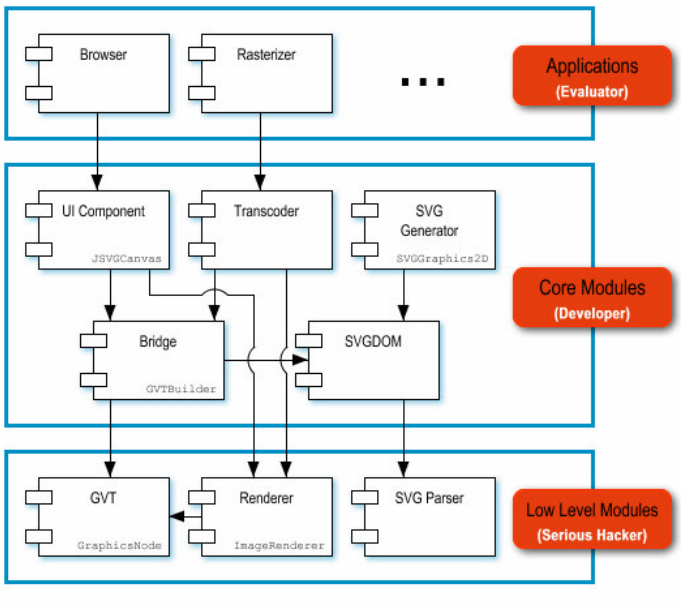
Batik (se [31]) er en rimelig moden java implementasjon av SVG-spesifikasjonen som er del av Apaches Jakarta del[44]. Siden utviklingsmiljø for TucGui er java og svg er grafikk-standarden ble Batik et naturlig valg. Den implementerer hele SVG 1.1 standarden (som var tilstrekkelig for TucGui sitt behov) og deler av 1.2 standarden (1.2, se [32], er heller ikke ennå helt spesifisert).

Batik sin arkitektur er lagdelt i forskjellige behov[45], fra overflatisk til hardcore (se figur 33).

[43] DOM: Document Object Model, se [30].

[44] Jakarta er en paraplybetegnelse på alle Apache sine prosjekter som utvikles for java.

[45] Jeg så for meg å holde på i "developer" planet, men som alltid er ting vanskeligere enn forutsett og jeg måtte inn i kildekoden og se.



Figur 33 - Batik sin lagdelte arkitektur (hentet fra [67]).

En viktig ting ved valg av teknologi er ikke bare at den er moden og uten de verste barnesykdommene. Den bør være veldokumentert og ha et aktivt miljø av brukere og utviklere. Batik's dokumentasjon var ok og det var et høyst levende miljø av utviklere som utveksler tanker via deres mailing-liste. Dette er gode indikasjoner på at teknologien er et bra valg.

ANT

ANT[46] (se [68]) kan sammenlignes med Build, et verktøy for å lette prosessen fra kildekode til distribuerbar kode. ANT kan automatisere en lang rekke oppgaver, f.eks. kompilere, lage jar, lage javadoc, kopiere filer. Oppgaver kan kjøres sekvensielt eller parallellt og kan avhenge av hverandre eller av om en variabel har en spesiell verdi. De fleste IDE-er[47] for java bruker ANT for build-prosessen, også ide-et jeg brukte (Netbeans). Det gjør at man kan endre build-prosessen slik man trenger. For eksempel trengte jeg å signere jar filer. Det finnes det ant task for, så det var en rimelig lett oppgave å inkludere signering i build-prosessen slik at jeg slapp gjøre det manuelt hver gang.

Med en så kompleks build-prosess er det interessant å kunne visualisere den. [33] er et lovende men umodent verktøy for visualisering og editering av ant-skript. [34] er eksempel på verktøy kun for visualisering.

[46] I motsetning til SVG er ANT en de facto standard som Jame Duncan Davidson kom opp med da han holdt på med Tomcat (både Tomcat og Ant er nå jakarta prosjekter).

[47] For eksempel Borland JBuilder, Netbeans og Eclipse.

Java Webstart

Java webstart (JWS), referanse implementasjon av java network launch protocol (se [35]), er en måte å distribuere programvare på. Den er laget slik at det skal være minst mulig mas for brukeren å komme i gang med programmet. I mange tilfeller holder det med et klikk på en link, så skjer resten av seg selv. Det som trengs er at klienten har java web start installert (kommer standard med java standard edition). All informasjon som trengs for å starte programmet ligger i en JNLP fil. Filer med "jnlp" etternavn er standard knyttet til JWS slik at når man klikker på en JNLP fil (lokalt eller via nett) er det JWS som håndterer filen.

Standard blir programmer kjørt i sandbox miljø (samme som for applets; ingen fil-tilgang, kun kommunisere med server hvor programmet ble hentet), men programmet kan be om ubegrenset tilgang. Da må blant annet jar filene som lastes ned være signert.

Dersom programmet oppdateres (på serversiden) vil klienten neste gang han starter programmet først laste ned det som er nytt. For et miljø hvor det er mange klienter vil det skape endel datatrafikk, men det gjør all patching av programvare. Fordelen med JWS er at oppdateringen så og si skjer transparent for sluttbruker.

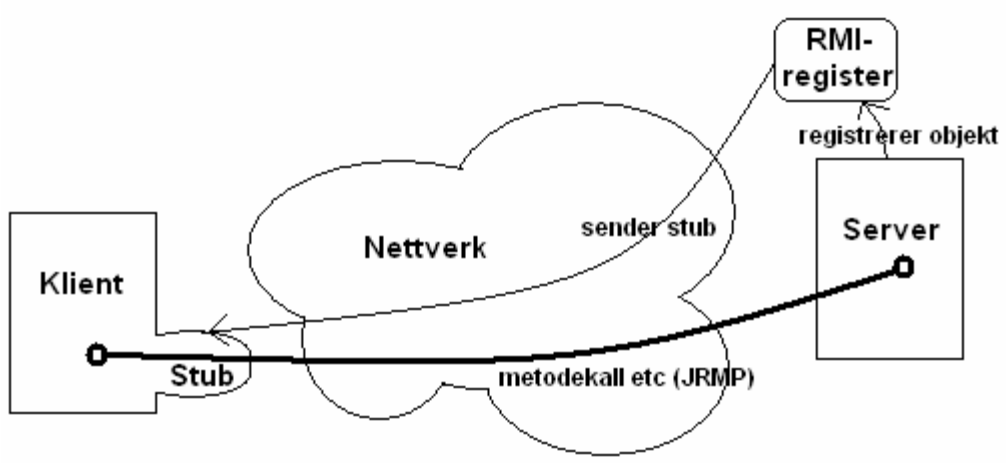
Java Remote Method Invocation (Java-RMI)

For at brukere av TucGui skal kunne sitte på andre maskiner enn serveren trengs en måte for klient og server å kommunisere på. Java RMI (se [2] for eksempler på applikasjoner og [3] for grundig beskrivelse av RMI) er en måte for JVM-er[48] på forskjellige datamaskiner å dele objekter. For klienten er det nesten like enkelt som å instansiere objekter lokalt.

Serveren instansierer det objektet den vil dele med omverdenen og registrerer det i rmi registeret (et eget program). Klienter som vil bruke dette objektet må vite hvor rmi-registeret kjører og spørre det etter objekter. Når en klient ber om et objekt via en oppslagsmekanisme (Java Naming and Directory Interface, JNDI, se [37]), får den tilbake en "stub" som skjuler all kommunikasjon med serveren. Stub-en videresender metodekall og gir tilbake retur-verdier (dette skjer via JRMP[49]). Se Figur 34 for oversikt.

[48] Java Virtual Machine.

[49] JRMP – Java Remote Method Protocol.



Figur 34 - enkel skisse på hvordan klient/server kommuniserer med rmi.

Alternativer til Java-RMI er for eksempel CORBA[50], DCOM[51] eller Web Services men siden Java er valgt utviklings-språk og Java-RMI er en enkel og veldokumentert teknologi falt valget på den.

VGJ

VGJ (Visualizing Graphs with Java, se [45]) er et åpen kilde, gratis java api for blant annet graf-utlegging. Det tilbyr en spring-algoritme (som jeg ønsket å bruke, se "Konsept-utlegging"), men også forskjellige andre graf-utleggingsalgoritmer.

Regulære Uttrykk (Regexp)

Siden GeneTUC sin kunnskapsbase er tekst-filer, er det brukt tegulære uttrykk for å hente ut og legge inn relasjoner og klasser. Alternativt kunne kunnskapsbasen blitt flyttet over til en database, som nok ville gjort vedlikehold enklere, men det har ikke vært prioritert. Dessuten er det viktig at TugGui ikke krever dyptgripende endringer i TUC. Så TucGui har måttet forholde seg til tekstfiler, og da er regulære uttrykk en glimrende måte å finne fram på.

Regulære uttrykk gjør det mulig å søke i tekst med komplekse mønstre, f.eks. ved bruk av ordklasser og jokertegn. Man kan tenke at et regulært uttrykk er kompaktformen av masse enkelt-strenger som man ønsker å finne. Hvis jeg f.eks. ønsker å finne alle engelske ord på tre bokstaver er det 26^3 mulige kombinasjoner å se etter. Å trykke inn alle disse kombinasjonene og se om de finnes tar latterlig lang tid, men med regulære uttrykk trenger man bare skrive "[^\w][\w]{3}[^\w]" hvor "\w" er klassen med alle bokstaver. ^ betyr "alle andre bokstaver enn de som er i denne klassen".

[50] Common Object Request Broker, kun en spesifikasjon. Finnes mange implementasjoner.

[51] Distributed Component Object Model, Microsoft standard.

Det finnes flere forskjellige algoritmer for løsing av regulære uttrykk, f.eks. NFA og DFA, (se [38]), og forskjellige "dialekter", dvs. hvilke ordklasser som er tilgjengelige og hvilke andre operatorer som kan brukes, f.eks. POSIX og Perl (se [38]).

Stort sett alle språk har biblioteker for regulære uttrykk, men mens f.eks. java krever at man må kalle metoder på vanlig måte har Perl[52] og Ruby[53] en veldig tett integrasjon med regulære uttrykk. Det er egne, enkle, konstruksjoner for å bruke regulære uttrykk.

Selv om det finnes flere forskjellige biblioteker for regulære uttrykk i java, f.eks [9], blir det i dette prosjektet brukt java standard implementasjon som er god nok. Hva den støtter av ordklasser kan leses på [39].

Implementasjon

I denne delen redegjør jeg for aspekter direkte tilknyttet implementasjonen av TucGui.

All koden finnes i Appendiks 6 – Kildekode.

Dokumentasjon

Tilgjengelig dokumentasjon er:

- Utrulling og brukerdokumentasjon (se kapittel "Appendiks 1 – Utrulling og brukerdokumentasjon").
- JavaDoc (dokumentasjon som beskriver koden, se kapittel "Appendiks 5 – JavaDoc").
- Kommentert kildekode (se kapittel " Appendiks 6 – Kildekode").
- UML klassediagrammer (se kapittel "Appendiks 4 – Klassediagrammer").

Løsning på navigasjonsproblem

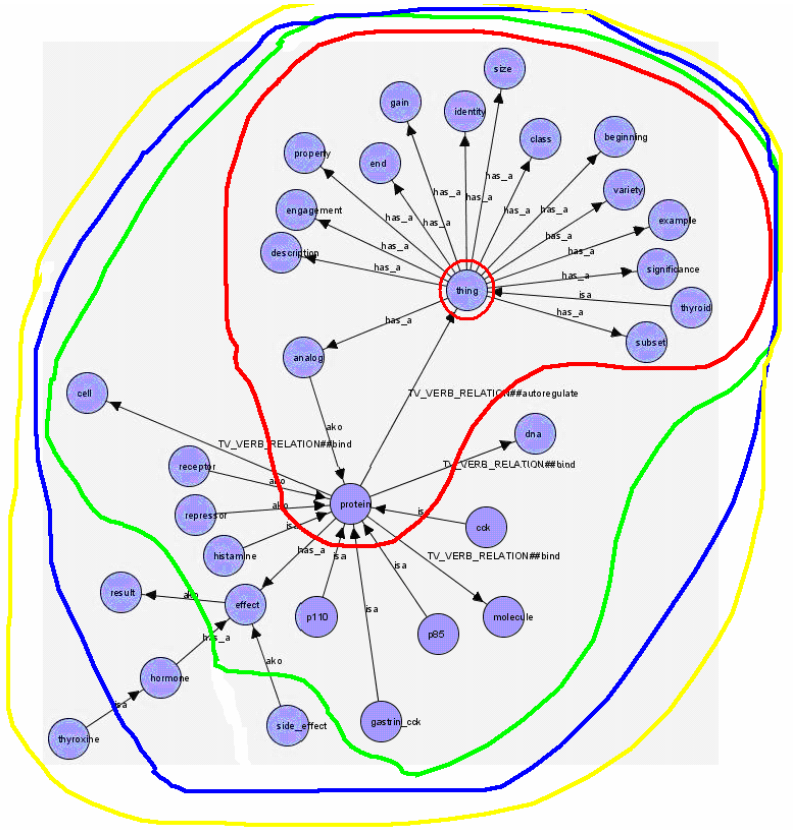
Som nevnt i kapittel "Navigere i store datamengder", er det ikke ønskelig å visualisere hele GeneTUC sin kunnskapsbase på en gang. Dette kapitlet beskriver løsningen som blir brukt i TucGui.

En "brute force" implementasjon av spring-force algoritmen, som vgj implementerer, tar i(n^2+k), hvor n er noder, k er kanter og i er iterasjoner. Dersom antall iterasjoner er mye

[52] Practical Extraction and Report Language.

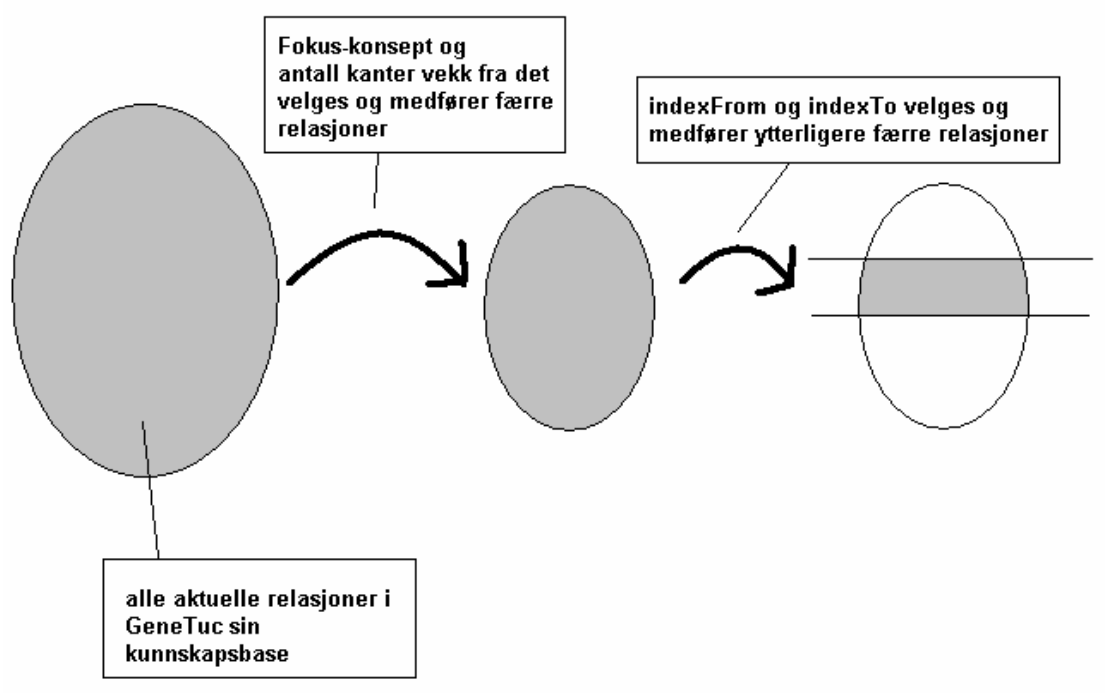
[53] Ruby (www.ruby-lang.org) er et yngre språk enn Perl, men bruker samme konstruksjon for regulære uttrykk. Mest sannsynlig for å også tiltrekke folk fra Perl-samfunnet.

mindre enn antall noder blir kjøretiden O(n^2). GeneTUC sin kunnskapsbase som semantisk nett inneholder mange noder (flere tusen), som betyr at det med VGJ ikke er overkommelig å legge ut hele grafen i rimelig tid. Dessuten er det ikke interessant å se hele det semantiske nettet på en gang. Man er stort sett interessert i en del av det. Derfor løste jeg problemet med for mange noder ved at man velger en fokus-node og hvor mange kanter vekk fra denne noden man ønsker (”maks avstand”). Figur 35 viser hvilke noder som blir inkludert når fokusnode er ”thing” (markert med rød ring) for maks avstand = 1, 2, 3 og 4, markert med henholdsvis rød, grønn, blå og gul strek.



Figur 35 - illustrasjon av maks avstand fra fokusnode.

Men det er fortsatt problem med enkelte konsepter, f.eks. ”thing”, som bare en kant vekk medfører veldig mange noder (det er mange noder som er direkte koblet til ”thing”). Løsningen på det er indexFrom og indexTo som bestemmer hvilket intervall av aktuelle relasjoner som skal hentes. Dette er en filtreringsprosess (se figur 36) som starter med alle relasjoner som eksisterer i TucGui sine kilder som deretter kuttes ned pga fokus-konsept og antall kanter man er interessert i vekk fra fokus-konsept, den mengden man da sitter igjen med (en liste med relasjoner) kuttes ytterligere ned vha. indexFrom og indexTo.



Figur 36 - filtreringsprosess fra alle relasjoner til et spesifikt interessant sett med relasjoner.

Dersom antall noder ut fra siste steg i filtreringsprosessen fortsatt er så stort at spring-force algoritmen bruker lang tid, blir den avbrutt etter 10 sekunder og bruker får beskjed om at algoritmen ble stanset. Siden spring-force algoritmen er en iterativ prosess (søk mot minste totale energi, se kapittel "Spring-force algoritme for graf-utlegg") kan det godt hende at grafen ser ok ut selv om den ble stanset, derfor bruker serveren det algoritmen ble ferdig med når den genererer SVG som sendes til klienten (det algoritmen ble ferdig med er helt sikkert bedre enn at alle konseptene ligger tilfeldig plassert, som de gjør i utgangspunktet).

Brukertest

Bortsett fra initielle møter for å presentere konseptet og ideene jeg hadde, har jeg hatt to møter med brukere på MTFS[54], en gang i april og en gang i mai. I april demonstrerte jeg den funksjonaliteten som så langt var implementert. Entusiasmen var stor, og jeg fikk mye nyttig tilbakemelding hvorav noe ble inkorporert i systemet og noe ble plassert i kapittel "Videre arbeid".

Da jeg var der i april lot jeg domene-ekspert selv bruke systemet. Jeg hadde på forhånd lagt inn endel data som ikke var sjekket av eksperter. Disse dataene gikk domene-ekspert over ved å slette relasjoner og klasser som var feil og endre navn der det var nødvendig. Programmet fungerte tilfredsstillende.

[54] Medisinsk Teknisk Forsknings Senter

Videre arbeid

Det har tatt mye tid og energi å få opp det TucGui er i dag, men på flere områder er det fordet rom for utvidelser og forbedringer. I dette kapitlet ser jeg på hva som kan gjøres videre med TucGui.

Ytelse

Det er et problem når man ber om for mange konsepter og relasjoner. Flaskehalsen ligger i Batik (å bygge GVT treet, som er en Batik-intern representasjon av SVG-dokumentet) og i spring-force algoritmen. Som nevnt før er det uansett ikke interessant å se hele kunnskapsbasen på en gang. Det holder med et utsnitt. Dermed ligger utfordringen i å kunne gjøre det utsnittet på en intelligent måte. Slik det er i dag, med fokusnode, maks-distanse og indeks fra/til, er problemet tilfredsstillende løst, men som nevnt i kapittel "Enorme datamengder" er det også andre, måter å løse dette på.

Oppgradert visualisering

Slik visualiseringen er i dag er det kun relasjoner og konsepter med deres navn som vises. Under lister jeg opp mulige visuelle forbedringer av TucGui.

- Boks med metadata – Dersom man hviler musepekeren over en relasjon eller et konsept kunne det kommet frem en boks med metadata, for eksempel konfidens-vekt (hvor trygg man er på den kunnskapen), endringslogg, kommentar og hvem som opprettet entiteten (se kapittel "Metadata"). Noe metadata måtte kunne editeres (direkte i boksen eller ved at en ny dialogboks kom opp) og noe ikke (f.eks. "sist endret av" eller "opprettet av").
- Kommentarer til relasjoner – mange av GeneTUC sine relasjoner har kommentarer i Prolog-filene. Disse kommentarene kan vises og editeres i metadataboksen.
- Gruppering av relasjoner. For bedre oversikt (for eksempel viser figur 42 et ganske overfylt skjermbilde) kunne det vært en valgmulighet at alle relasjoner av en viss type, f.eks. ako, inn til et konsept ble gruppert (dette kunne også skjedd automatisk dersom antall relasjoner oversteg en gitt terskelverdi) og vist som en tykk strek for å illustrere at den skjuler mange relasjoner. Dermed kan lettere observere andre relasjoner, f.eks. has_a relasjoner. En Gruppe med relasjoner kunne vært annotert med antall relasjoner den inneholder.
- Relasjonslinjer som bøyer av for å unngå å treffe konsepter og andre relasjonslinjer. Man kan tegne en relasjon som f.eks. en bezier kurve (SVG har støtte for dette) slik at streker ikke går under konsept-ellipser og i minst mulig grad krysser andre relasjonslinjer. Dette ville gitt en enda mer oversiktelig graf, men ville krevet lengre prosesseringstid for å regne ut hvordan relasjonslinjene måtte bli plassert ut.
- Fargekode de noder, f.eks. egen farge for noder som har "has_a" relasjoner inn til seg (dvs. at de er egenskaper ved en klasse). Et annet scenarie er når en automatisert

metode har lagt til data, kan disse merkes med f.eks. rødfarge slik at når en ekspert går inn for å validere nye data er det lett å se hva som er nytt.

- Forskjellige relasjonslinjer – for å skille relasjonstyper (isa, ako, has_a, tv_templ etc.) kan man ha forskjellige linjer som formidler hvilken relasjonstype de er symbol for. For eksempel kan relasjonene ha forskjellig farge, utseende på pil, mønster på linjen eller tykkelse på linjen. Dermen skjuler man prefiks-notasjonen som brukes nå og får en enda tydeligere visualisering.

Funksjonalitet

Mulighet for å importere eksterne data. Et scenario er at en biolog har hentet ut data fra en database med protein-interaksjoner og har dataene som XML eller annet strukturert format som inneholder data som kan transformeres til binærerelasjoner mellom konsepter. Da ville det vært nyttig å kunne importere disse via TucGui til GeneTUC.

Bruksområde

I kapittel "Visualisere GeneTUC" skrev jeg om hvilke områder GeneTUC kan visualiseres, og selv om jeg der med skjermbilder[55] viste hvordan de forskjellige områdene kan visualiseres i TucGui, er implementasjonen rettet mot å editering og visualisering av klassehierarki sammen med verb- og adjektiv-relasjoner. Men det skal ikke mye til for at TucGui skal kunne visualisere alle omtalte områder; ved å implementere nye datakilder for TQL eller bevistre kan TucGui visualisere disse akkurat som simulert henholdsvis i kapittel "Visualisere TQL" og "Visualisere bevistre".

Konklusjon

Biologi som forskningsområde er stort og behovet for koordinering og samkjøring av forskningsmiljøer har kommet mer og mer i fokus (f.eks. ERA-NET). For å bringe forskningsfronten videre må vi samarbeide. Akkurat som at de forskjellige delene i hjernen samarbeider for å oppnå mer enn delene alene må mennesker samarbeide og trekke i samme retning for å oppnå mer enn individene greier hver for seg.

GeneTUC er et ekspertsystem som til nå har hatt en kunnskapsakkvisisjonsfase som har vært til hinder for en raskere utvikling. Det har vært få domene-eksperter som har hatt muligheten til å validere GeneTUC sin kunnskapsbase og prosessen har vært tregere enn nødvendig. Dette har TucGui tatt tak i ved å legge til rette for at domene-eksperter selv kan validere og øke kunnskapsbasen som ligger til grunn for GeneTUC.

TucGui er utviklet med modne teknologier og det er lagt vekt på at systemet skal være utvidbart og fleksibelt. Java i kombinasjon med SVG har funger meget bra. Slik systemet er nå

[55] Noen av skjermbildene er editert for å kunne godt hvise hvordan TucGui *kunne* vært brukt. For eksempel implementerer ikke TucGui relasjoner som ikke er rette streker.

er det brukbart, men som med alt er det rom for forbedring. SVG som plattform for grafikk gir mange spennende muligheter.

Tilbakemeldingene fra domene-eksperter har vært udelt positiv: å få et grafisk grensesnitt hvor man visuelt kan se kunnskapen som GeneTUC benytter er både nyttig og motiverende.

Motivasjon er energi-givende. Det gir retning og kraft. Jeg vil påstå motivasjon er den viktigste faktoren for å få noe gjort[56]. Et pent og brukervennlig visuelt grensesnitt motiverer forskere til å orke å bidra til at et kunnskapsbasen til et ekspertsystem (som de kanskje ikke direkte ser nytten av) holdes konsistent og stadig utvides. I framtiden vil vi kanskje ha en internasjonal kunnskapsbase som er brukbar for alle ekspertsystemer og som har et visuelt grensesnitt som er med på å motivere forskere til å bruke det så fort de oppdager ny kunnskap.

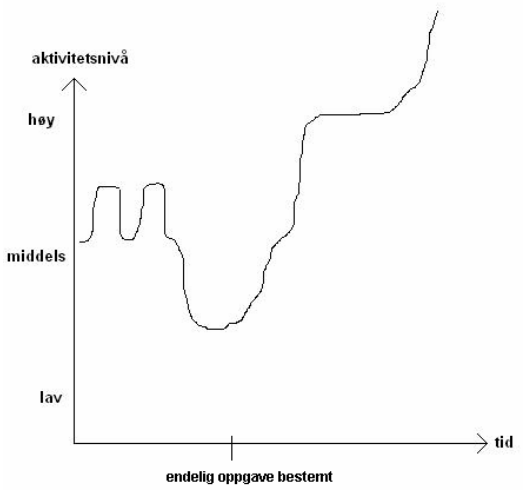
Den dagen forskeren samtidig med at han skriver artikkelen som beviser og forklarer et funn trykker denne nye kunnskapen inn i en kunnskapsbase har vi kommet et steg videre.

[56] Ta en dum og en lur person som begge skal løse en vanskelig oppgave. Hvis den lure personen totalt mangler motivasjon og den dumme er motivert vil den lure aldri komme med en løsning, mens den dumme muligens vil det.

Fritekst tanker

Prosessen å skrive diplom har vært en reise i stress og hard jobbing. Underveis har jeg gjort meg noen tanker som jeg legger til her helt på slutten av oppgaven.

Om prosessen



Figur 37 - aktivitetsgraf

Som nevnt i forordet har prosessen som dette siste semesteret har vært til tider artet seg som en slitsom og stressende en.

Grovt viser figur 37 hvordan aktivitetsnivået har vært dette siste semesteret. I løpet av de første ukene var jeg innom flere forskjellige oppgaver, hvorav ingen var klart definert. Av forskjellige grunner viste det seg at de tidligere oppgavene ikke var plausible å fortsette på. Til slutt endte jeg opp med en noe generell tittel ”visualisere GeneTuc”, som etter hvert endte i en fin oppgave som jeg er fornøyd med å ha jobbet med.

Etter at endelig oppgave ble valgt, både på grunn av vi var kommet et godt stykke inne i semesteret og at jeg likte oppgaven, var det full fart mot mål. Det var hardt arbeid fra dag en og det var stort sett ”diplom” som sto i hodet på meg. Jeg brukte så å si hele første halvpart på systemutvikling og andre halvpart på å skrive oppgaven.

Arbeidsbelastningen har vært konstant høy både fordi det har vært mye å gjøre og fordi jeg har har krevd mye av meg selv. Det har vært lærerikt å jobbe alene og måtte ta ansvar for at det faktisk blir gjort noe. Struktur og planlegging er noe jeg fant ut at var nødvendig for å greie å komme i mål. Derfor la jeg overordnede ukesplaner og detaljerte dagsplaner som mer eller mindre ble fulgt. Tett oppfølging er noe jeg innså verdien av, så at Tore Amble har vært romslig med møtetid har vært kjempefint.

Selv om vi nok noen ganger har snakket forbi hverandre er jeg veldig fornøyd med veiledningen. Det er tydelig at Tore vet hva han snakker om, så jeg har aldri vært i tvil om at dette skulle bli bra.

Om systemutviklingen

Som nevnt startet jeg med å utvikle et system for interaksjon mot GeneTuc sin kunnskapsbase. Selv om jeg egentlig hadde tenkt at krav skulle settes helt i starten, har systemet har blitt til i en iterativ prosess hvor vi etter hvert har funnet ut hvordan ting kan være. Jeg er fornøyd med måten utviklingen har gått på.

At jeg etter hvert måtte gå over på å faktisk skrive diplomen gjorde at jeg mer eller mindre stoppet kodingen. Så selv om systemet nå er brukbart, er det ting som ikke er helt på plass. For eksempel støttes ikke alle templat-relasjonene.

Uansett er denne teksten samt det implementerte systemet sammen med all dokumentasjon et godt startpunkt for videre utvikling og som inspirasjon for hvordan GeneTuc kan visualiseres.

Referanseliste

1 Raul Corazzon , "Ontology. A resource guide for philosophers", http://www.formalontology.it/

2 "An Overview of RMI Applications", http://java.sun.com/docs/books/tutorial/rmi/overview.html

3 "Understanding Java RMI Internals", http://www.developer.com/java/ent/article.php/3455311

4 Tore Amble, "The Understanding Computer".

5 Anders Andenæs, "GeneTUC", diplom-oppgave 2000

6 Sætre, Tveit, Ranang, Steigedal, Lægreid, Thommessen og Stunes. "gProt: Annotating Protein Interactions using Google and Gene Ontology"

7 J. Gennari, M. A. Musen, R. W. Fergerson, W. E. Grosso, M. Crubézy, H. Eriksson, N. F. Noy, S. W. Tu The Evolution of Protégé: An Environment for Knowledge-Based Systems Development. 2002.

8 Ashburner, M. et al., Gene ontology: tool for the unification of biology. The Gene Ontology Consortium, Nature Genet., 25:25–29, 2000.

9 Apache jakarta regexp, http://jakarta.apache.org/regexp/

10 Hjemmeside for Reactome, http://www.reactome.org

11 Hjemmeside for SCRUM, http://www.controlchaos.com/

12 Sætre, Tveit, Steigedal, Lægreid, "Semantic Annotation of Biomedical Litterature using Google"

13 Rational Unified Process, RUP, http://www-306.ibm.com/software/awdtools/rup/

14 Internet Relay Chat, IRC, http://www.irc.org/

15 Chincotta, Underwood, Ghani, Papadopoulou, Wresinski - Memory Span for Arabic Numerals and Digit Words: Evidence for a limited-capacity, Visuo-spatial Storage System

16 Leithead, Nejdl, Olmedill, Seamons, Winslett, Yu og Zhang, "How to Exploit Ontologies in Trust Negotiation", http://trust.mindswap.org/trustWorkshop/papers/12-f.pdf

17 Kokar, Wang, "An Example of Using Ontologies and Symbolic Information in Automatic Target Recognition.", http://www1.coe.neu.edu/~kokar/publications/SPIE2002.pdf

18 Test Driven Development, TDD, http://www.agiledata.org/essays/tdd.html

19 http://www.seas.gwu.edu/~simhaweb/software/jwordnet/

20 Eksempel på forskjellige graf-layouts - http://www.yworks.com/en/products_yfiles_practicalinfo_gallery.htm

21 Anders Lund Fredriksen, Fredrik Medby Hagen, "Elektroniske Emosjoner"

22 GO4J, Gene Ontologi api for Java, http://bioinformatics.org/project/?group_id=474

23 SVG 1.1 spesifikasjon, www.w3.org/TR/SVG/

24 Inkscape åpen kilde, vektor basert tegneprogram, www.inkscape.org

25 W3C vedlikeholdt liste over svg-implementasjoner, sist oppdatert 10. desember 2004, http://www.w3.org/Graphics/SVG/SVG-Implementations

26 SVG for mobile enheter, http://www.xml.com/pub/a/2004/08/18/sacre.html

27 SVG spesifikasjon for små enheter, http://www.w3.org/TR/SVGMobile/

28 Tanoue, Matoba, Yoshikawa, Uemura, "Graphical Representation of Gene Ontology in Scalable Vector Graphics", http://www.jsbi.org/journal/GIW01/GIW01P110.pdf

29 W3C's CSS spesifikasjon, http://www.w3.org/Style/CSS/

30 W3C's SVG-DOM spesifikasjon, http://www.w3.org/TR/SVG/svgdom.html

31 Apache Batik, http://xml.apache.org/batik/

32 SVG 1.2 spesifikasjon, http://www.w3.org/TR/SVG12/

33 Ghatica, https://ghatica.dev.java.net/

34 Antgraph, http://www.ericburke.com/downloads/antgraph/

35 JSR 56: Java Network Launching Protocol and API, http://jcp.org/en/jsr/detail?id=56

36 Java RMI spesifikasjon, http://java.sun.com/j2se/1.4.2/docs/guide/rmi/spec/rmiTOC.html

37 Java Naming and Directory Interface, http://java.sun.com/j2se/1.4.2/docs/guide/jndi/

38 Regular expression i wikipedia, http://en.wikipedia.org/wiki/Regular_expression

39 Regulære uttrykk i java, http://java.sun.com/j2se/1.5.0/docs/api/java/util/regex/Pattern.html

40 John F. Sowa, "Semantic networks", http://www.jfsowa.com/pubs/semnet.htm

41 Connectionism, Stanford Encyclopedia of Philosophy, http://plato.stanford.edu/entries/connectionism/

42 Dave Marshall, "Extending Semantic Nets", http://www.cs.cf.ac.uk/Dave/AI2/node62.html#SECTION00082300000000000000

43 Cranefield, Purvis, "UML as an Ontology Modelling Language", http://citeseer.ist.psu.edu/cranefield99uml.html

44 Bourquin, "Genetic Algorithm for Aesthetic Graph Layout", http://tecfa.unige.ch/perso/yvan/GeneticGraph/

45 "Drawing graphs with VGJ", http://www.eng.auburn.edu/department/cse/research/graph_drawing/graph_drawing.html

46 "A Brief History of Artificial Intelligence", http://www.comphist.org/computing_history/new_page_1.htm

47 Nick Milton, "Knowledge Acquisition", http://www.epistemics.co.uk/Notes/63-0-0.htm

48 Dublin Core Metadata Element Set. Versjon 1.1, http://dublincore.org/documents/dces/

49 Java properties, http://java.sun.com/j2se/1.5.0/docs/api/java/util/Properties.html

50 Alani, "TGVizTab: An Ontology Visualisation Extension for Protégé", http://eprints.ecs.soton.ac.uk/8326/01/Alani-VIKE-camera-ready.pdf

51 Nettside: Alani, TGVizTab A TouchGraph Visualization Tab for Protégé 2000, http://www.ecs.soton.ac.uk/~ha/TGVizTab/

52 Ann Parker, "Speeding the hunt, high speed DNA sequencing", http://www.llnl.gov/str/Balch.html

53 Jorge D. Cortese, "Array of options", http://www.the-scientist.com/2000/05/29/26/1

54 Fakta-ark om ERA-NET, http://kilden.forskningsradet.no/oversikter/eu/LCE_Fakta_era1.pdf

55 Ketan Babaria, "Using Treemaps to Visualize Gene Ontologies"

56 Mi Zhou, Yan Cui, "GeneInfoViz: Constructing and visualizing gene relation networks", http://www.bioinfo.de/isb/2004/04/0026/main.html

57 GeneInfoViz web-system, http://genenet.org/geneinfoviz/search.php

58 Jaap Kemps, "Visualizing WordNet Structure"

59 Montes, Gelbukh, Lopez, Yates, "Text Mining With Conceptual Graphs" http://ccc.inaoep.mx/~mmontesg/publicaciones/2001/ClusterCGs-nlpke01.pdf

60 Gilad Mishne, "Source Code Retrieval using Conceptual Graphs", http://staff.science.uva.nl/~gilad/pubs/MoL-2003-04.text.pdf

61 SVG Tiny spesifikasjon, http://www.w3.org/TR/SVGMobile/

62 Harry Delugach, CharGer, http://www.cs.uah.edu/~delugach/CharGer/

63 Conceptual Graphs, iso standard draft specification, http://www.jfsowa.com/cg/cgstand.htm

64 Brachman, Fikes, Levesque, "Krypton: A Functional Approach to Knowledge Representation."

65 Quillian, "Semantic Memory"

66 "The Java is Faster than C++ and C++ Sucks Unbiased Benchmark", http://kano.net/javabench/

67 Batik Architecture Overview, http://xml.apache.org/batik/architecture.html

68 Apache ANT prosjektets hjemmeside, http://ant.apache.org/

69 OMG spesifikasjon, Unified Modelling Language (UML) versjon 1.5, http://www.omg.org/technology/documents/formal/uml.htm

70 Shapiro, Stuart C. (2000), "SNePS: A Logic for Natural Language Understanding and Commonsense Reasoning" [Postscript], in Lucja M. Iwanska, & Stuart C. Shapiro (eds.), Natural Language Processing and Knowledge Representation: Language for Knowledge and Knowledge for Language (Menlo Park, CA/Cambridge, MA: AAAI Press/MIT Press): 175-195.

71 Quigley, Eades, "FADE: Graph drawing, clustering and visual Abstraction", http://www.it.usyd.edu.au/~aquigley/papers/aq-gd2000.pdf

72 Demonstrasjon av hvordan spring-force algoritmen fungerer, http://www.backspaces.net/Models/layout.html

73 Hjemmeside for Protégé, http://protege.stanford.edu/

74 Jack W. Reeves, "Code as Design: Three essays by Jack W. Reeves", http://www.developerdotstar.com/mag/articles/PDF/DevDotStar_Reeves_CodeAsDesign.pdf

Appendiks 1 – Utrulling og brukerdokumentasjon

I dette appendikset dokumenteres hvordan TucGui utrulles på serversiden samt hvordan brukere på klientsiden benytter systemet.

Klargjøring

Før TucGui er klart for bruk må det (dersom det trengs) bygges, dvs. at filer pakkes til jar (Java ARchive) som så signeres med en MD5 hash (må gjøres siden jar-filene lastes ned til klient via JWS og da må klient kunne vite at jar-filene kommer fra den man tror de kommer fra og at de ikke er endret i løpet av overføringen).

Bygging skjer ved å skrive "%ant_homedir%/ant.bat i den katalog hvor build.xml (build.xml inneholder all informasjon som ant trenger for å bygge systemet) ligger, dvs. TucGui sin hjemmekatalog. Resultatet av bygge-prosessen ligger i build katalogen i TucGui sin hjemmekatalog. Build-katalogen må gjøres tilgjengelig (kan gjerne flyttes) via en web-server slik at klienten kan nå start.jnlp (inneholder all informasjon som trengs for å starte applikasjonen via JWS) og resten av filene i build katalogen.

På serversiden

For å starte opp TucGui server gjøres følgende:

1. Start rmiregistret ved å kjøre rmiregistry.exe (er med i JRE[57] og ligger i %java_home%/bin).
2. Start serveren ved å stå i katalogen hvor GeneTUC_kb_grensesnitt pakken er og kjøre "java –cp . GeneTUC_kb_grensesnitt.server.Controller [config-fil]" hvor [config-fil] er fullstendig sti til konfigureringsfilen (se under) som skal brukes eller kjøre start.bat som ligger i rotkatalogen til TucGui.
3. Serveren skriver ut diverse beskjeder og ender opp med "server startet". Nå kan klienter koble til.

All konfigurering av serveren skjer vha en konfigureringsfil på tekstformat (se figur 38). Filen inneholder java properties (se [49]) som brukes av serveren. Kommentar-linjer starter med "#". Alle TucGui sine properties starter med "tucgui.". Det man setter opp i konfigureringsfilen er hvilke datakilder som skal brukes, hvor kodebasen for rmiregisteret er og hvor mye tilbakemelding om hva som skjer med serveren man vil ha til konsollet.

- tucgui.printlevel – et tall som angir hvor mye informasjon som serveren skal skrive til konsollet. Hvert nivå har et tall (info, debug, kritisk) som er en potens[58] av 2. For å få

[57] JRE, Java Runtime Environment

med de nivåene man vil adderer man sammen nivå-tallene (som tilsvarer bitvis ANDing). I eksemplet i Figur 38 skrives alt ut (1+2+4=7).

- tucgui.datasources – en streng bestående av de datakilder som skal brukes. Første datakilde er mål for nye relasjoner. Formatet på strengen er ”[datatype],[typespesifikk streng];”. Bruk doble back-slashes[59].
    - ”;” (semikolon) deler datakilder.
    - ”,” (komma) deler kildetype og typespesifikk streng.
    - ”[datatype]” indikerer hva slags kilde det er[60], f.eks. ”file”.
    - ”[typespesifikk streng]” er en streng den datatypen trenger for å laste filen, i tilfellet ”file” er det et filnavn.

```
java.rmi.server.codebase=file:/z:\\5kl\\diplom\\produsert_kode\\build\\classes\\

tucgui.printLevel = 7

#dersom man legger til nye relasjoner/konsepter er det den første datakilden som blir lagringsplass
tucgui.datasources=file,z:/5kl/diplom/genetuc data/nye.txt;file,Z:\\5kl\\diplom\\genetuc
data\\aux_classes.pl;file,Z:\\5kl\\diplom\\genetuc data\\facts.pl;
```

Figur 38 - eksempel på konfigureringsfil for TucGui.

Når serveren starter tolkes de datakilder som er spesifisert i konfigureringsfilen. Serveren skriver ut informasjon om hva den driver med (se figur 39) og ender med ”server startet”. Da er den klar til å ta imot klienter. I figur 39 er det filene endringer.txt, aux_classes.pl, facts.pl og semantic.pl som tolkes.

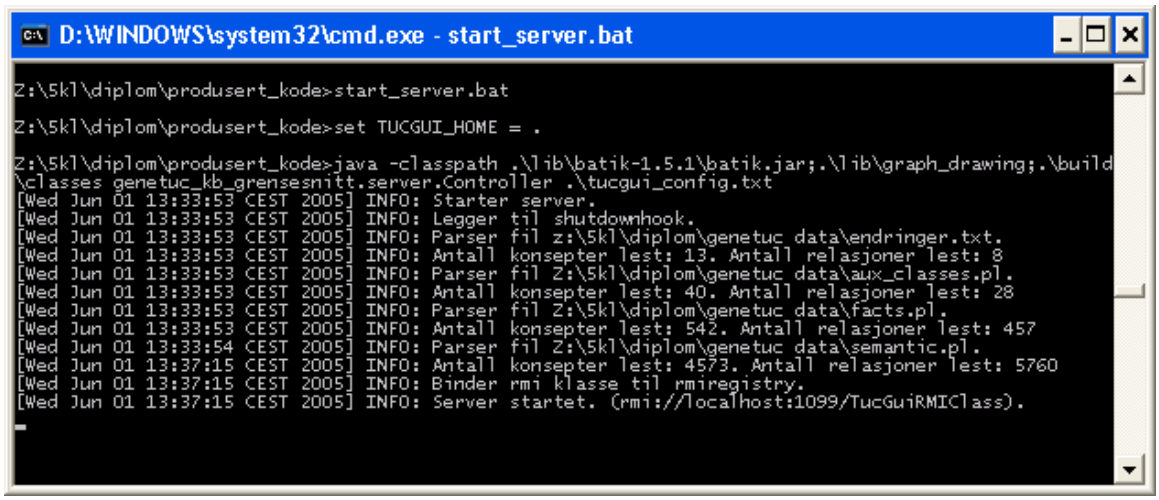
Når klienter kobler seg på og ber om data skriver serveren også ut informasjon om hvor mye data som sendes over.

---

[58] Grunnen til at det er potens av 2 (1,2,4,8,16 etc.) og ikke sekvensielt (1,2,3,4 etc) er at hvis det hadde vært sekvensielt hadde printlevel = 4 vært tvetydig (3+1=4 og 4 = 4) mens med 2-er potenser er det aldri tvetydig; et gitt tall kan kun representeres med ett sett av 2-er potenser.

[59] Tekststrengen blir først til String i java, og ”\” tolkes da som escape-tegn.

[60] Som internt i TucGui brukes for å finne en klasse som kan håndtere den kilden.

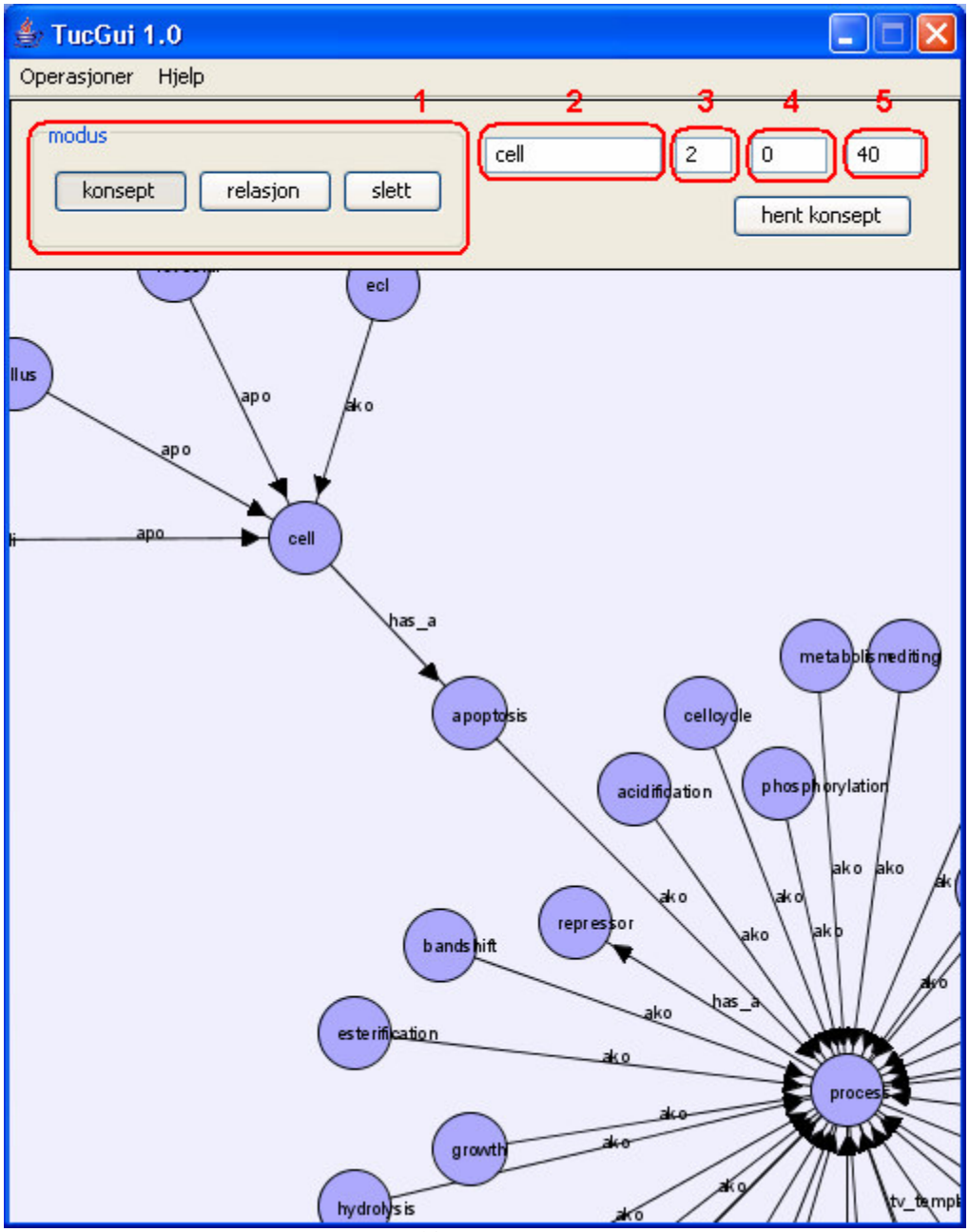


Figur 39 - figuren viser hva serveren skriver ut fra den starter til den er klar til å ta imot klienter.

På klientsiden

Grensesnittet på klientsiden startes ved at bruker åpner start.jnlp (som må være tilgjengelig på en eller annen måte, se kapittel "Klargjøring".

TucGui sitt hovedvindu er vist i figur 40.



Figur 40 - TucGui sitt hovedvindu.

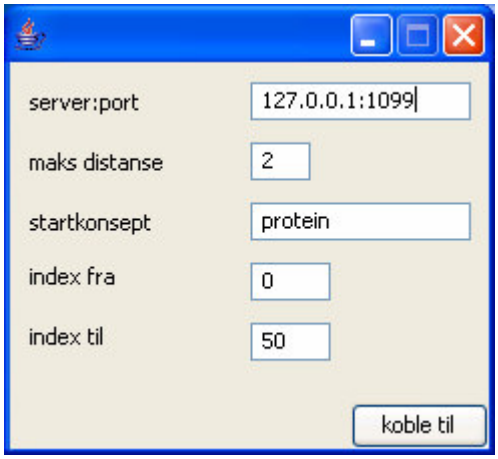
Vinduet består av følgende deler:

- Menylinjen (øverst)
  - Operasjoner – her kan man koble til server eller av slutte programmet. Når man velger å koble til server dukker skjermbildet vist i figur 41 opp. Her velger man server og portnummer, fokusnode, hvor mange steg vekk fra fokusnoden man vil ha med og start-/slutt-indeks (for forklaring til fokusnode, maks distanse og start-/slutt-indeks, se kapittel ”Løsning på navigasjonsproblem”).

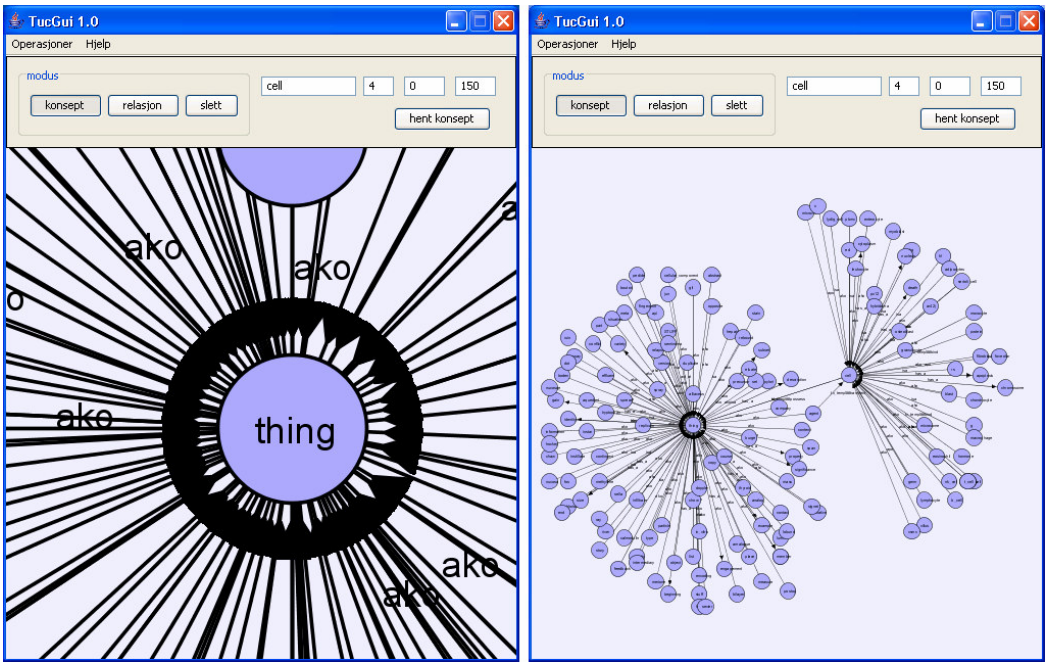
Dersom noe galt skjer under oppkobling får bruker beskjed om det via en dialogboks. Dersom alt går bra kommer fokuskonsept frem.

- Hjelp – se DOM (Document Object Model, den datastruktur som ligger til grunn for de som blir vist (konsepter og relasjoner)).

- Funksjonslinje – her er de funksjoner som brukes når man jobber mot en ontologi.
  - Modus (merket 1 i figur 40) – velger hvilken modus man er i (dvs. hva som skjer når man klikker i tegnebrettet under). Uavhengig av modus kan man gi konsepter og relasjoner nytt navn.
    - Konsept – i denne modus kan man lage nye konsepter (ved å klikke i et åpent område på tegnebrettet) eller flytte på dem (ved å klikke og dra et konsept).
    - Relasjon – i denne modus kan man lage nye relasjoner ved å trykke på et konsept og dra relasjonen (vises mens man drar) til et annet konsept.
    - Slett – i denne modus kan man slette konsepter (klikk på konsept) eller relasjon (klikk på relasjon, det er lettest å treffe om man trykker på pilen og ikke på streken).
  - Hente nytt konsept (merket 2,3,4 og 5 i figur 40) – Til høyre i funksjonslinjen finnes bokser for å hente nytt fokuskonsept. I 2 skriver man fokuskonsept, 3 er maks distanse, 4 er indeks fra og 5 er indeks til. Etter at man har skrevet inn ønskede verdier trykker man "hent konsept".
- Tegnebrettet – nederst i vinduet finner man området hvor visualiseringen skjer. Hva som skjer når man klikker bestemmes av modus (som beskrevet over). I konsept- og relasjon-modus kan man endre fokuskonsept ved å dobbelklikke et konsept (samme effekt som å bruke boksene merket 2,3,4,5 i figur 40 og trykke "hent konsept"), da blir maksdistanse, indeks fra og til hentet fra boksene merket 3,4,5 i Figur 40 og det konseptet man dobbelklikket på blir hentet. Uavhengig av modus kan man
  - panne (flytte det "vindu" man har mot grafen) ved å holde nede skift og venstre museknapp og flytte musepekeren rundt.
  - zoome (se figur 42) ved å holde nede skift og høyre museknapp og flytte musen opp og ned. Dette gjør det lett å navigere og få oversikt.



Figur 41 - dialogboks for å koble til server.



Figur 42 - zoome inn for å granske eller zoome ut for oversikt.

Appendiks 2 – Akronym ordliste

Akronym	Hele navnet
AI	Artificial Intelligence
AKO	A Kind Of
APO	A Part Of
ASV	Adobe SVG Viewer
CG	Conceptual Graph
CORBA	Common Object Request Broker Architecture
CSS	Cascading Style Sheets
DCOM	Distributed Component Object Model
DFA	Deterministic Finite Automaton
DNA	DeoxyriboNucleic Acid
ECMA	European Computer Manufacturers Association
GO	Gene Ontology
HAS_A	Has A
IDE	Integrated Development Environment
ISA	Is A
ISO	International Organization for Standardization
JNLP	Java Network Launch Protocol
JRMP	Java Remote Method Protocol
JVM	Java Virtual Machine
JWS	Java WebStart
JWS	Java WebStart
KA	Knowledge Acquisition
KI	Kunstig Intelligens
NFA	Nondeterministic Finite Automaton
OMG	Object Management Group
POSIX	Portable Operating System Interface for UniX
RMI	Remote Method Invocation
RUP	Rational Unified Process
SK	Skolem Konstant
SNePS	Semantic Network Processing System
SQL	Structured Query Language
SVG	Scalable Vector Graphics
SVG-DOM	SVG-Document Object Model
TDD	Test Driven Development
TQL	Tuc Query Language
TUC	The Understanding Computer
UML	Unified Modeling Language
VGJ	Visualizing Graphs in Java
W3C	World Wide Web Concortium
XMI	XML Metadata Interchange
XML	eXtensible Markup Language

Appendiks 3 – XML markup for smiley

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!-- Created with Inkscape (http://www.inkscape.org/) -->
<svg
   xmlns:svg="http://www.w3.org/2000/svg"
   xmlns="http://www.w3.org/2000/svg"
   xmlns:xlink="http://www.w3.org/1999/xlink"
   id="svg2"
   height="1052.3622"
   width="744.09448"
   y="0.00000000"
   x="0.00000000"
   version="1.0">
  <defs
     id="defs3">
    <linearGradient
       id="linearGradient2068">
      <stop
         id="stop2070"
         offset="0.00000000"
         style="stop-color:#ffff00;stop-opacity:1.0000000;" />
      <stop
         id="stop2072"
         offset="1.0000000"
         style="stop-color:#bebe00;stop-opacity:1.0000000;" />
    </linearGradient>
    <linearGradient
       gradientUnits="userSpaceOnUse"
       xlink:href="#linearGradient2068"
       id="linearGradient2066"
       y2="421.14679"
       x2="299.10825"
       y1="322.33316"
       x1="146.01674" />
  </defs>
  <g
     id="layer1">
    <path
       id="path1291"
       style="fill:url(#linearGradient2066);fill-opacity:1.0000000;fill-
rule:evenodd;stroke:#000000;stroke-width:3.0000000;stroke-
linejoin:miter;stroke-miterlimit:4.0000000;stroke-opacity:1.0000000"
       transform="matrix(1.000000,0.000000,0.000000,1.142744,45.71429,-
109.2745)"
       d="M   300.00000   375.21933   A   88.571426   88.571426   0   1   1
122.85715,375.21933 A 88.571426 88.571426 0 1 1  300.00000 375.21933 z"
/>
    <path
       id="path2076"
       style="fill:#ffffff;fill-opacity:1.0000000;fill-
rule:evenodd;stroke:#000000;stroke-width:3.0000000;stroke-
linecap:butt;stroke-linejoin:miter;stroke-miterlimit:4.0000000;stroke-
opacity:1.0000000"
       d="M 142.69326,324.82337 C 142.69326,324.82337 211.83509,419.70953
275.21510,404.02586     C     338.59511,388.34220     363.56300,346.78050
```

```
361.64240,310.70807     C     230.08085,387.55802     142.69326,324.82337
142.69326,324.82337 z " />
    <path
       id="path3596"
       style="fill:#ffffff;fill-opacity:1.0000000;fill-
rule:evenodd;stroke:#000000;stroke-width:4.0000000;stroke-
linejoin:miter;stroke-miterlimit:4.0000000;stroke-opacity:1.0000000"

transform="matrix(1.000000,0.000000,0.000000,0.678341,812.9442,94.62313)"
       d="M  -579.82755  286.66656  A  25.253813  26.263966  0  1  1    -
630.33517,286.66656 A 25.253813 26.263966 0 1 1  -579.82755 286.66656 z"
/>
    <path
       id="path3598"
       style="fill:#ffffff;fill-opacity:1.0000000;fill-
rule:evenodd;stroke:#000000;stroke-width:4.0000000;stroke-
linejoin:miter;stroke-miterlimit:4.0000000;stroke-opacity:1.0000000"

transform="matrix(1.000000,0.000000,0.000000,0.678341,908.9087,91.59253)"
       d="M  -579.82755  286.66656  A  25.253813  26.263966  0  1  1    -
630.33517,286.66656 A 25.253813 26.263966 0 1 1  -579.82755 286.66656 z"
/>
    <path
       id="path1295"
       style="fill:#000000;fill-opacity:1.0000000;fill-
rule:evenodd;stroke:#000000;stroke-linejoin:miter;stroke-
opacity:1.0000000"

transform="matrix(0.457967,0.000000,0.000000,0.507243,345.5840,190.8343)"
       d="M  -254.28571  183.79076  A  20.000000  20.000000  0  1  1    -
294.28571,183.79076 A 20.000000 20.000000 0 1 1  -254.28571 183.79076 z"
/>
    <path
       id="path3600"
       style="fill:#000000;fill-opacity:1.0000000;fill-
rule:evenodd;stroke:#000000;stroke-linejoin:miter;stroke-
opacity:1.0000000"

transform="matrix(0.457967,0.000000,0.000000,0.507243,425.4005,184.7423)"
       d="M  -254.28571  183.79076  A  20.000000  20.000000  0  1  1    -
294.28571,183.79076 A 20.000000 20.000000 0 1 1  -254.28571 183.79076 z"
/>
    <path
       id="path5122"
       style="fill:none;fill-opacity:0.75000000;fill-
rule:evenodd;stroke:#000000;stroke-width:10.224747;stroke-
linecap:round;stroke-linejoin:miter;stroke-miterlimit:4.0000000;stroke-
opacity:1.0000000"
       d="M 165.94156,225.21933 L 221.20130,180.93361" />
    <path
       id="path5882"
       style="fill:none;fill-opacity:0.75000000;fill-
rule:evenodd;stroke:#000000;stroke-width:8.6981554;stroke-
linecap:round;stroke-linejoin:miter;stroke-miterlimit:4.0000000;stroke-
opacity:1.0000000"
       d="M 333.75000,225.05015 L 267.67857,198.24564" />
  </g>
```

```
</svg>
```

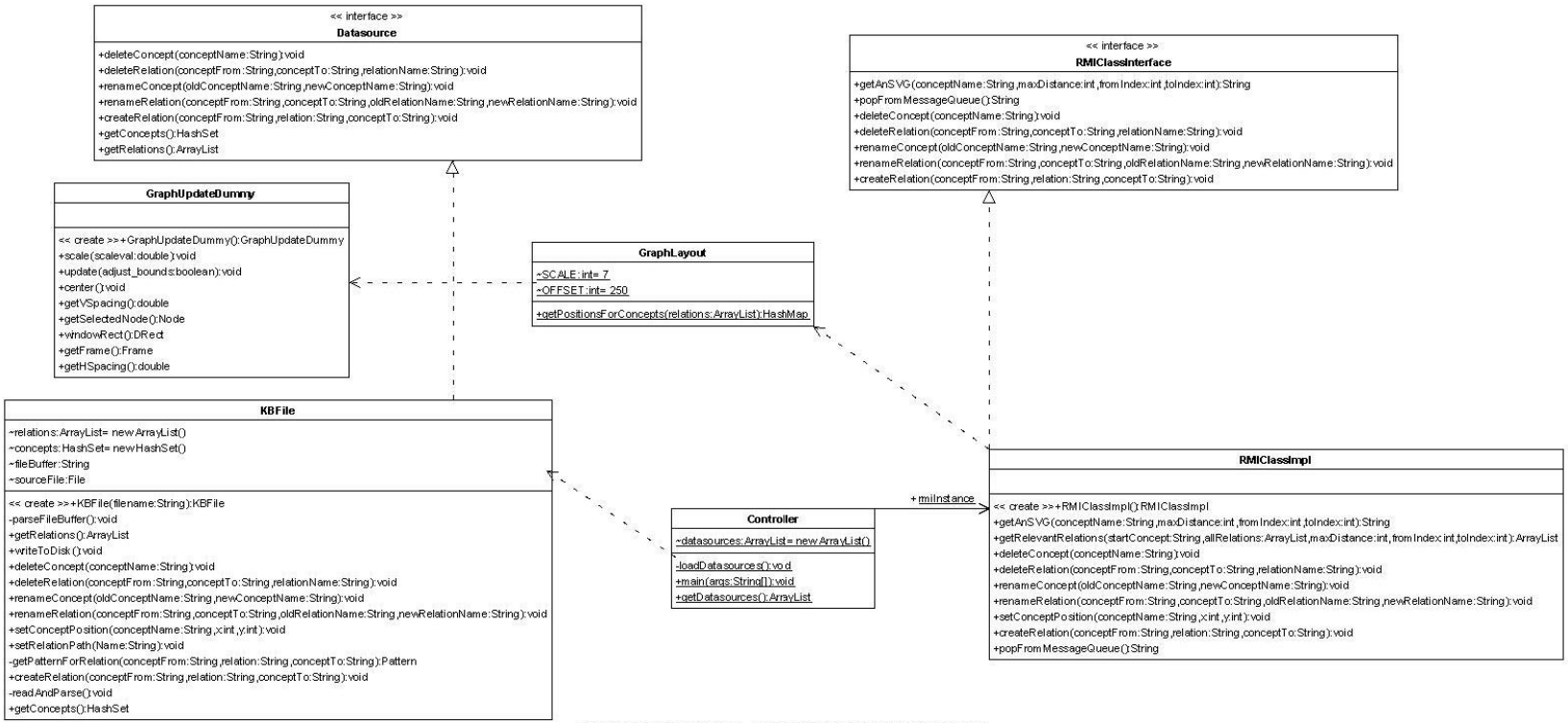
Appendiks 4 – Klassediagrammer

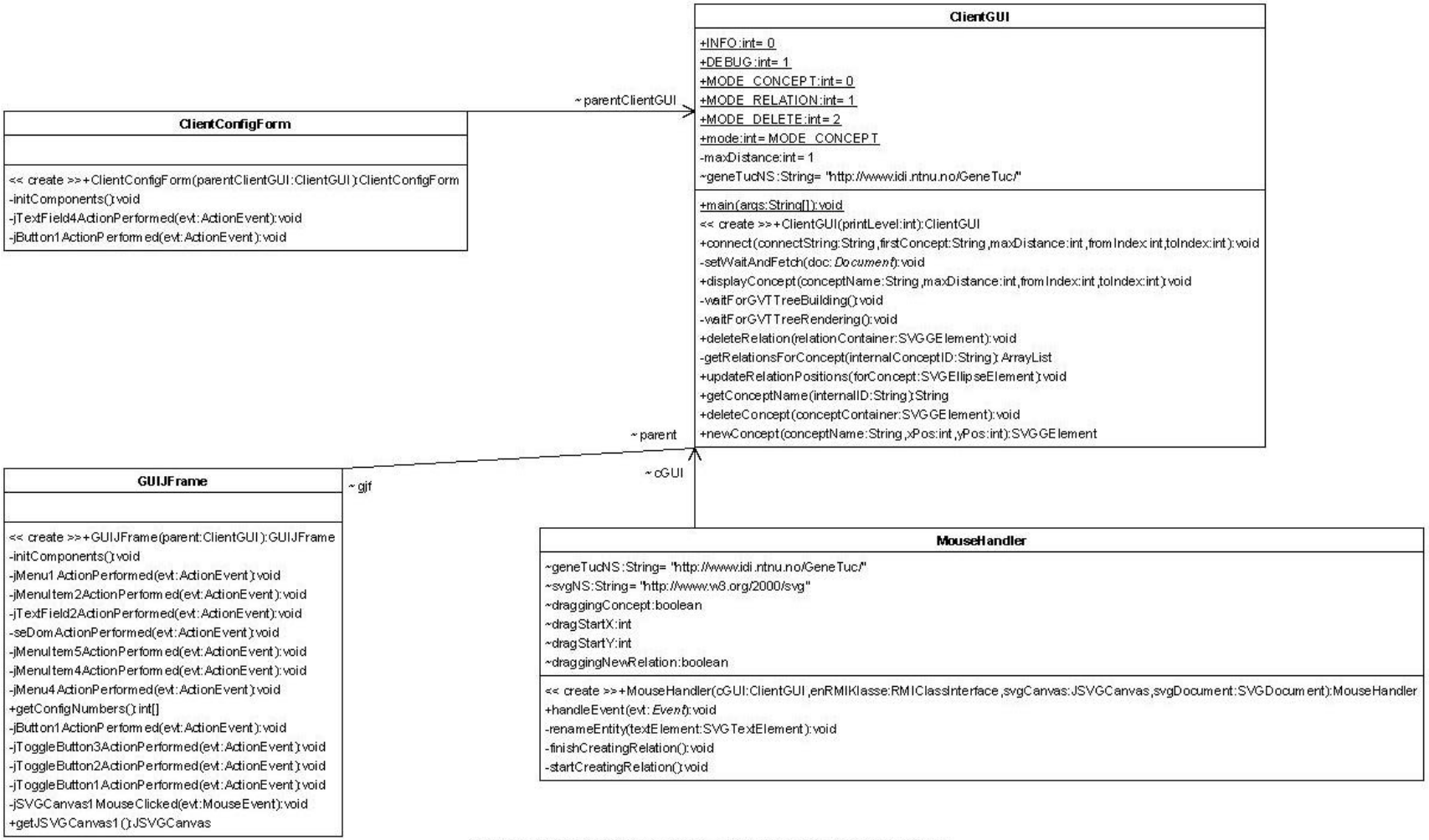
Den vanlige måten å bruke uml-diagrammer er som et verktøy for kommunikasjon i utviklingsprosessen, som grunnlag for generering av kode-skjeletter og som dokumentasjon.

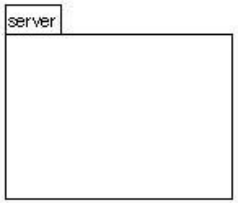
UML klassediagrammer for TucGui er her vedlagt som dokumentasjon.

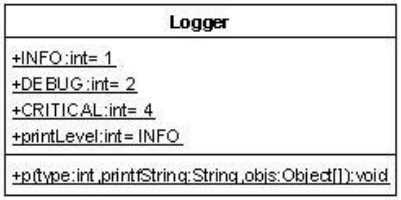
Siden jeg har jobbet alene med dette prosjektet og siden jeg i de prosjektene jeg har gjort alene tidligere har vært veldig fornøyd med penn og papir som verktøy for design, brukte jeg pen og papir og hodet i design-fasen[61]. Men klasse-diagrammer er ypperlig til dokumentasjons-foremål, så derfor har jeg laget disse uansett, dog i etterkant og semi-automatisk, siden det er mye lettere.

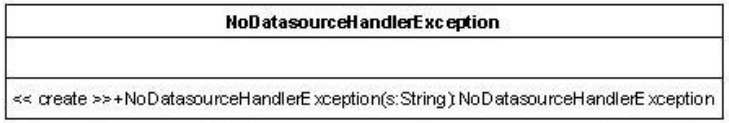
[61] [74] er for øvrig en interessant artikkel som argumenterer for at kode er den eneste skikkelige design i programvare-prosjekter.

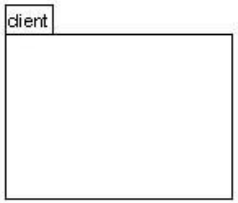


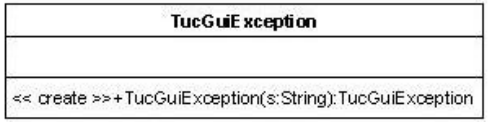


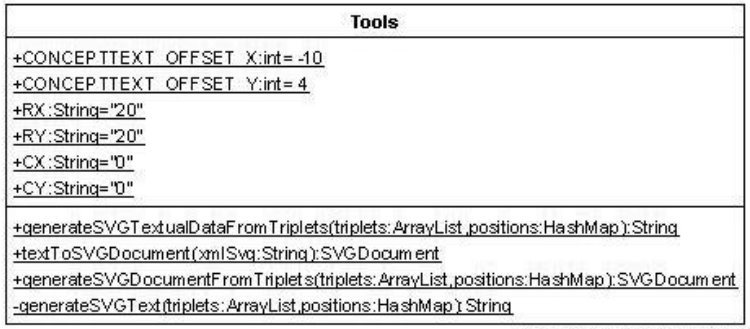












Appendiks 5 – JavaDoc

Dette appendikset inneholder javadoc generert fra javadoc kommentarer i koden. RTFDoclet er brukt for å få et "word-vennlig" format (standard doclet genererer html).

Package genetuc_kb_grensesnitt

public class **genetuc_kb_grensesnitt.NoDatasourceHandlerException** extends java.lang.Exception

Enkel klasse for å indikere at det ikke finnes en datakildehåndterer til ønsket datakilde

Constructors

public NoDatasourceHandlerException(
String s)

public class **genetuc_kb_grensesnitt.TucGuiException** extends java.lang.Exception

Enkel subklasse av Exception for å identifisere TucGui unntak.

Constructors

public TucGuiException(
String s)

public class **genetuc_kb_grensesnitt.Tools**

Klassen inneholder diverse verktøy-metoder for å generere svg-dokumenter.

Constructors

public Tools()

Methods

public static java.lang.String generateSVGTextualDataFromTriplets(
ArrayList triplets,
HashMap positions)
Metoden genererer svg-tekst (se diplom teksten for forklaring til struktur) basert på tripletter med relasjoner (konsept relasjon konsept, f.eks. Jeg eren gutt)

public static org.w3c.dom.svg.SVGDocument textToSVGDocument(
String xmlSvg)
Lager SVGDocument av tekst (som må være svg-xml)

public static org.w3c.dom.svg.SVGDocument generateSVGDocumentFromTriplets(
ArrayList triplets,
HashMap positions)
Metode som tar inn tripletter med relasjoner (konsept relasjon konsept, f.eks. Jeg eren gutt) og returnerer SVGDocument

Fields

public static final CONCEPTTEXT_OFFSET_X
Bestemmer x- og y-offset for tekst (konseptnavn) sånn den vises i ellipsene hos klienten

public static final CONCEPTTEXT_OFFSET_Y
Bestemmer x- og y-offset for tekst (konseptnavn) sånn den vises i ellipsene hos klienten

public static final RX
Bestemmer radius og senter (offset) for konsept-ellipsene som vises hos klient

public static final RY
Bestemmer radius og senter (offset) for konsept-ellipsene som vises hos klient

public static final CX
Bestemmer radius og senter (offset) for konsept-ellipsene som vises hos klient

public static final CY
Bestemmer radius og senter (offset) for konsept-ellipsene som vises hos klient

public class **genetuc_kb_grensesnitt.Logger**

Klassen brukes for å ha en sentral plass hvor output fra server og klient (hver for seg) går via. Tanken er at man lett kan implementere at tekst fra logg skal kunne sendes tilfeldig sted, f.eks. over nett/til en mail adresse etc. En annen ting er at man har forskjellige nivåer av hvor mye som blir skrevet ut, f.eks. info, debug, kritisk.

Constructors **public Logger()**

Methods **public static void p(**
int type,
String printfString,
Object[] objs)
Metoden logger (per nå: til skjerm) hvis nivået beskjeden er på er ønsket.
Parameters
type - hvilket nivå beskjeden er (kritisk, info debug), bruk f.eks. Tools.INFO
printfString - en streng som skal skrives ut, samme type strenger som System.printf tar
objs - objekter som refereres i printfString

Fields **public static final INFO**

public static final DEBUG

public static final CRITICAL

public static printLevel

Package genetuc_kb_grensesnitt.server

public class **genetuc_kb_grensesnitt.server.GraphUpdateDummy**

Skall-klasse siden Spring implementasjonen krever en slik (den brukes egentlig for å oppdatere grafikk (i et annet system). Som vi ikke skal gjøre i denne settingen).

Constructors	**public GraphUpdateDummy()**
Methods	**public void scale(** **double scaleval)**
	public void update(**boolean adjust_bounds)**
	public void center()
	public double getVSpacing()
	public EDU.auburn.VGJ.graph.Node getSelectedNode()
	public EDU.auburn.VGJ.util.DRect windowRect()
	public java.awt.Frame getFrame()
	public double getHSpacing()

public class **genetuc_kb_grensesnitt.server.GraphLayout**

Constructors	**public GraphLayout()**
Methods	**public static java.util.HashMap getPositionsForConcepts(** **ArrayList relations)** Metoden tar inn en liste med relasjoner som konverteres til en Graph som Spring algoritmen tar inn. **Returns** Map med konseptnavn og posisjoner

public interface **genetuc_kb_grensesnitt.server.RMIClassInterface** implements java.rmi.Remote

Interface for Remote klasse som aksesseres fra klient.

Methods	**public java.lang.String getAnSVG(** **String conceptName,** **int maxDistance,** **int fromIndex,** **int toIndex)** Metoden returnerer svg på textlig form (xml) **Parameters** conceptName - navn på konsept som er i fokus maxDistance - hvor mange kanter vekk fra konsept som er i fokus

skal hentes? Jo lengre distanse, jo fler noder => lengre tid for graf-algoritme å bli ferdig

fromIndex - optimeringstiltak: selv med maxDistance = 1 kan det bli for mange noder (f.eks. hvis "thing" er fokuskonsept). Da setter indexFrom og indexTo et intervall av relasjoner som hentes.

toIndex - optimeringstiltak: selv med maxDistance = 1 kan det bli for mange noder (f.eks. hvis "thing" er fokuskonsept). Da setter indexFrom og indexTo et intervall av relasjoner som hentes.

public void deleteConcept(
String conceptName)

Sletter et konsept.

public void deleteRelation(
String conceptFrom,
String conceptTo,
String relationName)

Sletter en relasjon

public void renameConcept(
String oldConceptName,
String newConceptName)

Endrer navn på et konsept

public void renameRelation(
String conceptFrom,
String conceptTo,
String oldRelationName,
String newRelationName)

Endrer navn på en relasjon

public void createRelation(
String conceptFrom,
String relation,
String conceptTo)

Oppretter en ny relasjon. Nye relasjoner blir lagt i den første datakilden.

public java.lang.String popMessageFromQueue()

Returnerer det som ligger på toppen av meldingskøen. Eller null

Fields **public static final messageFIFOQueue**

public interface **genetuc_kb_grensesnitt.server.Datasource**

Interfacet definerer en datakilde.

Methods **public void deleteConcept(**
String conceptName)

sletter også alle relasjoner konseptet er med i

public void deleteRelation(
String conceptFrom,
String conceptTo,
String relationName)

Sletter relasjon

public void renameConcept(
String oldConceptName,
String newConceptName)

Endrer navn på et konsept. Konsepter antaes å være unike

public void renameRelation(
String conceptFrom,
String conceptTo,
String oldRelationName,
String newRelationName)

Gir nytt navn til relasjon

public void createRelation(
String conceptFrom,
String relation,
String conceptTo)

Lager ny relasjon

public java.util.HashSet getConcepts()
Returnerer konsepter

public java.util.ArrayList getRelations()
Returnerer relasjoner

public void immediateWrite()
skriver til kilde umiddelbart. server er på vei ned. Hvis datakilden ikke støtter skriving implementeres denne tom.

public class **genetuc_kb_grensesnitt.server.KBFile** implements genetuc_kb_grensesnitt.server.Datasource

Klassen implementerer datakilde interfacet mot prolog-filer som tradisjonelt er kunnskapskilde i GeneTuc.

Constructors

public KBFile(
String filename)

Methods

public java.util.ArrayList getRelations()
Metode som returnerer de relasjoner som er parset inn fra fil

public void writeToDisk()
Metoden renamer gjeldende fil til [navn][tall].pl hvor tall er det neste tall som det ikke allerede eksisterer fil for. Skriver og tar backup for hver 10-ende kall til denne metoden.

public void deleteConcept(
String conceptName)
sletter også alle relasjoner konseptet er med i

public void deleteRelation(
String conceptFrom,
String conceptTo,
String relationName)

Sletter en relasjon

public void renameConcept(
String oldConceptName,
String newConceptName)

Endrer navn på et konsept. Konsepter antaes å være unike

public void renameRelation(
String conceptFrom,
String conceptTo,

String oldRelationName,
String newRelationName)

Endrer navn på en relasjon

public void createRelation(
String conceptFrom,
String relation,
String conceptTo)

Oppretter en ny relasjon

public java.util.HashSet getConcepts()
Returnerer konsepter som er hentet ut fra kildefilen.

public void immediateWrite()
Metoden skriver buffer til disk umiddelbart. Brukes når programmet avsluttes og endringer trenger å skrives umiddelbart

public class **genetuc_kb_grensesnitt.server.Controller**

Klassen setter opp og starter serveren.

Constructors **public Controller()**

Methods **public static void main(**
String[] args)
Metoden starter serveren. Eneste argument er filnavn til konfigureringsfilen.

public static java.util.ArrayList getDatasources()
Returnerer de datakildene som er satt opp og klare

Fields **public static rmiInstance**
rmiInstansen som er knyttet til registeret

public class **genetuc_kb_grensesnitt.server.RMIClassImpl** extends java.rmi.server.UnicastRemoteObject implements genetuc_kb_grensesnitt.server.RMIClassInterface

Klassen implementerer RMIClassInterface. Brukes som inngangsport til serveren fra klient.

Constructors **public RMIClassImpl()**

Methods **public java.lang.String getAnSVG(**
String conceptName,
int maxDistance,
int fromIndex,
int toIndex)
Se RMIClassInterface

public java.util.ArrayList getRelevantRelations(
String startConcept,
ArrayList allRelations,
int maxDistance,
int fromIndex,
int toIndex)

Metoden filtrerer mhp maxDistance, indexFrom og indexTo for at graf algoritmen skal bli ferdig innen rimelig tid.

public void deleteConcept(
String conceptName)
Se RMIClassInterface

public void deleteRelation(
String conceptFrom,
String conceptTo,
String relationName)
Se RMIClassInterface

public void renameConcept(
String oldConceptName,
String newConceptName)
Se RMIClassInterface

public void renameRelation(
String conceptFrom,
String conceptTo,
String oldRelationName,
String newRelationName)
Se RMIClassInterface

public void createRelation(
String conceptFrom,
String relation,
String conceptTo)
Se RMIClassInterface

public java.lang.String popMessageFromQueue()

Package genetuc_kb_grensesnitt.client

public class **genetuc_kb_grensesnitt.client.ClientConfigForm** extends javax.swing.JFrame

Klassen representerer vinduet som kommer opp når man konfigurerer tilkobling til server

Constructors

public ClientConfigForm(
ClientGUI parentClientGUI)

Methods

public static void main(
String[] args)
Starter klienten. I tilfelle JWS er det JWS som kaller metoden.

public class **genetuc_kb_grensesnitt.client.GUIJFrame** extends javax.swing.JFrame

Klassen representerer hoved-vinduet til klient-side gui-et.

Constructors

public GUIJFrame(
ClientGUI parent)

Methods

public int[] getConfigNumbers()
Returnerer de valg som er gjort mhp maxDistance, indexFrom og indexTo. Disse brukes både når knapp for å hente konsept trykke og når man dobbelklikker et konsept

public org.apache.batik.swing.JSVGCanvas getJSVGCanvas1()

public class **genetuc_kb_grensesnitt.client.ClientGUI**

Klassen setter opp og starter klient-siden

Constructors

public ClientGUI(
int printLevel)
instansierer klienten
Parameters
printLevel - hvor mye skal logges? Debug/Info, @see INFO

Methods

public static void main(
String[] args)
starter klienten

public void connect(
String connectString,
String firstConcept,
int maxDistance,
int fromIndex,
int toIndex)
Kobler til server
Parameters
connectString - server og portnummer rmiregisteret kjører på.
firstConcept - det konseptet som skal hentes
maxDistance - @see maxDistance

fromIndex - av optimeringshensyn. gjør et utvalg (setter nedre grense) fra listen med relasjoner som er relevante tatt firstConcept og maxDistance i betraktning. @see genetuc_kb_grensesnitt.server.RMIClassImpl#getRelevantRelations
toIndex - av optimeringshensyn. gjør et utvalg (setter øvre grense) fra listen med relasjoner som er relevante tatt firstConcept og maxDistance i betraktning. @see genetuc_kb_grensesnitt.server.RMIClassImpl#getRelevantRelations

public void displayConcept(
String conceptName,
int maxDistance,
int fromIndex,
int toIndex)
Når klienten allerede er koblet til serveren og vi hente et nytt konsept brukes denne metoden.
Parameters
conceptName - det konseptet som skal hentes
maxDistance - @see maxDistance
fromIndex - av optimeringshensyn. gjør et utvalg (setter nedre grense) fra listen med relasjoner som er relevante tatt firstConcept og maxDistance i betraktning. @see genetuc_kb_grensesnitt.server.RMIClassImpl#getRelevantRelations
toIndex - av optimeringshensyn. gjør et utvalg (setter øvre grense) fra listen med relasjoner som er relevante tatt firstConcept og maxDistance i betraktning. @see genetuc_kb_grensesnitt.server.RMIClassImpl#getRelevantRelations

public void deleteRelation(
SVGGElement relationContainer)
kalles når bruker vil slette en relasjon.
Parameters
relationContainer - det G (group) elementet som innkapsler relasjonen (se forklaring til svg-struktur i diplomen

public void updateRelationPositions(
SVGEllipseElement forConcept)
Metoden traverserer alle relasjoner for gitt konsept og tegner dem på nytt
Parameters
forConcept - konseptet hvis relasjoner skal rekalkuleres

public java.lang.String getConceptName(
String internalID)
Hjelpemetode som tar intern id og returnerer ekte konseptnavn (som blir brukt i kunnskapsbasen)
Parameters
internalID - intern id for konseptet man vil ha ekte navn til

public void deleteConcept(
SVGGElement conceptContainer)
Sletter et konsept, både i gui og på server. Sletting av konsept resulterer i sletting av dets relasjoner. Dersom sletting av relasjoner medfører at et konsept står alene bli også dette konseptet slettet. (pl filene inneholder ikke ensomme facts)
Parameters
conceptContainer - group element som innkapsler konseptet som ønskes slettet

public org.w3c.dom.svg.SVGGElement newConcept(
String conceptName,
int xPos,
int yPos)
Metode for å opprette nytt konsept. Ingenting blir skrevet til kunnskapsbasen før dette konseptet blir brukt i en relasjon.
Parameters
conceptName - konseptnavn slik det vil være i kunnskapsbasen
xPos - x posisjon i gui-et
yPos - y posisjon i gui-et

Fields

public static final INFO
tilstandskonstant for loggemeldinger på info-nivå

public static final DEBUG
tilstandskonstant for loggemeldinger på debug-nivå

public static final MODE_CONCEPT
tilstandskonstant for konsept-modus

public static final MODE_RELATION
tilstandskonstant for relasjons-modus

public static final MODE_DELETE
tilstandskonstant for slette-modus

public static mode
variabel som indikerer om tilstand er relasjon, konsept eller slett

public class **genetuc_kb_grensesnitt.client.MouseHandler** implements org.w3c.dom.events.EventListener

Klassen håndterer alle hendelser fra musen.

Constructors

public MouseHandler(
ClientGUI cGUI,
RMIClassInterface enRMIKlasse,
JSVGCanvas svgCanvas,
SVGDocument svgDocument)
Instansierer MouseHandler

Methods

public void handleEvent(
Event evt)
Denne metoden blir kalt når en mus-hendelse skjer. Metoden tar tilstandsflagg og modus-knappenes tilstand i betraktning. Alle hendelsene mouseclick/drag/up/down ender opp her.

Appendiks 6 – Kildekode

Dette vedlegget inneholder all kildekode som er produsert. Det er vedlagt for kompletthets skyld.

```
package genetuc_kb_grensesnitt.client;

import java.net.MalformedURLException;
import java.rmi.NotBoundException;
import java.rmi.RemoteException;
import javax.swing.JOptionPane;

/**
 * Klassen representerer vinduet som kommer opp når man konfigurerer tilkobling til server
 *
 */
public class ClientConfigForm extends javax.swing.JFrame {

    /**bruker noen metoder i ClientGUI, så trenger referanse til den*/
    ClientGUI parentClientGUI;

    public ClientConfigForm(ClientGUI parentClientGUI) {
        this.parentClientGUI = parentClientGUI;
        initComponents();
    }

    /** Setter opp komponentene i vinduet
     */
    private void initComponents() {//GEN-BEGIN:initComponents
        jButton1 = new javax.swing.JButton();
        jTextField4 = new javax.swing.JTextField();
        jTextField1 = new javax.swing.JTextField();
        jTextField5 = new javax.swing.JTextField();
        jLabel3 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        jTextField2 = new javax.swing.JTextField();
        jTextField3 = new javax.swing.JTextField();
        jLabel1 = new javax.swing.JLabel();
        jLabel4 = new javax.swing.JLabel();
        jLabel5 = new javax.swing.JLabel();

        getContentPane().setLayout(new org.netbeans.lib.awtextra.AbsoluteLayout());

        setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
        setForeground(java.awt.Color.white);
        setMaximizedBounds(new java.awt.Rectangle(0, 0, 200, 200));
        setResizable(false);
        jButton1.setText("koble til");
        jButton1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });

        getContentPane().add(jButton1, new org.netbeans.lib.awtextra.AbsoluteConstraints(170, 170, -1, -1));

        jTextField4.setText("0");
        jTextField4.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jTextField4ActionPerformed(evt);
            }
        });

        getContentPane().add(jTextField4, new org.netbeans.lib.awtextra.AbsoluteConstraints(120, 100, 40, -1));

        jTextField1.setText("127.0.0.1:1099");
        getContentPane().add(jTextField1, new org.netbeans.lib.awtextra.AbsoluteConstraints(120, 10, 110, -1));

        jTextField5.setText("50");
        getContentPane().add(jTextField5, new org.netbeans.lib.awtextra.AbsoluteConstraints(120, 130, 40, -1));

        jLabel3.setText("startkonsept");
        getContentPane().add(jLabel3, new org.netbeans.lib.awtextra.AbsoluteConstraints(10, 70, -1, 20));
```

```
        jLabel2.setText("maks distanse");
        getContentPane().add(jLabel2, new org.netbeans.lib.awtextra.AbsoluteConstraints(10, 40, -1, 20));

        jTextField2.setText("2");
        getContentPane().add(jTextField2, new org.netbeans.lib.awtextra.AbsoluteConstraints(120, 40, 30, -1));

        jTextField3.setText("protein");
        getContentPane().add(jTextField3, new org.netbeans.lib.awtextra.AbsoluteConstraints(120, 70, 110, -1));

        jLabel1.setText("server:port");
        getContentPane().add(jLabel1, new org.netbeans.lib.awtextra.AbsoluteConstraints(10, 10, 60, 20));

        jLabel4.setText("index fra");
        getContentPane().add(jLabel4, new org.netbeans.lib.awtextra.AbsoluteConstraints(10, 100, -1, -1));

        jLabel5.setText("index til");
        getContentPane().add(jLabel5, new org.netbeans.lib.awtextra.AbsoluteConstraints(10, 130, -1, -1));

        pack();
    }//GEN-END:initComponents

    private void jTextField4ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_jTextField4ActionPerformed
        // TODO add your handling code here:
    }//GEN-LAST:event_jTextField4ActionPerformed

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_jButton1ActionPerformed

        /*connect kaster 3 exceptions som er interessante å håndtere "gracefully" slik at brukeren slipper rive av seg håret.
         dersom en exception blir kastet vises en feilmelding og brukeren får mulighet til å endre inn-data, hvis det ikke kastes
         exception blir bare vinduet disposet (tilkobling er oppe)*/

        try {
            parentClientGUI.connect(jTextField1.getText(),                                                                    jTextField3.getText(),
Integer.parseInt(jTextField2.getText()),Integer.parseInt(jTextField4.getText()),Integer.parseInt(jTextField5.getText()));

        } catch(MalformedURLException mfue){
            JOptionPane.showMessageDialog(this,"urlen var ikke vel-formet. eks: 127.0.0.1:1099 ","Feil i URL",JOptionPane.ERROR_MESSAGE);
            return;
        } catch(RemoteException re){
            JOptionPane.showMessageDialog(this,"kommunikasjonsfeil: "+re.toString(),"Kommunikasjonsfeil",JOptionPane.ERROR_MESSAGE);
            return;
        } catch(NotBoundException nbe){
            JOptionPane.showMessageDialog(this,"Klassen var ikke bundet i rmi-registeret. Sjekk serveren. ","rmi klasse ikke
bundet",JOptionPane.ERROR_MESSAGE);
            return;
        }

        this.dispose();
    }//GEN-LAST:event_jButton1ActionPerformed

    /**
     * Starter klienten. I tilfelle JWS er det JWS som kaller metoden.
     */
    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new ClientConfigForm(null).setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JButton jButton1;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JTextField jTextField1;
    private javax.swing.JTextField jTextField2;
    private javax.swing.JTextField jTextField3;
    private javax.swing.JTextField jTextField4;
    private javax.swing.JTextField jTextField5;
    // End of variables declaration//GEN-END:variables

}

package genetuc_kb_grensesnitt.client;

import genetuc_kb_grensesnitt.Logger;
```

```
import genetuc_kb_grensesnitt.Tools;
import genetuc_kb_grensesnitt.TucGuiException;
import genetuc_kb_grensesnitt.server.RMIClassInterface;
import java.io.IOException;
import java.net.MalformedURLException;
import java.rmi.Naming;
import java.rmi.NotBoundException;
import java.rmi.RemoteException;
import java.util.ArrayList;
import javax.swing.JOptionPane;
import org.apache.batik.swing.*;
import org.apache.batik.swing.gvt.GVTTreeRendererAdapter;
import org.apache.batik.swing.gvt.GVTTreeRendererEvent;
import org.apache.batik.swing.svg.GVTTreeBuilderAdapter;
import org.apache.batik.swing.svg.GVTTreeBuilderEvent;
import org.apache.batik.swing.svg.SVGDocumentLoaderAdapter;
import org.apache.batik.swing.svg.SVGDocumentLoaderEvent;
import org.w3c.dom.events.*;
import org.w3c.dom.svg.*;
import org.w3c.dom.*;

/**
 * Klassen setter opp og starter klient-siden
 */

public class ClientGUI {

    /**tilstandskonstant for loggemeldinger på info-nivå*/
    final public static int INFO = 0;
    /**tilstandskonstant for loggemeldinger på debug-nivå*/
    final public static int DEBUG = 1;

    /**tilstandskonstant for konsept-modus*/
    final public static int MODE_CONCEPT = 0;
    /**tilstandskonstant for relasjons-modus*/
    final public static int MODE_RELATION = 1;
    /**tilstandskonstant for slette-modus*/
    final public static int MODE_DELETE = 2;

    /**variabel som indikerer om tilstand er relasjon, konsept eller slett*/
    public static int mode = MODE_CONCEPT;

    /**instansen som fungerer som knytning mellom server og klient*/
    RMIClassInterface enRMIKlasse;
    GUIJFrame gjf = new GUIJFrame(this);
    JSVGCanvas svgCanvas= gjf.getJSVGCanvas1();
    SVGDocument svgDocument;

    /** maxDistance angir hvor mange steg (relasjoner) vekk fra ønsket konsept som skal inkluderes, avhenger av indeksFra/Til også*/
    private int maxDistance = 1;

    /**navnerom for svg*/
    String svgNS = "http://www.w3.org/2000/svg";
    /** navnerom for å skille to og from i svg-dokument som sendes mellom server og klient*/
    String geneTucNS = "http://www.idi.ntnu.no/GeneTuc/";

    /** klasse for å følge med på om GVT-treet er ferdig bygget. Må vente til det er bygget før svgDokument kan hentes ut fra Batik*/
    GVTTreeBuilderAdapter gvtba = new GVTTreeBuilderAdapter() {
        public void gvtBuildCompleted(GVTTreeBuilderEvent evt) {
            synchronized(this){notifyAll();}
        }
        public void gvtBuildFailed(GVTTreeBuilderEvent evt){
            JOptionPane.showMessageDialog(gjf,"En feil skjedde under lasting av svg. "+evt.toString(),"Feil under
lasting",JOptionPane.ERROR_MESSAGE);
        }
        public void gvtBuildCancelled(GVTTreeBuilderEvent evt){
            JOptionPane.showMessageDialog(gjf,"En feil skjedde under lasting av svg. "+evt.toString(),"Feil under
lasting",JOptionPane.ERROR_MESSAGE);
        }
    };

    /** klasse for å følge med på om svgDokumentet man gir batik er lastet. Må vente til det er bygget før svgDokument kan hentes ut fra Batik*/
    SVGDocumentLoaderAdapter dla = new SVGDocumentLoaderAdapter(){
        public void documentLoadingCompleted(SVGDocumentLoaderEvent e){
            synchronized(this){notify();}
        }
    };

    /** starter klienten*/
    public static void main(String[] args){
```

```
        new ClientGUI(Logger.INFO);
    }

    /** instansierer klienten
     *@param printLevel hvor mye skal logges? Debug/Info, @see INFO
     */
    public ClientGUI(int printLevel) {
        Logger.printLevel = printLevel;
        svgCanvas.addGVTTreeBuilderListener(gvtba);
        svgCanvas.setDocumentState(JSVGCanvas.ALWAYS_DYNAMIC);
        svgCanvas.setEnableRotateInteractor(false);
        svgCanvas.loadSVGDocument(this.getClass().getClassLoader().getResource("logo.svg").toString());
        gjf.setVisible(true);
    }

    /** Kobler til server
     *@param connectString server og portnummer rmiregisteret kjører på.
     *@param firstConcept det konseptet som skal hentes
     *@param maxDistance @see maxDistance
     *@param fromIndex av optimeringshensyn. gjør et utvalg (setter nedre grense) fra listen med relasjoner som er relevante tatt firstConcept og
maxDistance i betraktning. @see genetuc_kb_grensesnitt.server.RMIClassImpl#getRelevantRelations
     *@param toIndex av optimeringshensyn. gjør et utvalg (setter øvre grense) fra listen med relasjoner som er relevante tatt firstConcept og
maxDistance i betraktning. @see genetuc_kb_grensesnitt.server.RMIClassImpl#getRelevantRelations
     */
    public void connect(String connectString, String firstConcept,
            int maxDistance, int fromIndex, int toIndex) throws MalformedURLException, NotBoundException, RemoteException{
        SVGDocument svgd = null;
        this.maxDistance = maxDistance;

        //kobler til server og henter svg på text format
        enRMIKlasse = (RMIClassInterface)Naming.lookup("rmi://"+connectString+"/TucGuiRMIClass");
        try{
            svgd = Tools.textToSVGDocument(enRMIKlasse.getAnSVG(firstConcept, maxDistance, fromIndex, toIndex));
            final SVGDocument svgdFinal = svgd; //må det får at metoden i tråden skal ha det bra
            (new Thread(){
                public void run(){setWaitAndFetch(svgdFinal);}
            }).start();

        } catch(Exception e){e.printStackTrace();}
        Thread t = new Thread("Thread_for_checking_messagequeue"){
            public void run(){

                try{
                    while(true){
                        String message = enRMIKlasse.popMessageFromQueue(); //poll returnerer null om listen er tom
                        if(message != null) JOptionPane.showMessageDialog(gjf,message,"Beskjed fra
serveren",JOptionPane.INFORMATION_MESSAGE);
                        this.sleep(3000);
                    }
                }catch(InterruptedException ie){
                    //bare for å stoppe tråden
                } catch(NullPointerException npe){npe.printStackTrace();}
                catch(RemoteException re){re.printStackTrace();}
            }
        };
        t.start();

    }

    /**Sett, vent og hent. Metode som setter svgCanvas sin document, venter på at GVT (en intern representasjon
     * av SVG-en Batik bruker) skal bygges og henter dokumentet som Batik følger med på tilbake. (Det dokumentet som man
     * putter inn blir klonet, så man kan ikke endre på det dokumentet man allerede har. Det må hentes ut igjen.*/
    private void setWaitAndFetch(Document doc){
        SVGDocument oldDoc = svgCanvas.getSVGDocument();
        svgCanvas.setDocument(doc);
        if(svgCanvas.getSVGDocument() == oldDoc) //hvis dokument ikke er endret ennå, fyr av en tråd som venter
            try{synchronized(gvtba){gvtba.wait(0);}}catch(Exception e){e.printStackTrace();}
        svgDocument = svgCanvas.getSVGDocument();
        EventListener listener = new MouseHandler(this, enRMIKlasse, svgCanvas, svgDocument);
        ((EventTarget)svgDocument).addEventListener("mousedown", listener,false);
        ((EventTarget)svgDocument).addEventListener("mouseup", listener,false);
        gjf.repaint();
    }

    /**Når klienten allerede er koblet til serveren og vi hente et nytt konsept brukes denne metoden.
     *@param conceptName det konseptet som skal hentes
     *@param maxDistance @see maxDistance
     *@param fromIndex av optimeringshensyn. gjør et utvalg (setter nedre grense) fra listen med relasjoner som er relevante tatt firstConcept og
maxDistance i betraktning. @see genetuc_kb_grensesnitt.server.RMIClassImpl#getRelevantRelations
```

```
    *@param toIndex av optimeringshensyn. gjør et utvalg (setter øvre grense) fra listen med relasjoner som er relevante tatt firstConcept og
maxDistance i betraktning. @see genetuc_kb_grensesnitt.server.RMIClassImpl#getRelevantRelations
    */
    public void displayConcept(String conceptName, int maxDistance, int fromIndex, int toIndex) throws NullPointerException, RemoteException,
IOException, TucGuiException{
        String svgText = enRMIKlasse.getAnSVG(conceptName, maxDistance, fromIndex, toIndex);
        final SVGDocument svgdFinal = Tools.textToSVGDocument(svgText); //må det får at metoden i tråden skal kose seg
        (new Thread(){
            public void run(){setWaitAndFetch(svgdFinal);}
        }).start();
    }

    /**metode som blokkerer til GVT tree er bygget*/
    private void waitForGVTTreeBuilding(){
        /*lager en Render-lytter som notifyer alle som venter på monitor på den selv når redering er ferdig.
        Dermed kan de som vil vente på rendering kalle wait på lytter-objektet.*/

        GVTTreeBuilderAdapter gvtba = new GVTTreeBuilderAdapter() {
            public void gvtBuildCompleted(GVTTreeBuilderEvent evt) {
                synchronized(this){notify();}
            }
            public void gvtBuildFailed(GVTTreeBuilderEvent evt){
                JOptionPane.showMessageDialog(gjf,"En feil skjedde under lasting av svg. "+evt.toString(),"Feil under
lasting",JOptionPane.ERROR_MESSAGE);
            }
            public void gvtBuildCancelled(GVTTreeBuilderEvent evt){
                JOptionPane.showMessageDialog(gjf,"En feil skjedde under lasting av svg. "+evt.toString(),"Feil under
lasting",JOptionPane.ERROR_MESSAGE);
            }
        };
        svgCanvas.addGVTTreeBuilderListener(gvtba);
        try{synchronized(gvtba){gvtba.wait(0);}}catch(Exception e){e.printStackTrace();}
    }

    /**metode som blokkerer til GVT tree er rendret*/
    private void waitForGVTTreeRendering(){
        /*lager en Render-lytter som notifyer alle som venter på monitor på den selv når redering er ferdig.
        Dermed kan de som vil vente på rendering kalle wait på lytter-objektet.*/

        GVTTreeRendererAdapter gvtra = new GVTTreeRendererAdapter() {
            public void gvtRenderingCompleted(GVTTreeRendererEvent evt) {
                synchronized(this){notify();}
            }
            public void gvtRenderingFailed(GVTTreeRendererEvent evt){
                JOptionPane.showMessageDialog(gjf,"En feil skjedde under lasting av svg. "+evt.toString(),"Feil under
lasting",JOptionPane.ERROR_MESSAGE);
            }
            public void gvtRenderingCancelled(GVTTreeRendererEvent evt){
                JOptionPane.showMessageDialog(gjf,"En feil skjedde under lasting av svg. "+evt.toString(),"Feil under
lasting",JOptionPane.ERROR_MESSAGE);
            }
        };
        svgCanvas.addGVTTreeRendererListener(gvtra);
        try{synchronized(gvtra){gvtra.wait(0);}}catch(Exception e){e.printStackTrace();}
    }

    /**kalles når bruker vil slette en relasjon.
     *@param relationContainer det G (group) elementet som innkapsler relasjonen (se forklaring til svg-struktur i diplomen
     */
    public void deleteRelation(SVGGElement relationContainer) throws Exception{
        String conceptToInternalID = ((SVGPathElement)((SVGGElement)relationContainer).getFirstChild()).getAttributeNS(geneTucNS,"to");
        String conceptFromInternalID = ((SVGPathElement)((SVGGElement)relationContainer).getFirstChild()).getAttributeNS(geneTucNS,"from");
        String relationName = relationContainer.getChildNodes().item(1).getChildNodes().item(0).getNodeValue();
        String conceptTo = getConceptName(conceptToInternalID);
        String conceptFrom = getConceptName(conceptFromInternalID);

        //sletter på server
        Logger.p(Logger.DEBUG, "deleteRelation: sletter på server...");
        enRMIKlasse.deleteRelation(conceptFrom, conceptTo, relationName);
        Logger.p(Logger.DEBUG, "ok\n");

        //sletter i GUI
        Logger.p(Logger.DEBUG, "deleteRelation: sletter i GUI...");
        ((Element)relationContainer.getParentNode()).removeChild(relationContainer);
        Logger.p(Logger.DEBUG, "ok\n");
    }

    /**Hjelpemetode som går igjennom SVG-DOM og finner de relasjoner som er knyttet til konsept med gitt internt navn
     *@param internalConceptID intern id (se forklaring på svg-struktur i diplomen) konsept man leter etter relasjonene til
     */
```

```
    private ArrayList getRelationsForConcept(String internalConceptID){
        Element forConcept = svgDocument.getElementById(internalConceptID);
        NodeList relations = svgDocument.getElementById("relations").getElementsByTagName("g");
        ArrayList returnList = new ArrayList();
        for(int i=0;i<relations.getLength();i++){
            SVGPathElement item = (SVGPathElement)((SVGGElement)relations.item(i)).getElementsByTagNameNS(svgNS,"path").item(0);
            String conceptName = internalConceptID;
            if(item.getAttributeNS(geneTucNS,"from").equals(conceptName)){
                returnList.add(item);
            }
            if(item.getAttributeNS(geneTucNS,"to").equals(conceptName)){
                returnList.add(item);
            }
        }
        return returnList;
    }

    /**Metoden traverserer alle relasjoner for gitt konsept og tegner dem på nytt
     *@param forConcept konseptet hvis relasjoner skal rekalkuleres
     */
    public void updateRelationPositions(SVGEllipseElement forConcept){
        NodeList relations = svgDocument.getElementById("relations").getElementsByTagName("g");
        for(int i=0;i<relations.getLength();i++){
            SVGPathElement item = (SVGPathElement)((SVGGElement)relations.item(i)).getElementsByTagNameNS(svgNS,"path").item(0);
            if(item.getAttributeNS(geneTucNS,"from").equals(((Element)forConcept.getParentNode()).getAttributeNS(null,"id"))){
                String[] sArray = item.getAttribute("d").split(" "); //d attributtet har space-separerte dataelementer. derfor spil på " "
                String temp="";
                sArray[1]= String.valueOf(forConcept.getCTM().getE()); sArray[2]=String.valueOf(forConcept.getCTM().getF());
                for(String s : sArray){temp = temp + s + " ";}
                item.setAttribute("d",temp);
                float x = (Float.parseFloat(sArray[1])+Float.parseFloat(sArray[sArray.length-2]))/2; //finner x verdi mellom paths startX og sluttX
                float y = (Float.parseFloat(sArray[2])+Float.parseFloat(sArray[sArray.length-1]))/2; //finner y verdi mellom paths startY og sluttY
                SVGTextElement relationText =
(SVGTextElement)((SVGGElement)relations.item(i)).getElementsByTagNameNS(svgNS,"text").item(0); //TODO: refactor; putt lange referanser
i egne vars
                relationText.setAttribute("x",String.valueOf(x));
                relationText.setAttribute("y",String.valueOf(y));
            }
            if(item.getAttributeNS(geneTucNS,"to").equals(((Element)forConcept.getParentNode()).getAttributeNS(null,"id"))){
                String[] sArray = item.getAttribute("d").split(" ");
                String temp="";
                sArray[sArray.length-2]= String.valueOf(forConcept.getCTM().getE()); sArray[sArray.length-
1]=String.valueOf(forConcept.getCTM().getF());
                for(String s : sArray){temp = temp + s + " ";}

                item.setAttribute("d",temp);

                float x = (Float.parseFloat(sArray[1])+Float.parseFloat(sArray[sArray.length-2]))/2; //finner x verdi mellom paths startX og sluttX
                float y = (Float.parseFloat(sArray[2])+Float.parseFloat(sArray[sArray.length-1]))/2; //finner y verdi mellom paths startY og sluttY
                SVGTextElement relationText =
(SVGTextElement)((SVGGElement)relations.item(i)).getElementsByTagNameNS(svgNS,"text").item(0);
                relationText.setAttribute("x",String.valueOf(x));
                relationText.setAttribute("y",String.valueOf(y));
            }
        }
    }

    /**Hjelpemetode som tar intern id og returnerer ekte konseptnavn (som blir brukt i kunnskapsbasen)
     *@param internalID intern id for konseptet man vil ha ekte navn til
     */
    public String getConceptName(String internalID){
        return ((Element)svgDocument.getElementById(internalID)).getChildNodes().item(1).getChildNodes().item(0).getNodeValue();
    }

    /**Sletter et konsept, både i gui og på server. Sletting av konsept resulterer i sletting av dets relasjoner.
     *Dersom sletting av relasjoner medfører at et konsept står alene bli også dette konseptet slettet. (pl filene inneholder ikke ensomme facts)
     *@param conceptContainer group element som innkapsler konseptet som ønskes slettet
     */
    public void deleteConcept(SVGGElement conceptContainer) throws Exception{
        String internalID = conceptContainer.getId();

        //sletter på server
        Logger.p(Logger.DEBUG, "deleteConcept: sletter på server...");
        enRMIKlasse.deleteConcept(getConceptName(internalID));
        Logger.p(Logger.DEBUG, "ok\n");

        //sletter relasjoner dette konseptet var med i
        Logger.p(Logger.DEBUG, "deleteConcept: sletter relasjoner konseptet var med i...");
        SVGGElement relations = (SVGGElement)svgDocument.getElementById("relations");
        NodeList nl = relations.getElementsByTagName("path");
```

```
        ArrayList<String> conceptsToBeChecked = new ArrayList<String>(); //sampler opp alle konsepter i andre enden av relasjoner og sjekker om de er ensomme etter at alle relasjoner fra dette konseptet er fjernet

        for(int i=nl.getLength()-1;i>=0;i--){
            Element pathElement = (Element)nl.item(i);
            String internalIDFrom = pathElement.getAttributeNS(geneTucNS,"from");
            String internalIDTo = pathElement.getAttributeNS(geneTucNS,"to");
            if(internalID.equals(internalIDFrom)){
                deleteRelation((SVGGElement)pathElement.getParentNode());
                conceptsToBeChecked.add(internalIDTo);
            }else if(internalID.equals(internalIDTo)){
                deleteRelation((SVGGElement)pathElement.getParentNode());
                conceptsToBeChecked.add(internalIDFrom); //mulig problem med selv-relasjoner her.
            }
        }

        for (String s : conceptsToBeChecked) if(getRelationsForConcept(s).isEmpty()) deleteConcept((SVGGElement)svgDocument.getElementById(s));
        Logger.p(Logger.DEBUG, "ok\n");

        //oppdaterer GUI
        Logger.p(Logger.DEBUG, "deleteConcept: sletter i GUI...");
        ((Element)conceptContainer.getParentNode()).removeChild(conceptContainer);
        Logger.p(Logger.DEBUG, "ok\n");
    }


    /**Metode for å opprette nytt konsept. Ingenting blir skrevet til kunnskapsbasen før dette konseptet blir brukt i en relasjon.
     *@param conceptName konseptnavn slik det vil være i kunnskapsbasen
     *@param xPos x posisjon i gui-et
     *@param yPos y posisjon i gui-et
     */
    public SVGGElement newConcept(String conceptName, int xPos, int yPos) throws Exception{
        SVGGElement gElem = (SVGGElement)svgDocument.createElementNS(svgNS,"g");
        SVGEllipseElement eElem = (SVGEllipseElement)svgDocument.createElementNS(svgNS,"ellipse");
        SVGTextElement tElem = (SVGTextElement)svgDocument.createElementNS(svgNS,"text");

        tElem.appendChild(svgDocument.createTextNode(conceptName));
        tElem.setAttribute("transform","translate("+Tools.CONCEPTTEXT_OFFSET_X+","+Tools.CONCEPTTEXT_OFFSET_Y+")");

        eElem.setAttribute("cx",Tools.CX);
        eElem.setAttribute("cy",Tools.CY);
        eElem.setAttribute("rx",Tools.RX);
        eElem.setAttribute("ry",Tools.RY);

        gElem.setAttribute("transform","translate("+xPos+","+yPos+")");
        gElem.setAttribute("id","concept_"+Math.random()); //interne id-er må være unike, noe de (så og si) blir med Math.random()
        gElem.appendChild(eElem);
        gElem.appendChild(tElem);

        return gElem;
    }
}

package genetuc_kb_grensesnitt.server;

import genetuc_kb_grensesnitt.Logger;
import genetuc_kb_grensesnitt.NoDatasourceHandlerException;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.rmi.ConnectException;
import java.rmi.Naming;
import java.util.ArrayList;
import java.util.StringTokenizer;


/**
 * Klassen setter opp og starter serveren.
 */
public class Controller {

    /**datakilder som serveren henter data fra og skriver data til*/
    static ArrayList<Datasource> datasources = new ArrayList<Datasource>();

    /**rmiInstansen som er knyttet til registeret*/
    public static RMIClassImpl rmiInstance;

    /**fabrikkmetode for å hente datakildehåndterer for en datakilde definert i serverens konfigureringsfil
     *Dersom systemet skal håndtere andre datakilder enn flate filer, f.eks. database/web, må mappinger for dem
```

```
     *legges til her.
     */
    private static Datasource getDatasource(String type, String parameter) throws NoDatasourceHandlerException{
        if(type.equals("file")) return new KBFile(parameter);
        else throw new NoDatasourceHandlerException("Finnes ingen håndterer for datakilde av type "+type);
    }

    /**laster datakildene som er definert i serverens konfigureringsfil*/
    private static void loadDatasources() throws NoDatasourceHandlerException{
        datasources.clear();
        String datasourcesConfig = System.getProperty("tucgui.datasources");

        for(String s : datasourcesConfig.split(";")) {
            StringTokenizer st = new StringTokenizer(s,",");
            String type = st.nextToken(), parameter = st.nextToken();
            try{datasources.add(getDatasource(type, parameter));} catch(NoDatasourceHandlerException ndshe){
                Logger.p(Logger.CRITICAL, "Fant ikke håndterer for %s (parameter = %s)",type,parameter);
            }
        }
    }

    /**Metoden starter serveren. Eneste argument er filnavn til konfigureringsfilen.*/
    public static void main(String[] args){
        Logger.p(Logger.INFO,"Starter server.");
        Logger.p(Logger.INFO,"Legger til shutdownhook.");
        Runtime.getRuntime().addShutdownHook(new MyShutDownThread());
        String serverString = "rmi://localhost:1099/TucGuiRMIClass";
        try{
            System.getProperties().load(new FileInputStream(args[0]));

            loadDatasources();
            Logger.p(Logger.INFO,"Binder rmi klasse til rmiregistry.");
            Logger.printLevel = Integer.parseInt(System.getProperty("tucgui.printLevel"));
            Naming.rebind(serverString, new RMIClassImpl());
            Logger.p(Logger.INFO,"Server startet. (%s).", serverString);
        } //TODO: hent debuglevel fra configfil, hent server/port fra configfil rename minTest
        catch(FileNotFoundException fnfe){
            System.out.println("\n***Kritisk: FileNotFoundException. Mest sannsynlig at serveren ikke fant konfigurerings-filen " +
                    "(sjekk path, bruk forward-slashes). Avslutter.");
            System.exit(1);;} catch(NumberFormatException nfe){
                System.out.println("\n***Kritisk: Integer.parseInt feilet. Sjekk tallverdi for tucgui.printlevel i konfigureringsfilen. Avslutter.");
                System.exit(1);;} catch(IOException ioe){
                    Logger.p(Logger.CRITICAL,"IO-feil: "+ioe.toString()+"\n\n");
                    ioe.printStackTrace();} catch(NoDatasourceHandlerException ndhe){Logger.p(Logger.CRITICAL,ndhe.toString());}
catch(IndexOutOfBoundsException iobe){System.out.println("første og eneste parameter til main metoden er kanonisk filnavn til
konfigureringsfil.");}
    }

    /**Returnerer de datakildene som er satt opp og klare*/
    public static ArrayList<Datasource> getDatasources(){
        return datasources;
    }

    /**Klassen er en tråd som registreres i Runtime.shutdownhook når Controller startes. Dette fordi datakilder kan
     *ha uskrevet data når bruker trykker ctrl+c for å stoppe server. Disse dataene må skrives til fil før server
     *stoppes
     */
}

class MyShutDownThread extends Thread{
    /**Metoden starter tråden*/
    public void run(){
        Logger.p(Logger.INFO,"Skriver data til kildefiler...");
        for(Datasource src : Controller.getDatasources()){
            try{src.immediateWrite();}
            catch(IOException ioe){ioe.printStackTrace();}
        }
        Logger.p(Logger.INFO,"Ferdig.");
    }
}

package genetuc_kb_grensesnitt.server;

import genetuc_kb_grensesnitt.NoDatasourceHandlerException;
import java.io.IOException;
import java.util.ArrayList;
import java.util.HashSet;

/**Interfacet definerer en datakilde.
 */
```

```
public interface Datasource {

    /**sletter også alle relasjoner konseptet er med i*/
    public void deleteConcept(String conceptName) throws IOException;

    /**Sletter relasjon*/
    public void deleteRelation(String conceptFrom, String conceptTo, String relationName) throws IOException;

    /**Endrer navn på et konsept. Konsepter antaes å være unike*/
    public void renameConcept(String oldConceptName, String newConceptName) throws IOException;

    /**Gir nytt navn til relasjon*/
    public void renameRelation(String conceptFrom, String conceptTo, String oldRelationName, String newRelationName) throws IOException;

    /**Lager ny relasjon*/
    public void createRelation(String conceptFrom, String relation, String conceptTo) throws IOException;

    /**Returnerer konsepter*/
    public HashSet<String> getConcepts();

    /**Returnerer relasjoner*/
    public ArrayList<String[]> getRelations();

    /**skriver til kilde umiddelbart. server er på vei ned. Hvis datakilden ikke støtter skriving implementeres denne tom.*/
    public void immediateWrite() throws IOException;
}

/*
 * GraphLayout.java
 *
 * Created on 23. april 2005, 11:05
 */

package genetuc_kb_grensesnitt.server;

import EDU.auburn.VGJ.algorithm.GraphUpdate;
import EDU.auburn.VGJ.algorithm.shawn.Spring;
import EDU.auburn.VGJ.graph.Graph;
import EDU.auburn.VGJ.util.DPoint;
import genetuc_kb_grensesnitt.Logger;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.HashMap;
import java.util.HashSet;

/**
 *
 * @author Fredrik
 */
public class GraphLayout {

    /**konstant for å skalere opp posisjoner fra spring algoritme*/
    final static int SCALE = 16;
    /**konstant for å flytte hele grafen slik at den blir midtstilt i gui-et*/
    final static int OFFSET = 250;

    /**Metoden tar inn en liste med relasjoner som konverteres til en Graph som Spring algoritmen tar inn.
     *@return Map med konseptnavn og posisjoner*/
    public static HashMap<String,int[]> getPositionsForConcepts(ArrayList<String[]> relations) {

        HashMap<String,int[]> positions = new HashMap<String,int[]>(); // returvariabel som populeres med posisjoner etterhvert.
        final Graph g = new Graph(); //final slik at den kan brukes i den indre Thread klassen nedenfor
        HashSet<String> concepts = new HashSet<String>(); //Set for å unngå å sjekke etter duplikater
        HashMap<String, Integer> nodes = new HashMap<String, Integer>(); // noder i Graph. Lages på grunnlag av konsepter i relations

        for(String[] r : relations) concepts.addAll(Arrays.asList(new String[]{r[0],r[2]})); //alle konsepter fra relations puttes inn i concepts

        Logger.p(Logger.INFO,"Antall concepter og relasjoner som skal plasseres ut: %s og %s",concepts.size(), relations.size());

        //lager noder i grafen
        Object[] obj = concepts.toArray();
        for(Object o : concepts.toArray())
            nodes.put((String)o,g.insertNode());

        //lager kanter i grafen
        for(String[] r : relations)
            g.insertEdge(nodes.get(r[0]),nodes.get(r[2]));

        //kjører spring algoritme på graf. Kjøres i egen tråd siden den kan ta noen sekunder. Eller mange timer (in which case den blir drept).
        final GraphUpdate gu = new GraphUpdateDummy();
```

```
        Logger.p(Logger.INFO,"Kjører spring-algorithme på graf.");
        Thread t = new Thread(){
            public void run(){
                Spring s = new Spring();
                Logger.p(Logger.DEBUG, "Returverdi fra spring-algoritme: %s", s.compute(g, gu));
            }
        };
        t.start();

        //venter på spring algoritmen, kverker den hvis den ikke er ferdig på 7000 millisekunder
        try{t.join(7000);}
        catch(InterruptedException ie){
            Logger.p(Logger.DEBUG, "Hoved-tråd ble avbrutt mens den ventet på Spring-tråd");
            RMIClassImpl.messageFIFOQueue.offer("Algoritmen som skulle plassere ut konsepter brukte for lang tid. Prøv med kortere maks
distanse og/eller søk etter konsepter som ikke er veldig sentrale");
        }
        if (t.isAlive()){
            Logger.p(Logger.INFO,"Tråden som kjører spring-algoritmen er for treg. Stopper den.");
            RMIClassImpl.messageFIFOQueue.offer("Spring algoritmen (for å plassere ut grafen) brukte for lang tid og er derfor stoppet. \nFor å la
algoritmen kjøre ferdig, prøv med kortere lengde og/eller mindre sentralt konsept og/eller mindre indeksintervall.");
            t.stop(); //TODO Thread.stop() er deprecated. Men ser ikke noe alternativ.
        } else
            Logger.p(Logger.INFO,"Ferdig med spring-algoritme.");

        //henter ut posisjoner
        for(String conceptName : nodes.keySet()) { //her blir dummyen ignorert, men den har satt sitt preg...
            DPoint dp = g.getNodeFromIndex(nodes.get(conceptName)).getPosition();
            positions.put(conceptName, new int[]{(int)dp.x*SCALE+OFFSET,(int)dp.y*SCALE+OFFSET});
        }
        return positions;
    }
}

package genetuc_kb_grensesnitt.server;

import EDU.auburn.VGJ.algorithm.GraphUpdate;
import EDU.auburn.VGJ.graph.Node;
import EDU.auburn.VGJ.util.DRect;
import java.awt.Frame;

/**Skall-klasse siden Spring implementasjonen krever en slik (den brukes egentlig for å oppdatere grafikk (i et annet
 * system). Som vi ikke skal gjøre i denne settingen). */

public class GraphUpdateDummy implements GraphUpdate{
    Node dummyNode = new Node();
    DRect dummyDRect = new DRect(0d,0d,100d,100d);
    Frame dummyFrame = new Frame();

    public GraphUpdateDummy() {
    }

    public void scale(double scaleval) {
    }

    public void update(boolean adjust_bounds) {
    }

    public void center() {
    }

    public double getVSpacing() {
        return 0d;
    }

    public EDU.auburn.VGJ.graph.Node getSelectedNode() {
        return dummyNode;
    }

    public EDU.auburn.VGJ.util.DRect windowRect() {
        return dummyDRect;
    }

    public java.awt.Frame getFrame() {
        return dummyFrame;
    }

    public double getHSpacing() {
        return 0d;
    }
}
```

```
package genetuc_kb_grensesnitt.client;

import genetuc_kb_grensesnitt.TucGuiException;
import java.io.IOException;
import java.rmi.RemoteException;
import javax.swing.JOptionPane;
import javax.swing.UIManager;

/**Klassen representerer hoved-vinduet til klient-side gui-et.
 */
public class GUIJFrame extends javax.swing.JFrame {
    /**GUIJFrame trenger å bruke ClientGui, derfor en referanse her*/
    ClientGUI parent;

    public GUIJFrame(ClientGUI parent) {
        try{
            UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
            initComponents();
            this.parent =  parent;
        }catch(Exception e){e.printStackTrace();}
    }

    private void initComponents() {//GEN-BEGIN:initComponents
        jPopupMenu1 = new javax.swing.JPopupMenu();
        jMenuItem2 = new javax.swing.JMenuItem();
        jCheckBoxMenuItem1 = new javax.swing.JCheckBoxMenuItem();
        jMenu2 = new javax.swing.JMenu();
        jRadioButtonMenuItem1 = new javax.swing.JRadioButtonMenuItem();
        jSVGCanvas1 = new org.apache.batik.swing.JSVGCanvas();
        jPanel1 = new javax.swing.JPanel();
        jPanel2 = new javax.swing.JPanel();
        jToggleButton1 = new javax.swing.JToggleButton();
        jToggleButton2 = new javax.swing.JToggleButton();
        jToggleButton3 = new javax.swing.JToggleButton();
        jTextField1 = new javax.swing.JTextField();
        jButton1 = new javax.swing.JButton();
        jTextField2 = new javax.swing.JTextField();
        jTextField3 = new javax.swing.JTextField();
        jTextField4 = new javax.swing.JTextField();
        jMenuBar2 = new javax.swing.JMenuBar();
        jMenu4 = new javax.swing.JMenu();
        jMenuItem4 = new javax.swing.JMenuItem();
        jMenuItem5 = new javax.swing.JMenuItem();
        jMenu1 = new javax.swing.JMenu();
        jMenuItem3 = new javax.swing.JMenuItem();
        seDom = new javax.swing.JMenuItem();
        jMenuItem1 = new javax.swing.JMenuItem();

        jPopupMenu1.setComponentPopupMenu(jPopupMenu1);
        jMenuItem2.setText("Itemkj");
        jMenuItem2.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jMenuItem2ActionPerformed(evt);
            }
        });

        jPopupMenu1.add(jMenuItem2);

        jCheckBoxMenuItem1.setText("CheckBox");
        jPopupMenu1.add(jCheckBoxMenuItem1);

        jMenu2.setText("Menu");
        jRadioButtonMenuItem1.setText("RadioButton");
        jMenu2.add(jRadioButtonMenuItem1);

        jPopupMenu1.add(jMenu2);

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        setTitle("TucGui 1.0");
        setResizable(false);
        jSVGCanvas1.setPreferredSize(new java.awt.Dimension(500, 500));
        jSVGCanvas1.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                jSVGCanvas1MouseClicked(evt);
            }
        });

        getContentPane().add(jSVGCanvas1, java.awt.BorderLayout.CENTER);
```

```
jPanel1.setLayout(new org.netbeans.lib.awtextra.AbsoluteLayout());

jPanel1.setBorder(new javax.swing.border.LineBorder(new java.awt.Color(0, 0, 0)));
jPanel1.setInheritsPopupMenu(true);
jPanel1.setPreferredSize(new java.awt.Dimension(480, 90));
jPanel2.setBorder(new javax.swing.border.TitledBorder("modus"));
jToggleButton1.setSelected(true);
jToggleButton1.setText("konsept");
jToggleButton1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jToggleButton1ActionPerformed(evt);
    }
});

jPanel2.add(jToggleButton1);

jToggleButton2.setText("relasjon");
jToggleButton2.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jToggleButton2ActionPerformed(evt);
    }
});

jPanel2.add(jToggleButton2);

jToggleButton3.setText("slett");
jToggleButton3.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jToggleButton3ActionPerformed(evt);
    }
});

jPanel2.add(jToggleButton3);

jPanel1.add(jPanel2, new org.netbeans.lib.awtextra.AbsoluteConstraints(10, 10, -1, 70));

jTextField1.setText("skriv konsept her");
jPanel1.add(jTextField1, new org.netbeans.lib.awtextra.AbsoluteConstraints(250, 20, -1, -1));

jButton1.setText("hent konsept");
jButton1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton1ActionPerformed(evt);
    }
});

jPanel1.add(jButton1, new org.netbeans.lib.awtextra.AbsoluteConstraints(380, 50, -1, -1));

jTextField2.setText("0");
jTextField2.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jTextField2ActionPerformed(evt);
    }
});

jPanel1.add(jTextField2, new org.netbeans.lib.awtextra.AbsoluteConstraints(390, 20, 40, -1));

jTextField3.setText("50");
jPanel1.add(jTextField3, new org.netbeans.lib.awtextra.AbsoluteConstraints(440, 20, 40, -1));

jTextField4.setText("2");
jPanel1.add(jTextField4, new org.netbeans.lib.awtextra.AbsoluteConstraints(350, 20, 30, -1));

getContentPane().add(jPanel1, java.awt.BorderLayout.NORTH);

jMenu4.setText("Operasjoner");
jMenu4.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jMenu4ActionPerformed(evt);
    }
});

jMenuItem4.setText("koble til server");
jMenuItem4.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jMenuItem4ActionPerformed(evt);
    }
});

jMenu4.add(jMenuItem4);
```

```
    jMenuItem5.setText("Avslutte");
    jMenuItem5.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            jMenuItem5ActionPerformed(evt);
        }
    });

    jMenu4.add(jMenuItem5);

    jMenuBar2.add(jMenu4);

    jMenu1.setLabel("Hjelp");
    jMenu1.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            jMenu1ActionPerformed(evt);
        }
    });

    jMenuItem3.setText("se hjelpetekstfil");
    jMenu1.add(jMenuItem3);

    seDom.setText("Se DOM");
    seDom.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            seDomActionPerformed(evt);
        }
    });

    jMenu1.add(seDom);

    jMenuItem1.setText("Om TucGui");
    jMenu1.add(jMenuItem1);

    jMenuBar2.add(jMenu1);

    setJMenuBar(jMenuBar2);

    pack();
}//GEN-END:initComponents

private void jMenu1ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_jMenu1ActionPerformed
    // TODO add your handling code here:
}//GEN-LAST:event_jMenu1ActionPerformed

private void jMenuItem2ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_jMenuItem2ActionPerformed
    // TODO add your handling code here:
}//GEN-LAST:event_jMenuItem2ActionPerformed

private void jTextField2ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_jTextField2ActionPerformed
    // TODO add your handling code here:
}//GEN-LAST:event_jTextField2ActionPerformed

/**Kalles når menyvalg for "se dom" aktiveres*/
private void seDomActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_seDomActionPerformed
    org.apache.batik.util.gui.DOMViewer dw = new org.apache.batik.util.gui.DOMViewer();
    dw.setDocument(jSVGCanvas1.getSVGDocument());
    dw.pack();
    dw.setVisible(true);
}//GEN-LAST:event_seDomActionPerformed

/**Kalles når menyvalg for "avslutt" aktiveres*/
private void jMenuItem5ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_jMenuItem5ActionPerformed
    System.exit(0);
}//GEN-LAST:event_jMenuItem5ActionPerformed

/**Kalles når menyvalg for "koble til server" aktiveres*/
private void jMenuItem4ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_jMenuItem4ActionPerformed
    new ClientConfigForm(parent).setVisible(true);
}//GEN-LAST:event_jMenuItem4ActionPerformed

private void jMenu4ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_jMenu4ActionPerformed

}//GEN-LAST:event_jMenu4ActionPerformed

/**Returnerer de valg som er gjort mhp maxDistance, indexFrom og indexTo. Disse brukes både når knapp for å hente konsept
 *trykke og når man dobbelklikker et konsept
 */
public int[] getConfigNumbers(){
    return new int[]{
```

```
            Integer.parseInt(jTextField4.getText()), //maxDist
                Integer.parseInt(jTextField2.getText()), //index start
                Integer.parseInt(jTextField3.getText()) //index end
        };
    }

    /**Kalles når knapp for "hent konsept" trykkes*/
    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_jButton1ActionPerformed
        int[] nums = getConfigNumbers();
        try{parent.displayConcept(jTextField1.getText(), nums[0], nums[1], nums[2]);} catch(NullPointerException npe){
            JOptionPane.showMessageDialog(this,"NullPointerException, mest sannsynlig fordi kobling til server ikke er opprettet, gjør det via meny.","NullPointerException",JOptionPane.ERROR_MESSAGE);
            return;
        } catch(RemoteException re){
            JOptionPane.showMessageDialog(this,"RemoteException ble kastet. Dvs at noe rart har skjedd. Se stacktrace (sett på java console for å se stacktracen)","RemoteException",JOptionPane.ERROR_MESSAGE);
            return;
        } catch(IOException re){
            JOptionPane.showMessageDialog(this,"IOException ble kastet. Dvs at noe rart har skjedd. Se stacktrace (sett på java console for å se stacktracen)","IOException",JOptionPane.ERROR_MESSAGE);
            return;
        }

        catch(TucGuiException re){
            JOptionPane.showMessageDialog(this,"TucGuiException ble kastet. Dvs at noe rart har skjedd. Se stacktrace (sett på java console for å se stacktracen)","TucGuiException",JOptionPane.ERROR_MESSAGE);
            return;
        }
    }//GEN-LAST:event_jButton1ActionPerformed

    /**Kalles når toggle-knapp for delete mode trykkes*/
    private void jToggleButton3ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_jToggleButton3ActionPerformed
        jToggleButton1.setSelected(false);
        jToggleButton2.setSelected(false);
        ClientGUI.mode=ClientGUI.MODE_DELETE;
    }//GEN-LAST:event_jToggleButton3ActionPerformed

    /**Kalles når toggle-knapp for relation mode trykkes*/
    private void jToggleButton2ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_jToggleButton2ActionPerformed
        jToggleButton1.setSelected(false);
        jToggleButton3.setSelected(false);
        ClientGUI.mode=ClientGUI.MODE_RELATION;
    }//GEN-LAST:event_jToggleButton2ActionPerformed

    /**Kalles når toggle-knapp for concept mode trykkes*/
    private void jToggleButton1ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_jToggleButton1ActionPerformed
        jToggleButton2.setSelected(false);
        jToggleButton3.setSelected(false);
        ClientGUI.mode=ClientGUI.MODE_CONCEPT;
    }//GEN-LAST:event_jToggleButton1ActionPerformed

    private void jSVGCanvas1MouseClicked(java.awt.event.MouseEvent evt) {//GEN-FIRST:event_jSVGCanvas1MouseClicked

    }//GEN-LAST:event_jSVGCanvas1MouseClicked

    public org.apache.batik.swing.JSVGCanvas getJSVGCanvas1() {
        return jSVGCanvas1;
    }

    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JButton jButton1;
    private javax.swing.JCheckBoxMenuItem jCheckBoxMenuItem1;
    private javax.swing.JMenu jMenu1;
    private javax.swing.JMenu jMenu2;
    private javax.swing.JMenu jMenu4;
    private javax.swing.JMenuBar jMenuBar2;
    private javax.swing.JMenuItem jMenuItem1;
    private javax.swing.JMenuItem jMenuItem2;
    private javax.swing.JMenuItem jMenuItem3;
    private javax.swing.JMenuItem jMenuItem4;
    private javax.swing.JMenuItem jMenuItem5;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JPopupMenu jPopupMenu1;
    private javax.swing.JRadioButtonMenuItem jRadioButtonMenuItem1;
    private org.apache.batik.swing.JSVGCanvas jSVGCanvas1;
    private javax.swing.JTextField jTextField1;
    private javax.swing.JTextField jTextField2;
    private javax.swing.JTextField jTextField3;
    private javax.swing.JTextField jTextField4;
```

```
    private javax.swing.JToggleButton jToggleButton1;
    private javax.swing.JToggleButton jToggleButton2;
    private javax.swing.JToggleButton jToggleButton3;
    private javax.swing.JMenuItem seDom;
    // End of variables declaration//GEN-END:variables

}

package genetuc_kb_grensesnitt.server;

import genetuc_kb_grensesnitt.Logger;
import java.io.*;
import java.util.ArrayList;
import java.util.HashSet;
import java.util.regex.*;

/**
 * Klassen implementerer datakilde interfacet mot prolog-filer som tradisjonelt er kunnskapskilde i GeneTuc.
 *
 */
public class KBFile implements Datasource{

    /** Liste med relasjoner i form av tripletter i en string[3]; konsept, relasjon, konsept*/
    ArrayList<String[]> relations = new ArrayList<String[]>();
    /** Inneholder alle konsepter i denne filen. Bruker Set siden det ikke inneholder duplikater, gjør innmating litt enklere*/
    HashSet<String> concepts = new HashSet<String>();
    /**String som holder på tekstfilen.*/
    String fileBuffer;
    /**File som peker på filen denne KBFilen representerer*/
    File sourceFile;
    /**Tråd som holder øye med kildefilen og sier ifra dersom kildefilen endres utenom TucGui*/
    FileMonitor fileMonitor;
    /** teller opp antall endringer fra klient, brukes av writeToDisk*/
    int deltaCounter = 0;

    /** Separerer tag for tv_templ som vist i gui, og verb*/
    private static final String VERB_TEMPLATE_SEPARATOR = "##";
    /** blir puttet foran verb før det vises i gui, slik at det er mulig å skille relasjoner på triplett form og relasjoner på standard prolog predikat form*/
    private static final String VERB_TEMPLATE_PREFIX = "tv_templ" + VERB_TEMPLATE_SEPARATOR;

    /*Lager ny instans av KBFile.
     *@param filename hvilken prolog (.pl) fil denne KBFile skal bruke
     */
    public KBFile(String filename) { //TODO: med små modifikasjoner kan denne bli generell
        try{
            sourceFile = new File(filename);
            readAndParse();
            fileMonitor = new FileMonitor(); //starter fil monitorering etter at filen er lest og parset.
            fileMonitor.start();
        }    catch(FileNotFoundException    fnfe){Logger.p(Logger.CRITICAL,"KBFile:    Fant    ikke    filen    %s",filename);}    catch(IOException
ioe){Logger.p(Logger.CRITICAL,"IOException: %s",ioe.toString());}
    }

    /**leser inn alle aktive (ikke kommentert vekk) relasjoner fra filen
     * denne klassen representerer og populerer concepts og relations datastrukturene.
     * Antakelse: flerlinjet kommentar starter aldri på samme linje som noe som skal parses.
     */
    private void parseFileBuffer() throws IOException{
        Logger.p(Logger.INFO,"Parser fil %s.",sourceFile.toString());
        String buffer = "", temp;
        boolean multilineComment = false;

        //laster inn fil og fjerner alle kommentarer
        BufferedReader sr = new BufferedReader(new StringReader(fileBuffer));
        temp = sr.readLine();
        while (temp != null){

            /*antakelse: flerlinjet kommentar starter aldri på samme linje som noe som skal parses.
             f.eks. "a er_en b /* flerlinjet kommenar starter..." da vil de neste linjene slippe igjennom og kunne skape havoc*/

            if(temp.startsWith("/*")) multilineComment = true;
            if(temp.endsWith("*/")) multilineComment = false;
            //gjør noe kun hvis vi ikke er i en flerlinjet kommentar-blokk og da bare hvis linjen ikke utelukkende er kommentar
            else if(!multilineComment) buffer += (temp.matches("^[[:blank:]]*%.*"))?"":temp+"\n"; //TODO: tegn grafisk hvordan tekstfilen "prunes"
            temp = sr.readLine();
        }

        Matcher m = getPatternForRelation("\\w*?","\\w*?","\\w*?").matcher(buffer);

        relations.clear();
```

```
        while(m.find()){
            relations.add(new String[]{m.group(1),m.group(2),m.group(3)});
            concepts.add(m.group(1));
            concepts.add(m.group(3)); //siden concepts er et Set gjør det ikke noe om ett konsept addes flere ganger (som det antakeligvis vil)
        }

        Matcher m_tvTempl = getTvTemplPattern("\\w+","\\w+","\\w+").matcher(buffer);

        while(m_tvTempl.find()){
            relations.add(new                    String[]{m_tvTempl.group(2),VERB_TEMPLATE_PREFIX+m_tvTempl.group(1),m_tvTempl.group(3)});
//verb_template_prefix må legges til for å gjenkjenne denne relasjonen senere. VERB_SEPARATOR antaes å ikke opptre andre steder enn
som deletegn her
            concepts.add(m_tvTempl.group(2));
            concepts.add(m_tvTempl.group(3)); //siden concepts er et Set gjør det ikke noe om ett konsept addes flere ganger (som det antakeligvis
vil)
        }
        Logger.p(Logger.INFO,"Antall konsepter lest: %s. Antall relasjoner lest: %s",concepts.size(), relations.size());
    }

    /**Metode som returnerer de relasjoner som er parset inn fra fil*/
    public ArrayList<String[]> getRelations(){ return relations; }

    /**Metode som returnerer regexp Pattern for å søke etter tv_templ. Pattern har 4 grupper: 1,2,3,4 = verb,subjekt,objekt,kommentar
     *@param verb verbet i tv_templ(verb,subject,object)
     *@param subject subject i tv_templ(verb,subject,object)
     *@param object object i tv_templ(verb,subject,object)
     */
    private Pattern getTvTemplPattern(String verb, String subject, String object){
        return Pattern.compile("(?m)^\\s*tv_templ\\(\\s*("+verb+")\\s*,\\s*("+subject+")\\s*,\\s*("+object+")\\s*\\)\\s*\\.(.*)(?=$)");
    }

    /**Metoden renamer gjeldende fil til [navn][tall].pl hvor tall er det neste tall som det ikke allerede eksisterer fil for.
     *Skriver og tar backup for hver 10-ende kall til denne metoden.
     */
    public void writeToDisk() throws IOException{
        deltaCounter++;
        if(deltaCounter % 10 == 0){ //skriver til fil og lager backup kun for hver 10-ende kall til writeToDisk
            Logger.p(Logger.INFO,"Skriver til disk. (%s)",sourceFile.toString());
            //parseFileBuffer(); //repopulerer triplett-liste også. Liste og fil er konsistent.

            int counter = 1;
            File backupFile = new File(sourceFile.getAbsoluteFile().toString().replace(".",counter+"."));
            while(backupFile.exists()){
                counter++;
                backupFile = new File(sourceFile.getAbsoluteFile().toString().replace(".",counter+"."));
            }
            //skriver til backup
            FileReader raf = new FileReader(sourceFile);
            PrintWriter pwBackup = new PrintWriter(backupFile);
            char[] b = new char[999999]; //TODO: fix dette...
            int numBytes = raf.read(b);

            //gjør kun backup hvis filen inneholder noe
            if(numBytes!=-1){
                pwBackup.write(b,0,numBytes);
                pwBackup.close();
            }

            //skriver ut nye ting
            PrintWriter pwSource = new PrintWriter(sourceFile);
            pwSource.write(fileBuffer);
            pwSource.close();
            fileMonitor.setLastModified(System.currentTimeMillis());
        }
    }

    /**sletter også alle relasjoner konseptet er med i*/
    public void deleteConcept(String conceptName) throws IOException {
        Logger.p(Logger.INFO,"Sletter konsept %s.",conceptName);
        Pattern p1 = getPatternForRelation(conceptName, "\\w*?","\\w*?");
        Pattern p2 = getPatternForRelation("\\w*?","\\w*?",conceptName);
        Pattern tvTempl_p1 = getTvTemplPattern("\\w+", conceptName,"\\w+");
        Pattern tvTempl_p2 = getTvTemplPattern("\\w+", "\\w+", conceptName);

        String s;
        long startFP, endFP;

        fileBuffer = p1.matcher(fileBuffer).replaceAll(" ");
        fileBuffer = p2.matcher(fileBuffer).replaceAll(" ");
        fileBuffer = tvTempl_p1.matcher(fileBuffer).replaceAll(" ");
```

```
        fileBuffer = tvTempl_p2.matcher(fileBuffer).replaceAll(" ");

        writeToDisk();
    }
    /**Sletter en relasjon*/
    public void deleteRelation(String conceptFrom, String conceptTo, String relationName) throws IOException{
        Logger.p(Logger.INFO,"Sletter relasjon %s fra %s til %s.",relationName,conceptFrom, conceptTo);
        if(relationName.split("¤").length == 2)
            fileBuffer = fileBuffer.replaceAll(getTvTemplPattern(relationName.split("¤")[1], conceptFrom, conceptTo).toString()," "); //TODO: Ok at
konsepter kan forsvinne på denne måten også? Vises det i svg-en?
        else
            fileBuffer = fileBuffer.replaceAll(getPatternForRelation(conceptFrom, relationName, conceptTo).toString()," "); //TODO: Ok at konsepter
kan forsvinne på denne måten også? Vises det i svg-en?
        writeToDisk();
    }
    /**Endrer navn på et konsept. Konsepter antaes å være unike*/
    public void renameConcept(String oldConceptName, String newConceptName) throws IOException{
        Logger.p(Logger.INFO,"Omdøper konsept %s til %s.", oldConceptName, newConceptName);
        fileBuffer = fileBuffer.replaceAll("\\b"+oldConceptName+"\\b",newConceptName); //\b er "word boundary"
        writeToDisk();
    }

    /**Endrer navn på en relasjon*/
    public void renameRelation(String conceptFrom, String conceptTo, String oldRelationName, String newRelationName) throws IOException{
        Pattern p;
        boolean tv_templ = false;
        if(oldRelationName.split("¤").length == newRelationName.split("¤").length){
            if(oldRelationName.split("¤").length == 2){
                p = getTvTemplPattern(oldRelationName.split(VERB_TEMPLATE_SEPARATOR)[1], conceptFrom,conceptTo); //i tilfelle tv_templat
                tv_templ = true;
            } else
                p = getPatternForRelation(conceptFrom, oldRelationName, conceptTo); //i tilfelle "konsept relasjon konsept"
            Matcher m = p.matcher(fileBuffer);
            if(m.find()){
                Logger.p(Logger.INFO,this.sourceFile.getName()+" inneholder relasjon. Omdøper relasjon %s mellom %s og %s om til
%s.",oldRelationName, conceptFrom, conceptTo, newRelationName);
                String relationPattern = p.toString();
                String commentForRelation = m.group(4); //group 4 er kommentaren
                if(tv_templ)
                    fileBuffer                                                                                                                    =
fileBuffer.replaceAll(relationPattern,"tv_templ("+newRelationName.split("¤")[1]+","+conceptFrom+","+conceptTo+")."+commentForRelation);
                else
                    fileBuffer = fileBuffer.replaceAll(relationPattern,conceptFrom+" "+newRelationName+" "+conceptTo+"."+commentForRelation);
                writeToDisk();
            }
        }else{
            Controller.rmiInstance.messageFIFOQueue.offer("en relasjon må renames til samme type relasjon hvis tag er inkludert. tv_templ tag:
\"tv_templ¤verb\"");
        }
    }

    /** Returnerer regexp Pattern for relasjon på form "A eren B" [konsept] [relasjon] [konsept]*/
    private Pattern getPatternForRelation(String conceptFrom, String relation, String conceptTo){
        String regexp = "(?m)^\\s*("+conceptFrom+")\\s+("+relation+")\\s+("+conceptTo+")\\s*\\.(.*)(?=$)"; //start-whitespace-alfanum-whitespace-
alfanum-whitespace-alfanum-whitespace-hvasomhelst-end
        return Pattern.compile(regexp);
    }

    /**Oppretter en ny relasjon*/
    public void createRelation(String conceptFrom, String relation, String conceptTo) throws IOException{
        Logger.p(Logger.INFO,"Ny relasjon %s mellom %s og %s lagres i %s.", relation,conceptFrom,conceptTo,sourceFile.getName());
        if(relation.split("¤").length == 2)
            fileBuffer = fileBuffer + "\ntv_templ("+relation.split(VERB_TEMPLATE_SEPARATOR)[1]+","+conceptFrom+","+conceptTo+"). %lagt til av
GenGui "+(new java.util.Date(System.currentTimeMillis())).toString();
        else
            fileBuffer = fileBuffer + "\n"+conceptFrom+" "+relation+" "+conceptTo+". %lagt til av GenGui "+(new
java.util.Date(System.currentTimeMillis())).toString();
        writeToDisk();
    }

    /**Leser fra kildefilen og henter ut relasjoner, som puttes i bufferet*/
    private void readAndParse() throws FileNotFoundException, IOException{
        RandomAccessFile fr = new RandomAccessFile(sourceFile,"r");
        byte[] byteBuffer = new byte[(int)fr.length()];
        fr.seek(0);
        int bytes = fr.read(byteBuffer);
        fileBuffer = new String(byteBuffer);
        parseFileBuffer();
    }
```

```
    /**Returnerer konsepter som er hentet ut fra kildefilen.*/
    public HashSet<String> getConcepts(){ return concepts; }

    /**Klasse som kjører i egen tråd og følger med på om kildefilen endres av ekstern prosess. I så fall reparses kildefilen.*/
    class FileMonitor extends Thread{

        boolean run=true;
        long lastModifiedByTucGui = System.currentTimeMillis();
        long lastCheck = System.currentTimeMillis();

        /**Starter tråden*/
        public void run(){
            while(run){
                try{
                    if(sourceFile.lastModified() > lastModifiedByTucGui & sourceFile.lastModified() > lastCheck){
                        Logger.p(Logger.INFO,"Kildefil %s har blitt endret av andre enn brukere av TucGui. Leser og parser filen på nytt samt underretter
evt online brukere.", sourceFile.getName());

                        Controller.rmiInstance.messageFIFOQueue.offer("Kildefil "+sourceFile.getName()+" er endret av ekstern bruker. Leser filen på
nytt. (kan ta litt tid).");
                        lastCheck = System.currentTimeMillis();
                        readAndParse();
                    }
                    synchronized(this){ wait(60000);}; //venter 60 sekunder
                }catch(IOException ioe){ioe.printStackTrace();} catch(InterruptedException ie){ie.printStackTrace();}
            }
        }
        /**Sier ifra at tråden skal slutte å kjøre*/
        public void stopRunning(){run=false;}
        /**Setter når kildefilen sist ble endret av TucGui selv, slik at disse endringene ikke oppfattes som endring av ekstern prosess*/
        public void setLastModified(long millis){
            lastModifiedByTucGui = millis;
        }
    }

    /**Metoden skriver buffer til disk umiddelbart. Brukes når programmet avsluttes og endringer trenger å skrives
     *umiddelbart
     */
    public void immediateWrite() throws IOException{
            Logger.p(Logger.INFO,"Skriver til disk. (%s)",sourceFile.toString());
            //parseFileBuffer(); //repopulerer triplett-liste også. Liste og fil er konsistent.

            int counter = 1;
            File backupFile = new File(sourceFile.getAbsoluteFile().toString().replace(".",counter+"."));
            while(backupFile.exists()){
                counter++;
                backupFile = new File(sourceFile.getAbsoluteFile().toString().replace(".",counter+"."));
            }
            //skriver til backup
            FileReader raf = new FileReader(sourceFile);
            PrintWriter pwBackup = new PrintWriter(backupFile);
            char[] b = new char[999999]; //999999 bør holde til lese inn hele filen.
            int numBytes = raf.read(b);

            //gjør kun backup hvis filen inneholder noe
            if(numBytes!=-1){
                pwBackup.write(b,0,numBytes);
                pwBackup.close();
            }

            //skriver ut nye ting
            PrintWriter pwSource = new PrintWriter(sourceFile);
            pwSource.write(fileBuffer);
            pwSource.close();
            fileMonitor.setLastModified(System.currentTimeMillis());
    }
}

package genetuc_kb_grensesnitt;

/** Klassen brukes for å ha en sentral plass hvor output fra server og klient (hver for seg) går via.
 * Tanken er at man lett kan implementere at tekst fra logg skal kunne sendes tilfeldig sted, f.eks. over nett/til en mail adresse etc.
 * En annen ting er at man har forskjellige nivåer av hvor mye som blir skrevet ut, f.eks. info, debug, kritisk.
 */
public class Logger {

    final public static int INFO = 1; //skal det utvides, bruk 2^n som tall. n er beskjednivå. slik at det ikke blir kluss med AND-ing
    final public static int DEBUG = 2;
    final public static int CRITICAL = 4;
    public static int printLevel = INFO; //standard printlevel. Kan settes fra hvor som helst.
```

```
    /**Metoden logger (per nå: til skjerm) hvis nivået beskjeden er på er ønsket.
     * @param type hvilket nivå beskjeden er (kritisk, info debug), bruk f.eks. Tools.INFO
     * @param printfString en streng som skal skrives ut, samme type strenger som System.printf tar
     * @param objs objekter som refereres i printfString
     */
    public static void p(int type, String printfString, Object ... objs){
        String prefix = "["+new java.util.Date(System.currentTimeMillis()).toString()+"] ";
        printfString = printfString+"\n";
        switch(type){
            case DEBUG:
                if((printLevel & DEBUG) == DEBUG) {
                    prefix += "DEBUG: ";
                    System.out.printf(prefix+printfString,objs);
                }
                break;
            case INFO:
                if((printLevel & INFO) == INFO){
                  prefix += "INFO: ";
                  System.out.printf(prefix+printfString,objs);
                }
                break;
            case CRITICAL:
                if((printLevel & CRITICAL) == CRITICAL){
                  prefix += "CRITICAL: ";
                  System.out.printf(prefix+printfString,objs);
                }
                break;
            default:
                prefix += "(ukjent meldingstype): ";
                System.out.printf(prefix+printfString,objs);
                break;
        }
    }

}

package genetuc_kb_grensesnitt.client;

import genetuc_kb_grensesnitt.Logger;
import genetuc_kb_grensesnitt.server.RMIClassInterface;
import javax.swing.JOptionPane;
import org.apache.batik.dom.svg.SVGOMGElement;
import org.apache.batik.dom.svg.SVGOMPathElement;
import org.apache.batik.dom.svg.SVGOMTextElement;
import org.apache.batik.dom.svg.SVGOMTransform;
import org.apache.batik.swing.JSVGCanvas;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.Text;
import org.w3c.dom.events.Event;
import org.w3c.dom.events.EventListener;
import org.w3c.dom.events.EventTarget;
import org.w3c.dom.events.MouseEvent;
import org.w3c.dom.svg.SVGDocument;
import org.w3c.dom.svg.SVGEllipseElement;
import org.w3c.dom.svg.SVGGElement;
import org.w3c.dom.svg.SVGLocatable;
import org.w3c.dom.svg.SVGPathElement;
import org.w3c.dom.svg.SVGPoint;
import org.w3c.dom.svg.SVGTextElement;

/**Klassen håndterer alle hendelser fra musen.
 */
public class MouseHandler implements EventListener{

    /**svg dokumentet som er lastet i canvasen i hovedvinduet*/
    SVGDocument svgDocument;
    /**klassen som lever på serversiden*/
    RMIClassInterface enRMIKlasse;
    /**canvasen som viser fram svg grafikk*/
    JSVGCanvas svgCanvas;
    ClientGUI cGUI;
    /**det element i svg dom-en som er mål for en hendelse (f.eks. klikker man på en ellipse med musen blir ellipsen target). Settes i starten av
handle metoden*/
    Element target;
    /**navnerom for genetuc*/
    String geneTucNS = "http://www.idi.ntnu.no/GeneTuc/";
    /**navnerom for svg*/
    String svgNS = "http://www.w3.org/2000/svg";
```

```
    //variable spesifikke for flytting av konsept
    /**flagg som indikerer om et konsept blir dratt*/
    boolean draggingConcept;
    /** koordinat for start-posisjon for dragning*/
    int dragStartX, dragStartY;
    /** ellipsen som blir dratt*/
    SVGEllipseElement draggedEllipse;

    //variable spesifikke for ny relasjon
    /**flagg som indikerer om en ny relasjon blir dratt*/
    boolean dragging;
    /**refererer til det konseptet relasjon blir dratt fra*/
    SVGEllipseElement relationFrom;
    /** den nye relasjonens strek*/
    SVGPathElement Path;
    /** tekst for ny relasjon*/
    SVGTextElement relationText;

/** Instansierer MouseHandler*/
    public MouseHandler(ClientGUI cGUI, RMIClassInterface enRMIKlasse, JSVGCanvas svgCanvas, SVGDocument svgDocument) {
        this.enRMIKlasse = enRMIKlasse;
        this.cGUI = cGUI;
        this.svgCanvas = svgCanvas;
        this.svgDocument = svgDocument;
    }

    /**Denne metoden blir kalt når en mus-hendelse skjer.
     * Metoden tar tilstandsflagg og modus-knappenes tilstand i betraktning.
     * Alle hendelsene mouseclick/drag/up/down ender opp her.
     */
    public void handleEvent(Event evt){
        try{

            MouseEvent mevt = (MouseEvent)evt; //gjør om event til MouseEvent siden den har noen metoder som trengs her
            target = (Element)mevt.getTarget();

            /*må finne koordinater som muspilen klikket på fra vindu-koord til kanvas-koord (siden kanvasen kan være
             pannet og zoomet) det er her det blir problemer dersom kanvas er skalert i utgangspunktet eller
             vinduet blir skalert av bruker. Derfor er vinduets størrelse låst.*/

            SVGPoint pt = svgDocument.getRootElement().getCurrentTranslate();
            float scale = svgDocument.getRootElement().getCurrentScale();
            int svgKoordX = (int)((mevt.getClientX()/scale - (int)(pt.getX()/scale)));
            int svgKoordY = (int)((mevt.getClientY()/scale - (int)(pt.getY()/scale)));

            //linjen under kan brukes til debugging. skriver bare hvis ikke mousemove, siden det ellers blir veldig mye skriving.
            //if(!mevt.getType().equals("mousemove")) Logger.p(Logger.DEBUG, "%s %s\n",target, mevt.getType());

            String eventType = mevt.getType();

            /**besluttningstreet under er slik:
             *mousedown?
             *  dobbelklikk på ellipse?
             *    hent nytt konsept.
             *  tekst-element?
             *    rename.
             *  concept-mode?
             *    drar ikke konsept og musned på ellipse?
             *      start konseptdrag.
             *    hvis klikker på bakgrunn
             *      lag nytt konsept.
             *  relation-mode?
             *    target er konsept?
             *      start relasjonsdrag.
             *  delete-mode?
             *    target er konsept?
             *      slett konsept.
             *    target er relasjon?
             *      slett relasjon
             *mouseup?
             *  concept-mode?
             *    konseptdrag?
             *      avslutt konseptdrag.
             *  relation-mode?
             *    relationdrag og gyldig mål?
             *      lag relasjon.
             *    relationdrag og ugyldig mål?
             *      avslutt relationdrag uten å lage relasjon.
             *mousemove?
```

```
         *  drar relasjon?
         *    oppdater grafikk.
         *  drar konsept?
         *    oppdater grafikk.
         */

        if(eventType.equals("mousedown")){
            if(target instanceof SVGEllipseElement & mevt.getDetail() == 2){
                int[] nums = cGUI.gjf.getConfigNumbers();
                cGUI.displayConcept(target.getParentNode().getChildNodes().item(1).getFirstChild().getNodeValue(),nums[0],nums[1],nums[2]);
            }
            //uansett mode: hvis det blir klikket på textfelt skal renaming skje
            else if(target instanceof SVGTextElement) renameEntity((SVGTextElement)target);
            else
                switch(ClientGUI.mode){

                    case ClientGUI.MODE_CONCEPT:
                        if(!draggingConcept & target instanceof SVGEllipseElement){

                            //starter å flytte konsept
                            dragStartX = svgKoordX;
                            dragStartY = svgKoordY;
                            draggedEllipse = (SVGEllipseElement)target; //TODO: søk på mevt.getTarget...
                            ((EventTarget)svgDocument).addEventListener("mousemove", this, false);
                            draggingConcept = true;

                        }else if(target.getAttribute("id").equals("bg")){ //TODO ikke hardkod bg, trengs NS?
                            SVGGElement conceptGroup = (SVGGElement)svgDocument.getElementById("concepts"); //TODO: sjekk at konsepter blir
lagt i concepts g-en
                            String conceptName = JOptionPane.showInputDialog(svgCanvas,"skriv inn konseptnavn...");
                            if(conceptName != null)
                                conceptGroup.appendChild(cGUI.newConcept(conceptName,svgKoordX,svgKoordY));
                        }
                        break;

                    case ClientGUI.MODE_RELATION:
                        if(target.getNodeName().equals("ellipse"))
                            startCreatingRelation();
                        break;

                    case ClientGUI.MODE_DELETE:
                        if(target instanceof SVGEllipseElement)
                            cGUI.deleteConcept((SVGGElement)target.getParentNode());
                        else if(target instanceof SVGPathElement)
                            cGUI.deleteRelation((SVGGElement)target.getParentNode());
                        break;
                }
        }

        if(eventType.equals("mouseup")){
            switch(ClientGUI.mode){

                case ClientGUI.MODE_CONCEPT:
                    if(draggingConcept){
                        cGUI.updateRelationPositions(draggedEllipse);
                        ((EventTarget)svgDocument).removeEventListener("mousemove", this, false);
                        draggingConcept = false;
                        draggedEllipse = null;
                    }
                    break;

                case ClientGUI.MODE_RELATION:
                    if(dragging & ((Node)target) != relationFrom & target.getNodeName().equals("ellipse")) //TODO: nødvendig å caste til Node?
                        finishCreatingRelation();
                    else if(dragging & (target == relationFrom | !(target instanceof SVGEllipseElement))){
                        Path.getParentNode().getParentNode().removeChild(Path.getParentNode());
                        Path = null;
                        dragging = false;
                        relationFrom = null;
                        Path = null;
                        relationText = null;
                    }
                    break;
            }
        }

        if(eventType.equals("mousemove")){
            if(dragging){
                int newX = svgKoordX;
                int newY = svgKoordY;
```

```
                Path.setAttributeNS(null, "d", Path.getAttributeNS(null,"d").replaceFirst("[\\S\\.]*?\\s[\\S\\.]*?$", newX+ " "+newY));
            } else if(draggingConcept){
                SVGGElement gElement = (SVGGElement)draggedEllipse.getParentNode();
                SVGOMTransform trans = new SVGOMTransform();
                trans.setTranslate(svgKoordX,svgKoordY);
                gElement.getTransform().getBaseVal().initialize(trans); //TODO: er dette beste måten?
                cGUI.updateRelationPositions(draggedEllipse);
            }
        }
    }catch(Exception e){
        e.printStackTrace();
    }
}

private void renameEntity(SVGTextElement textElement) throws Exception{
    String oldName = textElement.getChildNodes().item(0).getNodeValue();
    String newName = JOptionPane.showInputDialog(svgCanvas,"skriv inn nytt navn...",oldName);

    if (newName != null){

        textElement.getChildNodes().item(0).setNodeValue(newName); //oppdaterer tekst i gui

        if (((SVGGElement)textElement.getParentNode().getParentNode()).getId().equals("relations")) {
            //en relasjon renames
            String                                                  conceptToInternalID                                                 =
((SVGPathElement)((SVGGElement)textElement.getParentNode()).getChildNodes().item(0)).getAttributeNS(geneTucNS,"to");
            String                                                  conceptFromInternalID                                               =
((SVGOMPathElement)((SVGOMGElement)textElement.getParentNode()).getChildNodes().item(0)).getAttributeNS(geneTucNS,"from");
            String conceptTo = cGUI.getConceptName(conceptToInternalID);//index 1 skal være texten
            String conceptFrom = cGUI.getConceptName(conceptFromInternalID);
            enRMIKlasse.renameRelation(conceptFrom, conceptTo, oldName, newName);
        }else{
            //da må det være konsept som renames
            enRMIKlasse.renameConcept(oldName, newName);
        }
    }
}

/**Metode for å avslutte laging av relasjon.*/
private void finishCreatingRelation() throws Exception{
    //oppdaterer gui clientside
    String relationName = JOptionPane.showInputDialog(svgCanvas,"skriv inn navn på relasjon...");
    if(relationName == null){
        Path.getParentNode().getParentNode().removeChild(Path.getParentNode());
        Path = null;
        dragging = false;
        relationFrom = null;
        Path = null;
        relationText = null;
    }else{
        Path.setAttributeNS(geneTucNS,"to",target.getParentNode().getAttributes().getNamedItem("id").getNodeValue());
        cGUI.updateRelationPositions(relationFrom);
        cGUI.updateRelationPositions(((SVGEllipseElement)target));
        Text relation = svgDocument.createTextNode(relationName);
        dragging = false;
        ((EventTarget)svgDocument).removeEventListener("mousemove",this, false);
        relationText.appendChild(relation);
        Path = null;
        relationText = null;

        String conceptFrom = relationFrom.getNextSibling().getFirstChild().getNodeValue();
        String conceptTo = target.getNextSibling().getFirstChild().getNodeValue();
        enRMIKlasse.createRelation(conceptFrom, relation.getNodeValue(), conceptTo);
    }
}

private void startCreatingRelation() throws Exception{
    dragging = true;
    relationFrom = (SVGEllipseElement)target;
    SVGGElement relGroup = (SVGGElement)svgDocument.getElementById("relations");

    //lager relasjonslinje
    SVGGElement relationContainer = (SVGGElement)svgDocument.createElementNS(svgNS,"g");
    Path = (SVGPathElement)svgDocument.createElementNS(svgNS,"path");
    relationText = (SVGTextElement)svgDocument.createElementNS(svgNS,"text");
    Path.setAttributeNS(geneTucNS,"from",relationFrom.getParentNode().getAttributes().getNamedItem("id").getNodeValue());     //id'ene     til
gruppene som inneholder ellipsene som er med i relasjonen
    Path.setAttributeNS(null,"marker-end","url(#triangel)");
    float x = ((SVGLocatable)relationFrom).getCTM().getE();
    float y = ((SVGLocatable)relationFrom).getCTM().getF();
```

```
        Path.setAttribute("d","M "+x+" "+y+" L "+x+" "+y); //må bare sette noe slik at updateRelations alg har noe å ta tak i
        relationContainer.appendChild(Path);
        relationContainer.appendChild(relationText);
        relGroup.appendChild(relationContainer);
        ((EventTarget)svgDocument).addEventListener("mousemove",this,false);
    }
}

package genetuc_kb_grensesnitt;

/**Enkel klasse for å indikere at det ikke finnes en datakildehåndterer til ønsket datakilde
 */
public class NoDatasourceHandlerException extends Exception{

    public NoDatasourceHandlerException(String s) {
        super(s);
    }

}

package genetuc_kb_grensesnitt.server;

import genetuc_kb_grensesnitt.Logger;
import genetuc_kb_grensesnitt.Tools;
import genetuc_kb_grensesnitt.TucGuiException;
import java.io.IOException;
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
import java.util.ArrayList;

import java.util.HashMap;
import java.util.List;

/** Klassen implementerer RMIClassInterface. Brukes som inngangsport til serveren fra klient.
 */
public class RMIClassImpl extends UnicastRemoteObject implements RMIClassInterface{

    public RMIClassImpl() throws RemoteException{
        super();
        Controller.rmiInstance = this;
    }
    /**Se RMIClassInterface*/
    public String getAnSVG(String conceptName, int maxDistance, int fromIndex, int toIndex) throws RemoteException, TucGuiException {
//TODO: refaktor: skift navn
        Logger.p(Logger.INFO,"Server ble bedt om å sende SVG for konsept %s", conceptName);
        String ret = "ERROR";
        ArrayList<String[]> allRelations = new ArrayList<String[]>(), collectedRelations = new ArrayList<String[]>();

        //henter ut relasjoner
        for(Datasource ds : Controller.getDatasources()) allRelations.addAll(ds.getRelations());
        collectedRelations = getRelevantRelations(conceptName, allRelations, maxDistance, fromIndex, toIndex);
        if(collectedRelations.size()>0){
            HashMap<String,int[]> hm = GraphLayout.getPositionsForConcepts(collectedRelations);
            ret = Tools.generateSVGTextualDataFromTriplets(collectedRelations, hm);
            //catch(Exception e){e.printStackTrace();}
            Logger.p(Logger.INFO, "Returnerer svg tekst. Tekstlengde: %s", ret.length());
            return ret;
        }else {
            messageFIFOQueue.offer("Ønsket konsept ga ingen relasjoner. Returnerer tomt svg-dokument.");
            return Tools.generateSVGTextualDataFromTriplets(new ArrayList<String[]>(), new HashMap<String,int[]>());
        }
    }

    /**Metoden filtrerer mhp maxDistance, indexFrom og  indexTo for at graf algoritmen skal bli ferdig innen rimelig tid.*/
    public ArrayList<String[]> getRelevantRelations(String startConcept, ArrayList<String[]> allRelations, int maxDistance, int fromIndex, int
toIndex){

        ArrayList<String> conceptsToBeChecked = new ArrayList<String>();
        ArrayList<String> conceptsToBeCheckedNext = new ArrayList<String>();
        ArrayList<String> checkedConcepts = new ArrayList<String>();
        conceptsToBeChecked.add(startConcept);
        ArrayList<String[]> collectedRelations = new ArrayList<String[]>();

        for(int i=1; i <= maxDistance;i++){
            while(!conceptsToBeChecked.isEmpty()){
                String currentConcept = conceptsToBeChecked.remove(0); //henter ut et konsept
                checkedConcepts.add(currentConcept);
                for(String[] relation : allRelations){
                    if (relation[0].equals(currentConcept) | relation[2].equals(currentConcept)) { //hvis konseptet er med i relasjonen, ta med relasjonen
```

```
                    if(!collectedRelations.contains(relation)) collectedRelations.add(relation); //ta vare på relasjon dersom den ikke allerede er tatt vare på
                    if(!checkedConcepts.contains(relation[0])) conceptsToBeCheckedNext.add(relation[0]); //sjekk om det er konsept som må sjekkes videre
                    if(!checkedConcepts.contains(relation[2])) conceptsToBeCheckedNext.add(relation[2]); //sjekk om det er konsept som må sjekkes videre
                }
            }
        }
        conceptsToBeChecked.addAll(conceptsToBeCheckedNext);
        conceptsToBeCheckedNext.clear();
    }
    List list;
    try{
        list = collectedRelations.subList(fromIndex,toIndex);
        list.add(collectedRelations.get(toIndex));  //inkluderer toIndex (subList ekskluderer index_til);
    }
    catch(IndexOutOfBoundsException iobe){
        if (fromIndex < 0)fromIndex=0;
        if (toIndex > collectedRelations.size()-1) toIndex = Math.max(collectedRelations.size() - 1,0); //Math.max; i tilfelle collected relations er tom
        list = collectedRelations.subList(fromIndex,toIndex);
        if(collectedRelations.size() != 0) list.add(collectedRelations.get(toIndex));  //inkluderer toIndex (subList ekskluderer index_til)
    }
    return new ArrayList<String[]>(list);
  }

  /**Se RMIClassInterface*/
  public void deleteConcept(String conceptName) throws RemoteException, IOException {
    for(Datasource ds : Controller.getDatasources()) ds.deleteConcept(conceptName);

  }

  /**Se RMIClassInterface*/
  public void deleteRelation(String conceptFrom, String conceptTo, String relationName) throws RemoteException, IOException {
    for(Datasource ds : Controller.getDatasources()) ds.deleteRelation(conceptFrom,conceptTo,relationName);
  }

  /**Se RMIClassInterface*/
  public void renameConcept(String oldConceptName, String newConceptName) throws RemoteException, IOException {
    for(Datasource ds : Controller.getDatasources()) ds.renameConcept(oldConceptName,newConceptName);
  }

  /**Se RMIClassInterface*/
  public void renameRelation(String conceptFrom, String conceptTo, String oldRelationName, String newRelationName) throws RemoteException, IOException {
    for(Datasource ds : Controller.getDatasources()) ds.renameRelation(conceptFrom,conceptTo,oldRelationName, newRelationName);
  }

  /**Se RMIClassInterface*/
  public void createRelation(String conceptFrom, String relation, String conceptTo) throws RemoteException, IOException {
    Controller.getDatasources().get(0).createRelation(conceptFrom,relation, conceptTo);
  }

  public String popMessageFromQueue() throws RemoteException {
    return messageFIFOQueue.poll();
  }
}

package genetuc_kb_grensesnitt.server;

import genetuc_kb_grensesnitt.TucGuiException;
import java.io.IOException;
import java.rmi.Remote;
import java.rmi.RemoteException;
//import java.rmi.Exception;

import java.util.LinkedList;

/**Interface for Remote klasse som aksesseres fra klient.
 */
public interface RMIClassInterface extends Remote{

  public final LinkedList<String> messageFIFOQueue = new LinkedList<String>(); //fifo meldingskø. server putter inn, klient leser periodisk ut. Støtter kun en bruker nå.*/

  /**Metoden returnerer svg på textlig form (xml)
   *@param conceptName navn på konsept som er i fokus
   *@param maxDistance hvor mange kanter vekk fra konsept som er i fokus skal hentes? Jo lengre distanse, jo fler noder => lengre tid for graf-algoritme å bli ferdig
```

```
    *@param fromIndex optimeringstiltak: selv med maxDistance = 1 kan det bli for mange noder (f.eks. hvis "thing" er fokuskonsept). Da setter indexFrom og indexTo et intervall av relasjoner som hentes.
    *@param toIndex optimeringstiltak: selv med maxDistance = 1 kan det bli for mange noder (f.eks. hvis "thing" er fokuskonsept). Da setter indexFrom og indexTo et intervall av relasjoner som hentes.
    */
   public String getAnSVG(String conceptName, int maxDistance, int fromIndex, int toIndex) throws RemoteException, TucGuiException ; //TODO: skift navn for å reflektere at det er xml tekst man får

   /**Sletter et konsept.*/
   public void deleteConcept(String conceptName) throws RemoteException, IOException;

   /**Sletter en relasjon*/
   public void deleteRelation(String conceptFrom, String conceptTo, String relationName) throws RemoteException, IOException;

   /**Endrer navn på et konsept*/
   public void renameConcept(String oldConceptName, String newConceptName) throws RemoteException, IOException;

   /**Endrer navn på en relasjon*/
   public void renameRelation(String conceptFrom, String conceptTo, String oldRelationName, String newRelationName) throws RemoteException, IOException;

   /**Oppretter en ny relasjon. Nye relasjoner blir lagt i den første datakilden.*/
   public void createRelation(String conceptFrom, String relation, String conceptTo) throws RemoteException, IOException;

   /**Returnerer det som ligger på toppen av meldingskøen. Eller null*/
   public String popMessageFromQueue() throws RemoteException;
}

package genetuc_kb_grensesnitt;

import java.io.IOException;
import java.io.StringReader;
import java.util.ArrayList;
import java.util.HashMap;
import org.apache.batik.dom.svg.SAXSVGDocumentFactory;
import org.apache.batik.util.XMLResourceDescriptor;
import org.w3c.dom.svg.*;

/**Klassen inneholder diverse verktøy-metoder for å generere svg-dokumenter.*/
public class Tools {

   /**Bestemmer x- og y-offset for tekst (konseptnavn) sånn den vises i ellipsene hos klienten*/
   public final static int CONCEPTTEXT_OFFSET_X = -10, CONCEPTTEXT_OFFSET_Y = 4;
   /**Bestemmer radius og senter (offset) for konsept-ellipsene som vises hos klient*/
   final public static String RX="20",RY="20",CX="0",CY="0";


   /**Metoden genererer svg-tekst (se diplom teksten for forklaring til struktur) basert på tripletter med relasjoner (konsept relasjon konsept, f.eks. Jeg eren gutt)*/
   public static String generateSVGTextualDataFromTriplets(ArrayList<String[]> triplets, HashMap<String,int[]> positions) throws TucGuiException {
      return generateSVGText(triplets, positions);
   }

   /**Lager SVGDocument av tekst (som må være svg-xml)*/
   public static SVGDocument textToSVGDocument(String xmlSvg) throws IOException{
      SAXSVGDocumentFactory ssdf = new SAXSVGDocumentFactory(XMLResourceDescriptor.getXMLParserClassName());
      return (SVGDocument)ssdf.createSVGDocument(null,new StringReader(xmlSvg));//TODO: fix URI
   }

   /**Metode som tar inn tripletter med relasjoner (konsept relasjon konsept, f.eks. Jeg eren gutt) og returnerer SVGDocument*/
   public static SVGDocument generateSVGDocumentFromTriplets(ArrayList<String[]> triplets, HashMap<String,int[]> positions) throws TucGuiException, IOException{
      return textToSVGDocument(generateSVGText(triplets, positions));
   }

   /**Privat metode som gjør jobben med å sette sammen det tekstlige svg dokumentet*/
   private static String generateSVGText(ArrayList<String[]> triplets, HashMap<String,int[]> positions) throws TucGuiException { //TODO: noe under her som ikke bør hardkodes?

      String svgDocHeader = "<?xml version=\"1.0\" encoding=\"UTF-8\" standalone=\"no\"?><?xml-stylesheet " +
            "href=\"http://www.idi.ntnu.no/~fredrhag/genetuc_gui/style.css\" type=\"text/css\"?><svg  " +
            "xmlns:xml=\"http://www.w3.org/XML/1998/namespace\"   xmlns:dc=\"http://purl.org/dc/elements/1.1/\"   " +
            "xmlns:cc=\"http://web.resource.org/cc/\"   xmlns:rdf=\"http://www.w3.org/1999/02/22-rdf-syntax-ns#\"   " +
            "xmlns=\"http://www.w3.org/2000/svg\"   xmlns:svg=\"http://www.w3.org/2000/svg\"   " +
            "xmlns:xlink=\"http://www.w3.org/1999/xlink\"  height=\"500\"   width=\"500\" viewBox=\"0 0 500 500\" overflow=\"visible\" " +
            "xmlns:GenGui=\"http://www.idi.ntnu.no/GeneTuc/\"   >  " +
            "<defs>    <marker id=\"triangel\"      viewBox=\"0 0 10 10\" refX=\"35\" refY=\"5\"       " +
            "markerUnits=\"strokeWidth\"      markerWidth=\"10\" markerHeight=\"10\"      orient=\"auto\">      " +
            "<path d=\"M 0 0 L 10 5 L 0 10 z\" />    </marker>  </defs>  <g id=\"alt\" >   " +
```

```
            "<rect  id=\"bg\" x=\"-10000\" y=\"-10000\" width=\"20000\" height=\"20000\"/>";
        String svgDocFooter = "</g></g></svg>";
        String relationPart = "<g id=\"relations\">", conceptPart = "</g><g id=\"concepts\">"; //conceptPart har en /g for å terminere g fra relasjonsdelen av svg-en

        ArrayList<String> processedConcepts = new ArrayList<String>();

        for(String[] t : triplets){
            String conceptFrom = t[0], conceptTo = t[2], relation = t[1];
            int[] posXYcFrom = positions.get(conceptFrom);
            int[] posXYcTo = positions.get(conceptTo);
            if(posXYcFrom == null || posXYcTo == null) throw new TucGuiException("posXY var null, mest sannsynlig årsak: fil med posisjoner inneholdt ikke konseptet (skal ikke skje)");
            relationPart += "<g><path d=\"M "+posXYcFrom[0]+" "+posXYcFrom[1]+" L "+posXYcTo[0]+" "+posXYcTo[1]+"\" marker-end=\"url(#triangel)\" GenGui:from=\""+conceptFrom+"\" GenGui:to=\""+conceptTo+"\"/>" +
                    "<text x=\""+(int)((posXYcFrom[0]+posXYcTo[0])/2)+"\" y=\""+(int)((posXYcFrom[1]+posXYcTo[1])/2)+"\">"+relation+"</text></g>";

            //sjekker om konseptene som er med i relasjonene er støtt på før, hvis ikke, lag markup for dem
            boolean containsConcept = false;

            for(String c : processedConcepts){if(c.equals(conceptFrom))containsConcept = true;}

            if(!containsConcept){
                int[] posXY = positions.get(conceptFrom);
                if(posXY == null) throw new TucGuiException("posXY var null, mest sannsynlig årsak: fil med posisjoner inneholdt ikke konseptet (skal ikke skje)");
                String transform="translate("+posXY[0]+","+posXY[1]+")";
                conceptPart += "<g id=\""+conceptFrom+"\" transform=\""+transform+"\"><ellipse rx=\""+RX+"\" ry=\""+RY+"\" cx=\""+CX+"\" cy=\""+CY+"\"/><text
transform=\"translate("+CONCEPTTEXT_OFFSET_X+","+CONCEPTTEXT_OFFSET_Y+")\">"+conceptFrom+"</text></g>";
                processedConcepts.add(conceptFrom);
            }

            containsConcept = false;

            for(String c : processedConcepts){if(c.equals(conceptTo))containsConcept = true;}

            if(!containsConcept){
                int[] posXY = positions.get(conceptTo);
                String transform="translate("+posXY[0]+","+posXY[1]+")";
                conceptPart += "<g id=\""+conceptTo+"\" transform=\""+transform+"\"><ellipse rx=\""+RX+"\" ry=\""+RY+"\" cx=\""+CX+"\" cy=\""+CY+"\"/><text transform=\"translate("+CONCEPTTEXT_OFFSET_X+","+CONCEPTTEXT_OFFSET_Y+")\">"+conceptTo+"</text></g>";
                processedConcepts.add(conceptTo);
            }
        }
        return svgDocHeader+relationPart+conceptPart+svgDocFooter;
    }
}

package genetuc_kb_grensesnitt;

/** Enkel subklasse av Exception for å identifisere TucGui unntak.*/
public class TucGuiException extends Exception{

    public TucGuiException(String s) {
        super(s);
    }
}
```