NORGES TEKNISK-NATURVITENSKAPELIGE UNIVERSITET

FAKULTET FOR INFORMASJONSTEKNOLOGI, MATEMATIKK OG ELEKTROTEKNIKK

HOVEDOPPGAVE

Kandidatens navn:	Sondre Skaug Bjørnebekk
Fag:	Datateknikk
Oppgavens tittel (norsk):	Matematiske uttrykk i digitale biblioteksystemer
Oppgavens tittel (engelsk):	Mathematical Expressions in Digital Libraries

Oppgavens tekst:

I forbindelse med nettbasert læring og andre former for publisering på nett er det ofte behov for å lagre og visualisere matematiske uttrykk. Denne diplomoppgaven skal vurdere ulike alternativer for lagring av matematiske uttrykk og visualisering av disse. Det skal også lages en prototype-implementasjon av lagring og visualisering basert på MathML og gjøres en vurdering av om denne metoden er brukbar ved stor trafikk.

Fullstendig oppgavetekst, inklusiv en punktvis presisering, finnes i appendiks A.

Oppgaven gitt:	14.01.2003
Besvarelsen leveres innen:	02.06.2003
Besvarelsen levert:	30.05.2003
Utført ved:	Institutt for datateknikk og informasjonsvitenskap
Veiledere:	Ingeborg Sølvberg, Trond Aalberg

Trondheim, 30.05.2003

Professor Ingeborg Sølvberg
Faglærer

Forord

Denne rapporten oppsummerer arbeidet med hovedoppgaven i sivilingeniørstudiet ved Institutt for datateknikk og informasjonsvitenskap, NTNU. Oppgaven er utført på heltid i 10. semester av studiet. Arbeidet startet i begynnelsen av januar 2003 og frist for levering av denne rapporten var 2. juni.

Oppgaven var initiert av Inspera AS[1], men arbeidet er gjennomført uavhengig av eventuelle senere bruksområder for resultatene. En annen interessent i arbeidet som gjøres er Aschehoug forlag som i løpet av høsten skal i gang med nettsted for fysikk og senere også for matematikk. Aschehoug og Inspera samarbeider om digitale læremidler ved at Aschehoug bruker Insperas tekniske plattform for sine nettsteder.

Undertegnede føler seg veldig privilegert etter å ha fått jobbet med et svært interessant emne og selv kunne styre hele prosessen, disponere tiden og gjøre prioriteringer (som selvsagt har blitt justert etter gode råd). Om noen av leserne av rapporten er potensielle nye studenter ved IDI vil jeg herved anbefale studiet på det varmeste!

Jeg har valgt å skrive selve rapporten på norsk, men all kode og implementasjon er gjort på engelsk.

Veileder for oppgaven har vært professor Ingeborg Sølvberg. Trond Aalberg har vært rådgiver i tekniske spørsmål. En stor takk for utmerket samarbeid og gode råd underveis rettes med dette til begge.

Fra Insperas side var det administrerende direktør Bjørn Rustberggard som tok initiativet til oppgaven. Arbeidet med oppgaven har vært svært givende og at Bjørn har vært interessert i fremdriften hele veien har vært en motivasjonsfaktor. Jeg vil få takke Bjørn for dette.

På hjemmebane har Trine-Marie vært svært forståelsesfull, selv om lange dager med arbeidet med ”diplomen” nok har gjort at jeg ikke har fått deltatt så mye som jeg skulle ønsket i bryllupsforberedelsene. Hun har også minnet meg på at det er viktig å jobbe skikkelig også før den siste innspurten, noe jeg var veldig glad for når den siste innspurten kom. Jeg får verdens beste kone og dette er bare et lite eksempel på det. Takk!

Trondheim, 30.05.2003

Sondre Skaug Bjørnebekk

[1] Inspera er et teknologiselskap med kjernekompetanse innen elektronisk læring, publiseringssystemer (CMS) og annen portalteknologi. Insperas hjemmesider finnes på http://www.inspera.no/.

Sammendrag

Denne oppgaven ser på representasjonsbehov for matematiske uttrykk når disse skal lagres for bruk i digitale bibliotek eller på web. Både matematisk mening og den symbolske visningen av uttrykkene er viktig i slike systemer. I tillegg kan det være aktuelt med visualiseringer, for eksempel i form av grafer.

I oppgaven gjøres et valg av et format for å støtte begge behovene nevnt over (semantikk og presentasjon) og en arkitektur for bruk av dette formatet til symbolsk visning og tegning av grafer blir utformet og presentert.

Det presenteres videre en prototype som bruker det valgte formatet og viser hvordan dette kan brukes både til å utnytte semantikken i det lagrede uttrykket for å tegne opp en graf og bruke presentasjonsinformasjonen til å vise uttrykket symbolsk. I tillegg vises en fullverdig implementasjon av krysspublisering fra samme lagrede matematikk-kilde til ulike sluttformater. Krysspublisering støttes også for grafer, samt at interaksjon og dynamiske muligheter med animasjon, zoom og visning av bestemte punkter er mulig.

En analyse av fordeler og ulemper med en distribuert modell for tegning av grafer blir gjort. Konklusjonen her er at det finnes gode argumenter for en distribuert modell, både med tanke på funksjonelle krav og mulighet til skalerbarhet. Skalerbarhet og ytelse blir vurdert i et separat kapittel, herunder med en praktisk simulering av mange sluttbrukere mot en tjenermaskin med prototypen kjørende. På bakgrunn av dette konkluderes det at evnen til å støtte svært mange samtidige brukere er omtrent hundre ganger større i den distribuerte modellen.

Implementasjonen i prototypen er dokumentert både i form av diagrammer, oversikt over klasser og deres struktur (i hovedsak i form av et appendiks) og ved at all kildekoden befinner seg på en CD-ROM vedlagt denne rapporten. Prototypen er fullt fungerende og alle mål som var satt til denne ved prosjektstart er nådd. Enkelte eksempler fra prototypen kan sees i praksis direkte fra vedlagte CD-ROM, uten noen serverplattform installert.

Med tanke på valg av format konkluderes det med at matematikk på web er et område hvor det ikke er definitivt etablerte såkalte ”best practices” for hva man bør støtte av standarder for å muliggjøre utveksling og sikre fremtidig nytte av dataene. Ved å velge et format som både ivaretar semantikk og presentasjon og lett kan oversettes til andre formater vil man være godt sikret mot fremtidige endringer i trender.

Innhold

FORORD ..... **III**

SAMMENDRAG ..... **V**

INNHOLD ..... **VII**

FIGURLISTE ..... **IX**

1. OPPGAVENS OMFANG OG INNHOLD **2**

2. INTRODUKSJON **4**

2.1. OVERORDNET INTRODUKSJON 4
2.2. HVORFOR KRYSSPUBLISERING? 4
2.3. XML SOM UTVEKSLINGSFORMAT PÅ INTERNETT 6
2.4. REPRESENTASJONSBEHOV OG METODER FOR MATEMATIKK 8

3. EN KORT INNFØRING I MATHML **12**

3.1. PRESENTASJONSDELEN AV MATHML 13
3.2. REPRESENTASJONSDELEN AV MATHML 13
3.3. KOMBINASJON AV PRESENTASJON OG SEMANTIKK I MATHML 13

4. EKSISTERENDE LØSNINGER OG KOMPONENTER **16**

4.1. KRYSSPUBLISERING 16
4.2. MATHML 19

5. UTFORMING AV ARKITEKTUR FOR PROTOTYPEN **22**

5.1. KRAV SOM MÅ STØTTES 22
5.1.1. Generelle krav *22*
5.1.2. Spesifikke krav for matematikkløsning *22*
5.2. VALG AV LAGRINGSFORMAT FOR MATEMATISKE UTTRYKK 24
5.3. ANSVARSFORDELING MELLOM KLIENT OG TJENER 27
5.4. ARKITEKTUR FOR VISNING I FORM AV SYMBOLSKE UTTRYKK 28
5.5. ARKITEKTUR OG TEKNOLOGI FOR GRAFTEGNEREN 30
5.6. KLASSEMODELL OG DATABASEMODELL 34
5.6.1. Klassemodell (Java) *34*
5.6.2. Databasemodell - Oracle *41*

6. PRESENTASJON AV UTVIKLET PROTOTYPE **44**

6.1. INNLEGGING OG VISNING AV MATEMATISKE UTTRYKK 44
6.2. GRAFTEGNER 48
6.3. OPPSUMMERING AV FUNKSJONALITET I PROTOTYPEN 54

7. VURDERING AV YTELSE OG SKALERBARHET **56**

7.1. TEORETISK MODELL FOR YTELSESVURDERING 56
7.2. PRAKTISK GJENNOMFØRING AV YTELSESMÅLINGER 59
7.3. TESTRESULTATER 60
7.4. KONKLUSJONER PÅ BASIS AV UTFØRT YTELSESTEST 63

8. AVSLUTNING OG OPPSUMMERING **64**

8.1. NYTTEVERDI AV OPPGAVEN 64
8.2. VIDERE ARBEID 64
8.3. KONKLUSJON OG AVSLUTNING 66

9. KILDER OG REFERANSER **68**

APPENDIKS A – OPPGAVETEKST **72**

APPENDIKS B – KILDEKODE OG DOKUMENTASJON AV LØSNING **74**

VEDLAGT CD-ROM 74
UTSKRIFT AV FORENKLET JAVADOC 75

APPENDIKS C – ELEKTRONISKE EKSEMPLER FRA GRAFTEGNEREN **154**

APPENDIKS D – TESTRESULTATER FRA YTELSESTEST **156**

Figurliste

Figur 1 – Krysspublisering: Arkivering for gjenbruk 5
Figur 2 – Krysspublisering: Ulike sluttformater fra samme lagrede kilde 6
Figur 3 – Matematikk i symbolsk form 8
Figur 4 – Matematikk i form av en graf 8
Figur 5 – Et uttrykk i XML-kodet OpenMath-format 10
Figur 6 – Et matematisk uttrykk i LaTEX-formatet 10
Figur 7 – Et uttrykk i presentasjonsrettet MathML 13
Figur 8 – Et uttrykk i semantisk rettet MathML 13
Figur 9 – Et uttrykk med kombinert mening og presentasjon i MathML 14
Figur 10 – Teknisk arkitektur for krysspublisering 16
Figur 11 – Håndtering av en forespørsel i Apache Cocoon 17
Figur 12 – Insperas n-lagsarkitektur 19
Figur 13 – Sammenligning av Content MathML og OpenMath for samme uttrykk 24
Figur 14 – En forespørsel i et generelt klient/tjenersystem 27
Figur 15 – Utregning av skjermkoordinater fra logiske koordinater 33
Figur 16 – Klassemodell for prototypen i Java 35
Figur 17 – Sekvensdiagram for kallene ved tegning av en graf i graftegneren 41
Figur 18 – Databasemodell for lagring av MathML i en Oracle-database 42
Figur 19 – Redigering av matematiske uttrykk med Design Science WebEQ 44
Figur 20 – Veiviser for innsetting av matematikk i symbolsk form eller graf i et dokument 45
Figur 21 – Utvidelsen av DTDen for Insperas dokumenter for bruk av matematiske objekter 46
Figur 22 – Demonstrasjon av krysspublisering for et dokument med matematiske uttrykk 47
Figur 23 – Matematiske uttrykk i form av graf og symbolsk visning i et dokument i HTML-format 48
Figur 24 - Matematiske uttrykk i form av graf og symbolsk visning i et dokument i PDF-format 49
Figur 25 – Parameterbasert uttrykk tegnet i graftegnerprototypen 50
Figur 26 – Zoom til valgt rektangel i prototypens graftegnerdel 51
Figur 27 – Markering av lokal ekstremalverdi i prototypens graftegnerdel 51
Figur 28 – Bildeserie fra bruk av graftegneren i animasjonsmodus 52
Figur 29 – Brukerstyrte parametre med graftegneren i animasjonsmodus 53
Figur 30 – Spesifisering av ledetekst og startverdi for brukerstyrte parametre 53
Figur 31 – Dekomponering av totalarbeidet krevet av en forespørsel (service demand) 56
Figur 32 – Gjennomstrømning som funksjon av antall brukere med sentralisert modell 60
Figur 33 – Gjennomstrømning som funksjon av antall brukere med distribuert modell 60
Figur 34 – Responstid som funksjon av antall brukere med sentralisert modell 61
Figur 35 – Responstid som funksjon av antall brukere med distribuert modell 61

1. Oppgavens omfang og innhold

Oppgaven bestod i å vurdere og velge representasjonsform for matematikk, samt lage en prototype på publisering og visualisering av matematiske uttrykk. Det var også et ønske om en ytelsestest med kvantifiserte betraktninger om hvorvidt prototypen kunne egne seg for bruk i større skala.

Som en del av prototypen ble det utviklet en graftegner. Graftegnerens virkeområde er begrenset til visualiseringer i form av todimensjonale grafer. Diskusjonen rundt arkitektur og selve implementasjonen av prototypen forutsetter dette, selv om det forsøkes å legge til rette for videre utvidelser.

Implementasjonen av prototypen ble gjort (og skulle i følge oppgaven gjøres) i Inspera Platform-rammeverket, som er J2EE-basert. Det var altså lagt visse føringer på teknologivalg ved at tjenerdelen av prototypen dermed skulle benytte Java.

Kopi av opprinnelig og fullstendig oppgavetekst[2] slik den ble levert av den eksterne initativtakeren til oppgaven, Inspera AS, finnes i appendiks A.

[2] Det vil si en marginalt justert utgave slik den ble godkjent av veileder etter innledende møte 15. januar 2003.

2. Introduksjon

2.1. Overordnet introduksjon

Mye av arbeidet med datamaskiner dreier seg direkte eller indirekte om matematikk. Som kjent opererer datamaskiner med binære tall og alle avanserte datastrukturer som benyttes vil i siste innstans oversettes til denne enkle representasjonsmåten som kun består av av/på, eller 0/1 om man vil.

I denne hovedoppgaven skal jeg ikke dvele ved representasjonsmåter på det laveste tekniske nivået. Dette er i seg selv interessant, men det er i stor grad områder hvor det er veletablert teori og til dels er dette også basiskunnskap innen mange av datafagene. Oppgaven dreier seg imidlertid om integrasjon og bruk av matematikk på et mye høyere abstraksjonsnivå – nemlig bruk av slike i digitale biblioteksystemer[3], spesielt med tanke på visning og levering over Internett.

Matematiske uttrykk har en mening eller semantikk, de representerer ideer og resonnementer. Visningen eller presentasjonen er også viktig, man trenger en forståelig, konsekvent og tiltalende notasjon. Som vi senere skal se finnes denne todelingen også i enkelte formater som benyttes for å lagre matematiske uttrykk.

Integrering av matematiske uttrykk i nettbaserte innholdstjenester er ikke trivielt, slik man kanskje først kunne ledes til å anta. Noe som kompliserer forholdet er at man ofte må forholde seg til brukere uten spesialprogramvare med mulighet for tolkning av spesielle formater. Hittil har derfor en utvei vært å lage de avanserte uttrykkene som bilder (med alternative tekster som i beste fall kan fortelle noe om den matematiske meningen) eller publisere i elektronisk papirformat som for eksempel PDF eller Postscript.

I denne rapporten oppsummeres arbeidet som har fokusert på integrering av matematiske uttrykk i et krysspubliseringssystem, herunder en enkel metode for graftegning innenfor samme system. Jeg starter med å se på litt av bakgrunnen for krysspublisering og hva som er motivasjonen for dette.

2.2. Hvorfor krysspublisering?

Med krysspublisering menes her publisering av innhold fra én kilde til flere ulike medier/sluttformater, slik at man kan gjenbruke informasjonen i ulike sammenhenger[4]. Eksempelvis kan man bruke samme kildemateriale i et egnet format til å produsere nettsider i sluttformatet HTML, elæringskurs i sluttformatet Macromedia Flash, eller elektronisk bok i sluttformatet PDF.

Krysspublisering brukes også om publisering i ulike former innenfor samme format (for eksempel presentasjon av samme innhold med ulikt utseende avhengig av målgrupper på web). Det er imidlertid den førstnevnte betydningen jeg vil konsentrere meg om i denne oppgaven.

[3] "Digitale biblioteksystemer" er et begrep som benyttes ganske ulikt av forskjellige forfattere og av næringslivet benyttes det relativt lite for tiden. Men mange av prinsippene og emnene jeg vil behandle er like relevante for det man gjerne kaller Content (eller Knowledge) Management Systems i næringslivet, selv om man ikke alltid vil tenke at disse systemene nødvendigvis er digitale bibliotek.

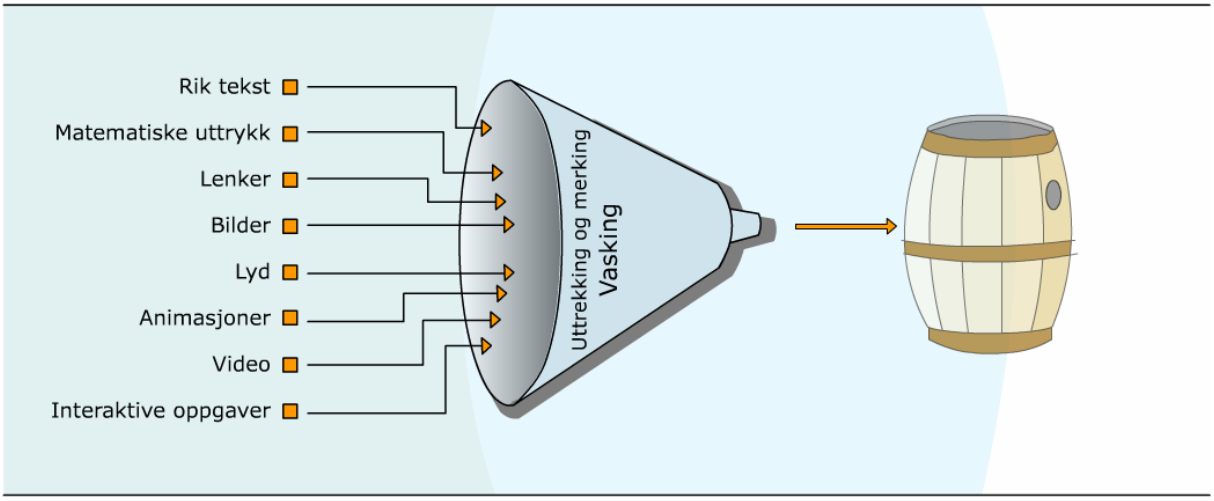
[4] Ofte brukes også begreper som database- og parallellpublisering.

Krysspublisering gjør at man kan *oppdatere kun ett sted* og likevel *publisere i flere medier* eller format, siden samme kilden brukes til å produsere de ulike sluttformatene. Dette gir en høyere informasjonskvalitet ved at informasjonen i alle formater alltid forblir oppdatert i forhold til kildematerialet. Det blir altså mindre vedlikeholdsjobb og en eliminerer risikoen for at en glemmer å gjøre de samme oppdateringene i alle format.

For å muliggjøre krysspublisering må to forhold løses:

1) Informasjonsbestanddelene må representeres og lagres medieuavhengig
2) Sammensatte sluttprodukter må kunne tas ut på ulike formater

For at alle informasjonsbestanddeler som inngår i et sluttprodukt som skal kunne representeres og lagres medieuavhengig, må deres innhold skilles fra form. Eksempel på slike bestanddeler er gitt i figuren under.

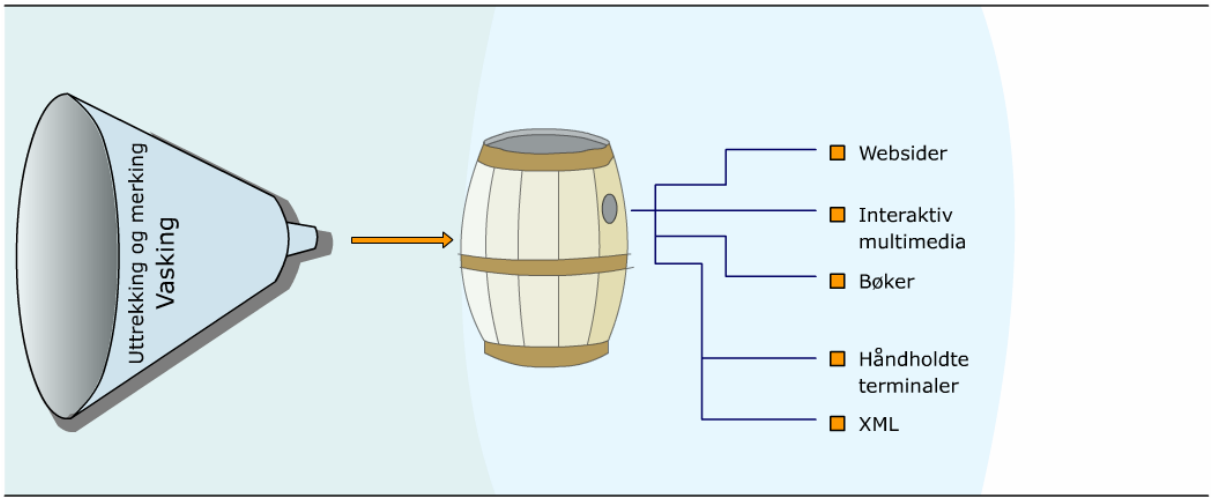


Figur 1 – Krysspublisering: Arkivering for gjenbruk

Figuren er hentet fra [INSPERA2003], den viser svært forenklet hvordan disse informasjonsbestanddelene vaskes og lagres i en ”innholdstønne”. Lagringen er medieuavhengig. I et system for krysspublisering vil matematiske uttrykk være en like sentral bestanddel som for eksempel bilder og lyd. Dette gjelder spesielt for informasjonsprodukter av mer teknisk art slik som tekniske publikasjoner og materiale for teknisk eller matematisk opplæring.

Tønnen og den filtrerende silen i figuren over, ofte kalt for et Content (eller Knowledge) Management System, blir altså svært sentral i produksjonssyklusen for innhold i organisasjonen. Dette setter bestemte krav til systemene og jeg ser litt nærmere på noen av disse i avsnitt 5.1.1, da med en noe mer teknisk vinkling enn her.

Når disse informasjonsbestandelene er representert og lagret i tønnen, må de kunne aggregeres og transformeres til sluttprodukter i ulike formater (for eksempel websider og katalog) som figuren under viser.



Figur 2 – Krysspublisering: Ulike sluttformater fra samme lagrede kilde

Herfra er ikke veien lang til det næringslivet ofte kaller ”print-on-demand”-løsninger hvor man automatiserer sideombrekking og stilsetting av trykksaker. I tillegg til det mulige bidraget til bedret informasjonskvalitet nevnt over, kan altså slike løsninger effektivisere både trykking og distribusjon av innhold ved at man parallellpubliserer fra samme kilde.

Ofte vil man i slike tilfeller levere informasjon tilpasset til målgruppene. Dette kalles gjerne selektiv eller personalisert publisering. Selektiv publisering kan gi kostnadsbesparelser for organisasjoner som tilbyr varianter av informasjonsprodukter til ulike segmenter. Et krysspubliseringssystem kan legge til rette for slik selektiv sammensetting av informasjon skreddersydd for en bestemt målgruppe.

Støtte for matematiske uttrykk, både symbolsk og i form av grafer, finnes bare i liten grad i dagens systemer for krysspublisering og det er heller ikke veldig mye publisert materiale om emnet. Med bakgrunn i dette forsøker jeg i denne oppgaven å bidra med ny forståelse og gjennom prototypen vise funksjonalitet som kan være nyttig i slike systemer.

I kapittel 4 ser jeg nærmere på metodikker og tekniske løsninger for krysspublisering og kommer da også inn på mer tekniske detaljer. Jeg vil imidlertid allerede nå se litt nærmere på egenskaper XML-formater har felles, siden slike vil bli diskutert i flere sammenhenger allerede tidlig i oppgaven.

2.3. XML som utvekslingsformat på Internett

World Wide Web Consortium (heretter W3C) er standardiseringsorganet fremfor noen for webrelaterte standarder og er en åpen sammenslutning av industri og akademia. XML, eXtensible Markup Language, er en standard for tekstbaserte dokumenter fra W3C. XML definerer en generisk syntaks med faste regler for at et dokument skal være gyldig[5], men er samtidig så fleksibelt at det har mange svært ulike bruksområder.

Man definerer selv sitt dataformat (eller dokumentformat om man vil) med sine egne elementer og regler for hvordan disse settes sammen. Eksempler på bruk av XML i stor skala

[5] Begrepet som brukes er ”well-formed” og dette har en helt spesiell betydning. Å gå i detalj om dette (eller annen XML-terminlogi som nok vil bli benyttet i rapporten) her er ikke mulig og i den grad slik kunnskap behøves for å få fullt utbytte av rapporten må det forutsettes kjent. [HAROLD2002] forklarer alle slike begreper på en ryddig og god måte.

er SVG-formatet (Scalable Vector Graphics) for vektorbasert grafikk og XHTML for websider. Det finnes utallige formater, både standardiserte og definerte av den enkelte organisasjon/utvikler. Bruksområdene varierer fra alt fra ad-hoc formater som brukes som et mellomlag i visning av websider til oversending av transaksjoner innenfor e-handel etter bestemte format.

Det er vanlig å dele inn XML-formater i datasentriske (ehandelstransaksjonene nevnt over vil være et typisk eksempel på dette) og dokumentsentriske (XHTML som brukes i websider er et typisk eksempel på det). Felles for begge formatfamiliene er et fokus på enklere utveksling mellom ulike systemer, organsiasjoner og programvare – ofte kalt interoperabilitet. For en mer fyldig oversikt over egenskaper, likheter og forskjeller se kapitlene 6, 7 og 14 i [HAROLD2002].

At det i XML er faste regler for gyldighet, gjør det lettere å skrive programvare som behandler et XML-dokument på en generisk måte. Dette gjør også at det finnes et svært godt utvalg verktøy for XML. Dette gjelder ikke minst for Java som benyttes som implementasjonsspråk for prototypen i dette prosjektet.

XML er nok til dels et overmarkedsført format – for det er jo nettopp bare et format det er. Det er *ikke* et programmeringsspråk, en nettverksprotokoll eller en database slik en del omtale kan synes å mene. At formatet er standardisert (og lesbart for utviklere) gjør likevel datautveksling og testing lettere, samt at man ikke på noen måte binder seg til en bestemt plattform.

XML i seg selv definerer kun syntaksregler og rammeverket for et format. Dette gir som nevnt gode muligheter for generelle verktøy. Det gir likevel ikke automatisk muligheter for maskinell behandling og tolking av selve innholdet. Imidlertid arbeides det på mange hold med standardisering av XML-baserte formater og MathML (Mathematical Markup Language), et format denne oppgaven vil inneholde mye informasjon om, er et eksempel på dette. Med definerte formater og standardisering av meningen av disse kan mange av markedsføringsbegrepene rundt XML ha en viss realisme i seg.

Nå som vi har sett litt på noen grunnleggende egenskaper både ved krysspublisering og ved XML som er det mest aktuelle formatet å benytte i Insperas rammeverk for dette, er det naturlig å gå videre med hovedtemaet for oppgaven – nemlig matematiske uttrykk og muligheter for krysspublisering av disse.

2.4. Representasjonsbehov og metoder for matematikk

Som nevnt i innledningen kan man dele behovene ved lagring av matematiske uttrykk grovt sett i to: Semantikk og presentasjon. Disse to sidene av matematiske uttrykk er selvsagt ikke atskilte, men sterkt avhengige. Oversetting mellom de to formene kan ikke alltid gjøres entydig eller uten å tape informasjon. Jeg vil se litt nærmere på dette, delvis i lys av konkrete formater og deres egenskaper. For en detaljert sammenligning og valg av format, se kapittel 5.2.

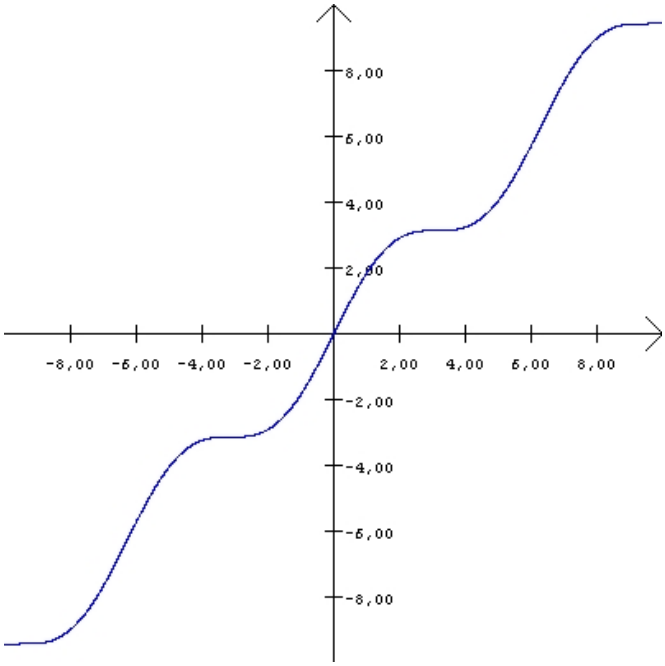
Semantikken i matematiske uttrykk er den underliggende meningen, altså hva uttrykket betyr. Konkrete eksempler på dette følger under avsnittet om MathML, som er et XML-basert format for lagring av matematiske uttrykk. For enkelte typer bruk, i denne oppgaven gjelder det spesielt tegning av grafer, er informasjon om den matematiske meningen til et uttrykk uunnværlig.

Presentasjonen og notasjonen er også viktig i forbindelse med matematiske uttrykk. Det er skrevet hele bøker om korrekt typografi for matematikk, så som regler for plassering av parenteser, innretting av hevet og senket skrift og når kursiv skal brukes.

I tillegg til presentasjon i symbolsk/typografisk form kan enkelte matematiske uttrykk også ha en representasjon i form av en graf. Under er uttrykket som vi oppfatter har meningen ”x pluss sinus til x” vist i form av symbolsk uttrykk til venstre og en graf til høyre. Grafen er for øvrig produsert ved hjelp av prototypen på en graftegner som er laget som en del av denne oppgaven.

$$x + \sin(x)$$

Figur 3 – Matematikk i symbolsk form



Figur 4 – Matematikk i form av en graf

Mulighet for visualisering av matematiske uttrykk i form av grafer er også et behov som kan dekkes godt av elektronisk innhold, siden man her kan kombinere statisk innhold med generering av grafer etter brukerens ønsker.

Et viktig mål med en standardisert lagringsmetode for matematiske uttrykk vil være interoperabilitet, altså mulighet for at ulike applikasjoner kan snakke sammen gjennom utveksling av det definerte formatet. I forbindelse med bruk på web eller digitale biblioteksystemer er det også en del nye krav som dukker opp, slik som søkbarhet og mulighet for interaktivitet.

For å støtte krysspublisering av matematiske uttrykk er det vesentlig å ha et standardisert format. Ved at formatet er standardisert står man friere til å velge ulike verktøy til de ulike delene av prosessen med krysspublisering, som vist i figurene "Figur 1 – Krysspublisering: Arkivering for gjenbruk" og "Figur 2 – Krysspublisering: Ulike sluttformater fra samme lagrede kilde". Dersom man velger et XML-basert format er man sikker på at ikke formatet er knyttet til et bestemt operativsystem og i tillegg vil man (siden formatet er lesbart for utviklere og standardisert i formen) få tilgang til dataene uavhengig av verktøyprodusenten til verktøyet som opprinnelig laget dataene.

MathML er et XML-basert format for å representere matematisk notasjon som kan beskrive både presentasjon og matematisk mening. Anbefalingen (W3C sin terminologi for det som i utviklermiljøene ofte refereres uformelt til som "standard") beskriver hvordan formatet kan brukes både frittstående og direkte sammen med XHTML, som er språket som brukes på websider. En kort innføring i MathML gis i kapittel 3 av denne rapporten.

OpenMath-initiativet[6] har publisert en standard for hvordan semantikken i matematiske uttrykk kan representeres i et XML-format. Det er nær sammenheng mellom OpenMath sitt XML-format og den delen av MathML som går på semantikken i uttrykkene (noe mer utførlig om dette kommer i kapittel 5.2). Det er også overlapp både mellom utviklermiljøene og representantene i komiteene for de to tilsynelatende konkurrerende standardene.

OpenMath opererer med tre lag i modellen, nemlig *privat*, *abstrakt* og *kommunikasjon*. Det private laget er ikke standardisert i OpenMath, men er derimot knyttet til applikasjonens interne format. Det abstrakte nivået svarer til objektene i OpenMath-standarden (se [OPENMATH2002]) og deres semantikk spesifiseres i såkalte Content Dictionaries. På kommunikasjonsnivået er OpenMath-objektene serialisert som en strøm av bytes og kan dermed flyttes over nettverk, eller til og fra en fil. XML-kodingen kan sies å være en del av kommunikasjonsnivået, men den hjelper oss samtidig (i motsetning til et binært format) å kunne diskutere det abstrakte nivået siden uttrykkenes struktur er menneskelig lesbart i XML.

Grunnsteinene i OpenMath-formatet er et OpenMath-objekt som har applikasjoner, binding av variable, variable og symboler. I eksemplet under ser vi hvordan disse representeres ved XML-elementene `OMOBJ`, `OMA`, `OMBIND`, `OMV` og `OMS`. Referansene til de nevnte content dictionaries gjøres via `cd`-attributtet. På neste side følger et enkelt eksempel i OpenMath der uttrykket $(a+b)*(c+d)$ er kodet.

[6] For mer informasjon om initiativet, basert i Helsinki, se hjemmesiden på http://www.openmath.org/.

```
<OMOBJ>
  <OMA>
      <OMS name="times" cd="arith1"/>
      <OMA>
         <OMS name="plus" cd="arith1"/>
         <OMV name="a"/>
         <OMV name="b"/>
      </OMA>
      <OMA>
          <OMS name="plus" cd="arith1"/>
          <OMV name="c"/>
          <OMV name="d"/>
      </OMA>
  </OMA>
</OMOBJ>
```

Figur 5 – Et uttrykk i XML-kodet OpenMath-format

Det hevdes blant annet i [CAPROTTI1999] at de to standardene ikke er konkurrenter, men komplementære. Dette stemmer nok i hovedsak, men det kan nok også virke som om den ene halvdelen av MathML konkurrerer direkte med OpenMath-formatet innen enkelte bruksområder. Jeg kommer litt mer tilbake til dette i det neste kapitlet om MathML. I tillegg kommer jeg tilbake med noen betraktninger rundt de to formatene i forbindelse med valg av lagringsformat som benyttes i prototypen som er implementert som en del av denne oppgaven.

I vitenskaplige miljøer har typesettingssystemet T_EX (se [KNUTH1986]) og det tilhørende formatet, designet av anerkjente Donald Knuth, fått spesielt godt fotfeste. Matematikkdelen i T_EX (og varianter av dette så som LaT_EX, som er populært på Linux) har fokus på presentasjonsdelen av problematikken med matematiske uttrykk. Selv om det finnes forsøk på å tolke den matematiske meningen i slike uttrykk er ikke notasjonen utformet med tanke på dette.

Formatet er egnet for publisering til ulike formater, men støtten er best til typiske utskriftsformater, som for eksempel postscript. Det finnes likevel løsninger for å oversette til websider i HTML-format med matematiske uttrykk som bilder. Under følger et eksempel der den generelle løsningen på andregradsligninger, $\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$, er skrevet i LaT_EX.

```
\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}
```

Figur 6 – Et matematisk uttrykk i LaT_EX-formatet

Vi ser at syntaksen er kompakt og har definerte navn for matematiske konstruksjoner. For en innføring i syntaksen og virkemåten til LaT_EX, se [WILKINS1995]. Her vises det også hvordan man skrur matematisk modus av og på i dokumenter, samt mer avanserte eksempler enn dette enkle som er vist over.

Det finnes også mange mer eller mindre proprietære formater for lagring av matematiske uttrykk. For eksempel har Microsoft Word en innebygd formeleditor, der fokus hovedsaklig

er på presentasjon. Matematiske programmer som Maple og Mathematica har sine egne formater og redigeringsverktøy. Den matematiske meningen må her selvsagt bevares, siden disse verktøyene gjør avanserte matematiske beregninger. Stephen Wolfram (se [WOLFRAM2003A] for en kort biografi) med sitt selskap, som gir ut programpakken Mathematica, har vært blant pådriverne for MathML og deres verktøy støtter eksport og import til MathML.

Et initiativ som kan være verdt å nevne er Digital Library of Mathematical Functions (DLMF[7]). DLMF er et prosjekt som skal ende opp både i en bok som skal erstatte den svært populære, men ganske utdaterte, ”Handbook of Mathematical Functions” ([ABRAMOWITZ1972]) utgitt av det amerikanske handelsdepartementet. DLMF skriver i sine mål at boken skal komme ut i 2004 og at nettstedet skal være fullt operativt i år. Foreløpig presenterer de arbeidet så langt som det har kommet på en midlertidig måte ved bruk av HTML og bilder, men de skriver at de planlegger å ta i bruk MathML når tiden er moden. Råmaterialet som benyttes av DLMF produseres i LaT$_E$X og skal oversettes semi-automatisk til et format (sannsynligvis OpenMath) som bedre bevarer semantikken for deretter å kunne eksporteres i ulike format.

[7] Hjemmesiden til DLMF er http://dlmf.nist.gov/.

3. En kort innføring i MathML

W3C har som nevnt standardisert et markupspråk for matematikk, nemlig MathML. Dette foreligger nå i versjon 2, fra 21. februar 2001, spesifisert i [W3C-MATHML2]. Siden det er en XML-basert standard integreres MathML enkelt med XHTML (som er HTML reformulert som en applikasjon av XML, se [W3C-XHTML]). Alt som ble sagt i avsnitt 2.3 om XML og fordeler ved utveksling gjelder derfor også MathML.

På sikt skal MathML kunne benyttes direkte i webdokumenter (XHTML) og vises av nettleserne (UA eller User Agents er termen som benyttes av W3C). Per i dag er støtten for dette noe spredt, slik at det også uten krysspubliseringsbehovet ville vært relevant å se på de temaene jeg gjør i denne oppgaven. [W3C-MATHMLOV] fra W3C gir en oversikt over hvordan publisere MathML direkte til ulike nettlesere via ymse plugins og tredjepartskomponenter.

At XML (og dermed MathML) er menneskelesbart og forståelig (til en viss grad) sees på som en fordel for utviklerne. Imidlertid er det ikke meningen at sluttbrukerne skal forholde seg til markeringsspråket, men heller bruke programmer som hjelper dem i det de vil gjøre og lagrer i MathML. MathML støttes allerede av mange leverandører, blant annet av tidligere nevnte Wolfram Research som leverer det mye brukte verktøyet for vitenskaplig matematikk, Mathematica (se [WOLFRAM2003B] for en beskrivelse av Mathematica).

MathML har også en del regler utover reglene den arver ved å være en XML-applikasjon, siden appendiks B i [W3C-MATHML2] spesifiserer regler for tolkning utover det en DTD kan spesifisere. XML Schema kan heller ikke, til tross for større uttrykksevne enn i en DTD, uttrykke slike regler i fullt omfang. En fordel ved å ha eksterne regler er at selve XML-dokumentene kan bli noe enklere, men en klar ulempe er at standardverktøyene for å validere XML i henhold til DTDer eller XML Schema ikke har noen mulighet til å hjelpe med kontroll av om dokumentene overholder slike regler.

MathML er laget for å lett kunne være kompatibel med ulike XML-baserte systemer, blant annet gjennom at alle elementer kan ha et xlink:href attributt. For mer detaljer om XLink se [W3C-XLINK] eller håndboken om XML-relatert teknologi fra O'Reilly, [HAROLD2002].

MathML er delt i to typer markup, en som er best egnet til å kode egenskaper for visning og en som er velegnet til å kode semantikken (meningen) i et matematisk uttrykk. Jeg vil i det følgende se nærmere på de to formatene og forskjeller, likheter og muligheter til å kombinere disse.

3.1. Presentasjonsdelen av MathML

Under ser vi hvordan uttrykket $(x\text{-}y)^2$ kan uttrykkes i presentasjonsdelen av MathML.

```
<mrow>
  <msup>
    <mfenced>
      <mrow>
        <mi>x</mi>
        <mo>-</mo>
        <mi>y</mi>
      </mrow>
    </mfenced>
    <mn>2</mn>
  </msup>
</mrow>
```

Figur 7 – Et uttrykk i presentasjonsrettet MathML

Vi ser at det er elementer for hevet og senket skrift (henholdsvis `msup` og `msub`), samt operatorer (`mo`) og identifikatorer/symboler (`mi`). Elementet `mfenced` angir noe som skal grupperes (i parenteser for eksempel) og `mrow` brukes til horisontalt etterfølgende materiale.

3.2. Representasjonsdelen av MathML

Vi tar nå fatt i samme uttrykk som ble brukt i eksemplet fra presentasjonsdelen av MathML og vil se hvordan dette tar seg ut når man vil ha frem meningen til uttrykket ved bruk av MathML. Her er altså uttrykket $(x\text{-}y)^2$ satt opp i content-delen av MathML:

```
<mrow>
  <apply>
    <power/>
    <apply>
      <minus/>
      <ci>x</ci>
      <ci>y</ci>
    </apply>
    <cn>2</cn>
  </apply>
</mrow>
```

Figur 8 – Et uttrykk i semantisk rettet MathML

Vi ser at notasjonen over kan minne mye om prefiksnotasjonen for funksjonskall i funksjonelle språk som Lisp eller Scheme. Vi ser også at barneelementenes plassering kan være signifikant og at det opplagt er regler for tolkning utover det som MathML arver fra XML.

3.3. Kombinasjon av presentasjon og semantikk i MathML

Man kan i MathML hekte på semantikk til presentasjonsutrykk eller omvendt. Det er ulike regler for hvordan dette kan gjøres, man kan spesifisere utseende til bare et deluttrykk eller ha helt parallelle trær for mening og formatering. Under følger et eksempel der man gjør

sistnevnte, altså har to deler i parallell for samme uttrykk, der vi ser at $(a+b)(c+d)$ i denne sammenheng er å tolke som et boolsk uttrykk.

```
<semantics>

  <mrow>
    <mrow><mo>(</mo><mi>a</mi> <mo>+</mo> <mi>b</mi><mo>)</mo></mrow>
    <mo>&InvisibleTimes;</mo>
    <mrow><mo>(</mo><mi>c</mi> <mo>+</mo> <mi>d</mi><mo>)</mo></mrow>
  </mrow>

  <annotation-xml encoding="MathML-Content">
    <apply>
       <and/>
       <apply>
          <xor/><ci>a</ci> <ci>b</ci>
       </apply>
       <apply>
          <xor/><ci>c</ci> <ci>d</ci>
       </apply>
    </apply>
  </annotation-xml>

</semantics>
```

Figur 9 – Et uttrykk med kombinert mening og presentasjon i MathML

Eksemplet over er basert på et eksempel i [W3C-MATHML2]. Det er også verdt å legge merke til at eksemplet over eksplisitt koder meningen i det matematiske uttrykket, noe som det ville vært svært vanskelig å dekode automatisk fra presentasjonsdelen (man måtte i alle fall ha visst noe om konteksten for å gjette på at det var et boolsk uttrykk og ikke for eksempel en numerisk multiplikasjon).

Dette har fordelen at man kan si noe om begge de to fundamentale behovene for matematiske uttrykk, men som vi ser blir det verbost (i praksis omtrent dobbelt så langt) og vi kan se for oss problemer ved at man ved endring må oppdatere to steder som må være synkronisert. Jeg kommer litt mer tilbake til dette i kapittel 5.2 om valg av format for prototypen jeg har laget.

I eksemplet over kan man si at semantiske markup er hektet på presentasjonsdelen. Det er også mulig å gjøre det motsatt, altså at presentasjonsdelen legges til et uttrykk i content-MathML.

4. Eksisterende løsninger og komponenter

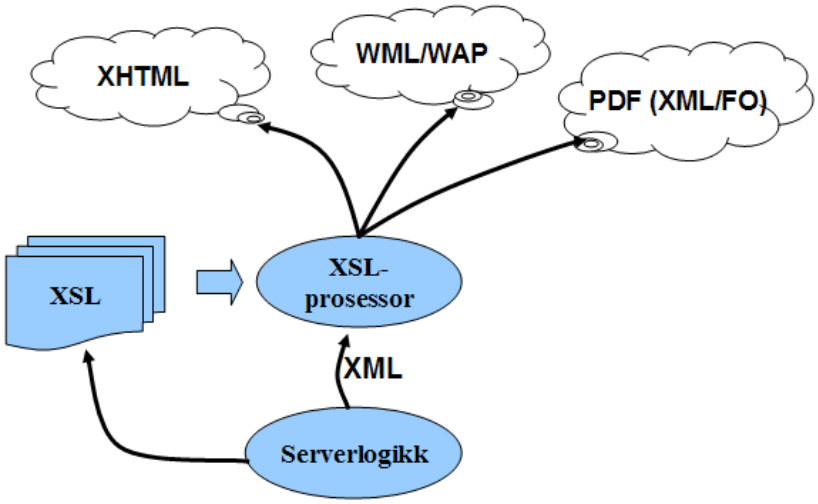
Som nevnt i kapittel 2.2 gjør krysspublisering at man kan *oppdatere kun ett sted* og likevel *publisere i flere kanaler* eller format, siden samme kilden brukes til å produsere de ulike sluttformatene. Inspera har et system for krysspublisering av tekstbaserte dokumenter, men har ingen støtte for avansert matematisk sats eller matematiske grafer.

I dette avsnittet vil jeg kort se på offentliggjorte modeller (og programvare) for krysspublisering generelt og for MathML spesielt.

4.1. Krysspublisering

Det finnes mange ulike rammeverk, produkter og metodikker for krysspublisering, både kommersielle og åpne. Felles for mange av disse er at de benytter en form for markeringsspråk i bunn, slik som XML (eller SGML som er forgjengeren[8] til XML). Jeg vil i det videre se litt nærmere på metodikken som ligger til grunn for slike, samt vise til noen konkrete eksempler på slike systemer.

Under følger en figur som viser den generelle arkitekturen i mange slike løsninger, der transformasjonen skjer på serversiden[9]. Som vi ser følger denne en lagdelt modell, med XML som et bindende element.



Figur 10 – Teknisk arkitektur for krysspublisering

På figuren er det indikert at valg av mal i form av XSL skjer ved hjelp av logikk i løsningen og tilsvarende at XML produseres dynamisk. Det trenger i prinsippet heller ikke være slik, serverlogikken kan være så enkel som en filbasert sammenslåing av XML og XSL-maler til et sluttformat som WML for mobiltelefoner eller XHTML for weblesere. Arkitekturen over passer også som en beskrivelse av Inspera Content, som er Inspera sitt system og

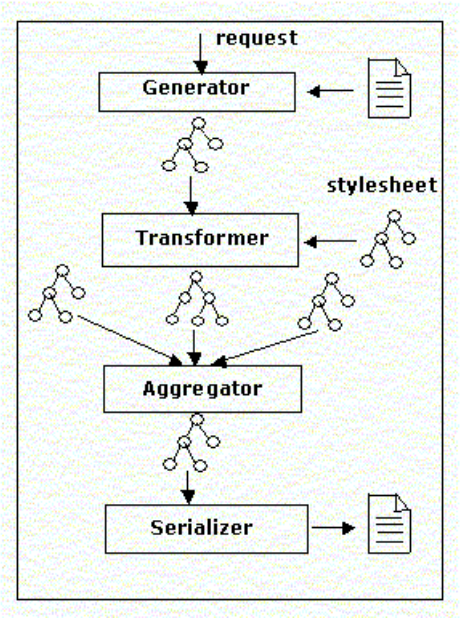
[8] XML er ikke helt teknisk et subsett av SGML, men ganske nær. XML ble i sin tid laget for å være et enklere format med mange av de samme fordelene som SGML, se også tidligere korte avsnitt om XML, kapittel 2.3. For mer utdypende infomrasjon om sammenhengen mellom XML og SGML, se for eksempel [CLARK1997].

[9] Figuren kan godt beskrive systemer som overlater til klienten å formatere XMLen også, da vil bare sluttformatet som leveres være "rå" XML som klienten sørger for å vise brukeren. Et eksempel på dette kan være nyere nettlesere som støtter XSLT direkte i nettleseren slik at transformasjonen kan gjøres der.

rammeverket prototypen skulle lages i. Sluttformatene er kun ment som eksempler og er ikke uttømmende.

Et viktig åpent rammeverk for krysspublisering er Apache Cocoon. Dette bygger på XML og Java og er basert på åpen kildekode og basarmodellen[10] for utvikling uten noen sterk sentralisert styring. Jeg vil kort se litt på dette rammeverket, som ble startet for å dekke behovet for et godt system for webpublisering for Java-prosjektet til Apache, samt påpeke noen fellestrekk ved dette og Inspera sin løsning.

Apache Cocoon er ikke knyttet til noe bestemt dokumentformat, men har en generell arkitektur for oversetting av XML-kilder[11] via XSL-transformasjoner til ulike formater. Selve arkitekturen legger opp til (om man vil; tvinger brukeren til) et strengt skille mellom presentasjon og innhold, som jo er noe av det mest sentrale i et krysspubliseringssystem. Cocoon kaller den siste oversettingen fra generert XML til sluttformat for serialisering. Under følger en figur fra hjemmesidene, [COCOON2003].



Figur 11 – Håndtering av en forespørsel i Apache Cocoon

Selv om Inspera i sin løsning, som de har kalt Inspera Content, ikke benytter Apache Cocoon har løsningene likevel mange fellestrekk gjennom at de benytter felles biblioteksfunksjonalitet. Dette gjelder for eksempel komponent for XSL-transformasjoner[12], generering av PDF[13] og at de bruker samme teknologi (J2EE) og dermed kan kjøre på de samme serverne.

[10] I motsetning til katedral, uttrykket stammer fra Eric S. Raymond sin nå klassiske artikkel "The Cathedral and the Bazar", også utgitt i boken [RAYMOND2001]. Bakgrunnen for uttrykket er at organiseringen av utviklingen med mange jevnbyrdige utviklere og frie diskusjoner kan minne om en støyende basar, mens den mer strengt hierarkiske modellen for organisering som følges i mange kommersielle organisasjoner. Denne kan på sin side minne mer om en katedral der man sier noe kun når man blir oppfordret til det og presten er suveren i sine valg.

[11] XML-kildene trenger ikke bare være XML-filer slik man kanskje først tenker på (selv om de *kan* være det), Cocoon opererer med begrepet Generator om kildene og disse kan generere XML både fra fil, database og helt andre kilder.

[12] Komponenten som benyttes er Apache Xalan, se http://xml.apache.org/xalan-j/.

[13] Komponenten som benyttes er FOP, se http://xml.apache.org/fop/.

Et viktig element i et krysspubliseringssystem er dokumentformatet som brukes til lagring (eller i alle fall utveksling, enkelte systemer kan bruke proprietære løsninger til selve lagringen). Dokumentformatet er ofte spesifisert i form av en DTD (se [HAROLD2002] for detaljer om hvordan syntaksen i en DTD er), altså en definisjon av dokumenttypen eller dokumentformatet om man vil. Et av de mest kjente formatene, spesielt utbredt innen teknisk dokumentasjon, er DocBook. Formatet er også brukt til mange bøker, spesielt O'Reilly forlag bruker dette mye. Den siste versjonen av DocBook-DTDen for XML er versjon 4.2, se [DOCBOOK4.2] eller [DOCBOOK-OASIS] for detaljer om vedlikeholdet av formatet.

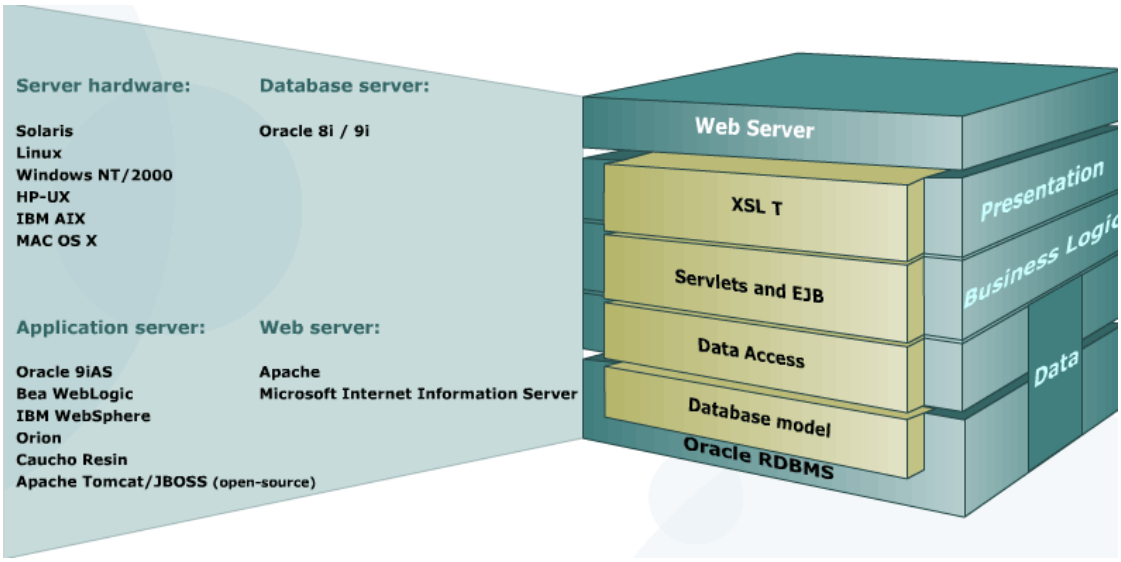
Inspera vurderte i sin tid nøye om de skulle bruke DocBook eller en variant av denne i sitt system. Dette valgte man å ikke gjøre, så XML-formatet i Inspera sitt system er proprietært og følger en egendefinert DTD. Litt mer om dokumentformatet[14] til Inspera og bakgrunnen for denne avgjørelsen finnes i en rapport fra hovedprosjekt ved Høgskolen i Bergen våren 2001, [BJØRNEBEKK2001].

Hovedtrekkene i begrunnelsen var at det var noen spesielle behov som ble for lite praktisk å i ivareta ved bruk av DocBook (for eksempel innknytning av animasjoner og illustrasjoner laget i Macromedia sitt Flash-format), samt at det for helt enkle dokumenter var enklere med det egendefinerte dokumentformatet (som kanskje ligger et sted mellom DocBook og XHTML, men nærmere DocBook og med en del spesielle utvidelser). I tillegg står man friere ved utvidelser siden man ved utvidelser av et definert format fort vil innføre inkompabiliteter (nå er det i alle fall tydelig og oversetting må skje eksplisitt).

Til tross for valget av proprietært format har DTDen såpass mange fellestrekk med DocBook at utveksling av de fleste dokumenter ikke er noe problem. Dette gjøres i Insperas system ved en oversetting i form av en XSL-transformasjon ved import/eksport slik at systemet kan ta inn data i form av (et lite subsett av) DocBook.

Komponentene i n-lagsarkitekturen i Insperas løsning kan illustreres som under. Her går det også frem litt om de ulike produsenter/leverandører som finnes til de ulike lagene (spesielt i valg av applikasjonsserver står man relativt fritt så lenge man benytter standardiserte grensesnitt i J2EE).

[14] Dokumentformatet er ikke identisk per i dag som det var den gangen, faktisk har det vært et par radikale endringer og flere mindre utvidelser. Dokumentformatet til Inspera er dog ikke noe stort poeng i denne oppgaven, da problemstillingene som behandles vil gjelde uavhengig av XML-formatet som benyttes i systemet.



Figur 12 – Insperas n-lagsarkitektur

Det er innenfor dette rammeverket og med disse teknologiske valgene at utviklingen av prototypen ble gjort. Inspera bruker begrepet Inspera Platform om sin tekniske løsning og inkluderer da basisdatamodellen (som vi skal se litt på senere) og Javaklasser for denne, samt klasser for håndtering av forespørsler.

Andre rammeverk som kan være verdt å nevne er Zope (Python-basert), Velocity (også Java-basert, noe overlapp med Cocoon, men opererer på et lavere abstraksjonsnivå) og JPublish (igjen Java-basert, her er abstraksjonsnivået høyere og man kan bruke Velocity som en malmotor i JPublish). Det finnes også mange varianter av publiseringsløsninger for PHP og Perl, der norske eZ Publish som benytter XML i lagringen virker å være blant aktørene med en relativt gjennomtenkt teknisk arkitektur (basert på PHP), selv om krysspublisering bare til dels er støttet.

4.2. MathML

MathML-verktøyene er i stadig utvikling og mange av disse, kanskje spesielt tilbudene med åpen kildekode, er i en relativt tidlig utviklingsfase. Dette bekreftes også av implementasjonsrapporten fra W3C angående MathML ([W3C-MIMP2001]), samtidig som man i samme dokument også ser at det er stor aktivitet og begynner å bli en del tilbydere. Jeg gjengir ikke innholdet i rapporten i detalj her, men interesserte lesere henvises til W3C sine websider for utdypende informasjon.

Jeg kommer litt tilbake til noen av tilbyderne som er beskrevet i implementasjonsrapporten i forbindelse med arkitekturen for prototypen jeg har laget, i kapittel 5.4.

Det er en løpende utvikling innen verktøy for matematikk på web. Siden rapporten nevnt i forrige avsnitt ble skrevet har flere verktøy kommet til og mange av verktøyene har blitt videreutviklet noe. [W3C-MIMPWEB] gir en mer løpende oppdatert status på verktøyfronten. [DESSCI-MOW] gir en god analyse av ulike behov i forbindelse med publisering av matematikk på web og tilhørende løsninger. Selve anbefalingene i denne artikkelen må selvsagt leses i lys av at artikkelen er publisert av en verktøyleverandør, som dermed neppe kan sies å være nøytral.

Det bør også nevnes at det finnes strategier for krysspublisering av matematikk som kun bruker MathML for visning i nettleseren (eller ikke i det hele tatt). Et typisk eksempel på dette er løsninger som bruker en variant av T_EX som lagringsformat og genererer HTML med bilder eller presentasjons-MathML for de matematiske uttrykkene for webbruk og postscript eller PDF til utskrift. [GURARI2000] presenterer en slik løsning (demonstrert på en MathML-konferanse i 2000).

5. Utforming av arkitektur for prototypen

5.1. Krav som må støttes

Først skal vi se på noen generelle krav til god arkitektur for løsningen. Dette henger også sammen med hvordan rammeverket som prototypen er implementert i fungerer. Deretter kommer en gjennomgang av spesifikke krav til matematikkløsningen, herunder en kort gjennomgang av krav fra ulike interessenter/brukergrupper.

5.1.1. Generelle krav

En del av kravene jeg går inn på her er midler for å oppnå målene jeg gikk inn på i avsnitt 2.2 og ikke mål i seg selv. Men at løsningen er teknisk elegant vil også gjøre sannsynligheten for at brukerne er fornøyde større, da videreutvikling blir enklere og sjansen for feil mindre.

Prototypen implementeres i et objektorientert system. Et viktig prinsipp innen objektorientert design er fornuftig bruk av arv. Ved å ha en generisk objekttype som innehar egenskapene som er felles for alle objekttyper, kan mange ulike typer objekter behandles likt – de *er* jo også et generisk objekt.

Typiske eksempler på slike fellesegenskaper i et krysspubliseringssystem vil være logging av bruk, muligheter til å tilknytte metadata, søkbarhet og rettighetsstyring.

I et krysspubliseringssystem bør det være kun ett sted å oppdatere informasjonen, selv om systemet har mange sluttformater. Det er en fordel om man har mulighet for revisjonskontroll slik at man kan rulle tilbake til en tidligere versjon dersom det er ønskelig. Det kan også være klokt å skille språkversjoner og logisk identitet slik at informasjonsobjekter i systemet kan forespørres på logisk identitet og leveres brukeren på det språket som passer best med hans preferanser.

Et skille mellom innhold og maler for utseende er en selvfølge i et krysspubliseringssystem (for å få til ulike sluttformater fra samme kilde, eller innhold, må jo i praksis dette støttes). Likevel er dette i seg selv et viktig poeng, spesielt i sammenligning med andre løsninger som ikke har fokus på krysspublisering.

5.1.2. Spesifikke krav for matematikkløsning

For å finne frem til kravene for prototypen vil jeg forsøke å se på brukerne av et system som prototypen vil være en del av og deres krav. Jeg ser på to brukergrupper som nok vil vektlegge litt ulike ting. Den første brukergruppen er den som publiserer det matematiske innholdet, for eksempel representert ved forfattere og redaktører i Aschehoug. Den andre brukergruppen er sluttbrukere av det publiserte materiale og tilhørende verktøy. I praksis vil dette si elever i videregående skole i Norge.

I tillegg til de to brukergruppene som er nevnt vil Inspera (som kom med opprinnelig initiativ og oppgavetekst) være både direkte interessent og det vi kan kalle en metainteressent. I det ligger det at de både setter krav og gir premisser, men også er interessert i de to andre gruppenes interesser.

Aschehoug har i flere år drevet flermedial publisering av XML-basert innhold sammen med Inspera. De vil også i denne sammenheng legge vekt på at det publiserte innhold kan tas ut både for utskrift (i form av PDF-dokumenter) og for visning på web. For Aschehoug vil også brukervennlighet ved innlegging av matematisk innhold spille en stor rolle, men dette er i stor grad definert som utenfor denne oppgaven.

Aschehoug vil også at man skal stå så fritt som mulig med tanke på eventuelt bytte av tredjepartsleverandør for visning av uttrykk og at man skal ha muligheten til senere å utvide til å benytte flere slike. Dermed er det viktig at formatet som velges beholder meningen i de matematiske uttrykkene, samtidig som det vises som ønsket.

For elevene i den videregående skolen vil noen av de samme tingene være nyttige (ta ut oppgaver o.l. i utskriftsvennlig form i tillegg til at de kan se de på web). Det er grunn til å tro at det for denne gruppen er viktig med dynamikk og at de selv enkelt og interaktivt kan gjøre sine tilpasninger. Selv om man neppe klarer å få alle til å synes matematikk er gøy (selv på web!), kan man håpe at slike muligheter kan virke læringsfremmende. Ifølge en undersøkelse gjort av det norske e-læringsselskapet Fronter ([FRONTER2003]) synes 8 av 10 at direkte tilbakemelding er viktig i et e-læringssystem[15].

Graftegneren skal bruke de samme matematiske uttrykkene (lagret i systemet) som også brukes for symbolsk visning. Slik kan samme uttrykk vises både i symbolsk form og som graf, gitt at det har en grafisk presentasjon. I tillegg skal også grafer kunne inkluderes både i websider (med de dynamiske mulighetene forventet av brukerne som forklart over) og direkte i PDF-dokumenter.

Insperas krav og forventninger kommer frem blant annet i oppgaveteksten. Inspera har i tillegg til de indirekte interessene nevnt over også interesse av de tekniske aspektene ved løsningen, herunder ved at de har rollen som verktøyprodusent og kanskje senere skal tilby løsninger basert på teknologien som publiseres som en del av denne oppgaven. Dette blir tatt hensyn til ved at Insperas teknologiske plattform brukes for implementasjon av prototypen, samt at hele kapittel 7 gjør en vurdering av ytelse og skalerbarhet som er nyttig med tanke på beslutninger rundt drift av løsningen. Inspera har også en uttalt politikk på at man vil støtte standarder og sette så få spesifikke krav til program- og maskinvare hos sluttbrukeren som mulig.

En kort oppsummering av kravene fra de tre definerte gruppene interessenter blir altså:

- Matematiske uttrykk må både kunne vises i symbolsk form og som graf
- Det er en fordel om graftegneren kan tilby brukeren dynamikk og muligheter for interaksjon
- Matematiske uttrykk må kunne tas ut i versjoner både for utskrift (og trykk) og web
- Den tekniske løsningen må være kompatibel med Inspera sin plattform og det skal settes så små krav til spesiell programvare som mulig
- Det må legges til rette for at et bytte av tredjepartskomponenter kan skje relativt enkelt

[15] For det første vet jeg ikke hvor statistisk korrekt denne undersøkelsen er utført og for det andre er min ekstrapolering fra at elever liker direkte og interaktiv tilbakemelding til at graftegneren må gi slik direkte i klienten i beste fall litt tynn. Imidlertid virker det også intuitivt riktig å anta at en generasjon vant til spill på nett og mobiltelefoner og graftegnere på kalkulatoren vil forvente dynamikk av en slik graftegner i nettleseren sin også.

5.2. Valg av lagringsformat for matematiske uttrykk

At W3C sitt MathML-format skulle velges i prototypen var nærmest et premiss for denne oppgaven (se oppgavetekst i appendiks A). Det var imidlertid ikke sagt noe om det var content- eller presentation-varianten som skulle velges. Under redegjør jeg for fordeler og ulemper ved de to, samt presenterer og begrunner det valget jeg har gjort. Jeg gir også en kort sammenligning med kanskje det eneste reelle alternativet til MathML for XML-basert lagring av semantikken i matematiske uttrykk, nemlig OpenMath-formatet. Det er fullt mulig å benytte en kombinasjon av OpenMath og presentasjonsrettet MathML.

OpenMath-formatet har sterkt fokus på at det skal være utvidbart. Dette gjelder for så vidt også (content-)MathML, men mens det i MathML finnes det enklere syntaks for de mest brukte uttrykkene (forenkelt kan vi si ”alt innenfor gymnasmatten”), skrives alt i OpenMath med konsekvent syntaks. Dette gjør at OpenMath-syntaksen blir mer verbos enn MathML for de mest brukte uttrykkene, mens utvidelser i MathML ikke blir like elegant og konsekvent syntaksmessig som de blir i OpenMath. Under ser vi et fragment fra de to ulike metodene for å kode matematisk semantikk i XML-format som illustrerer forskjellen, uttrykket er $(a+b)*c$.

```
<!-- Content MathML -->
<apply>
   <times/>
   <apply>
      <plus/>
      <ci>a</ci>
      <ci>b</ci>
   </apply>
   <ci>c</ci>
</apply>

<!-- OpenMath -->
<OMA>
   <OMS name="times" cd="arith1"/>.
   <OMA>
      <OMS name="plus" cd="arith1"/>
      <OMV name="a"/>
      <OMV name="b"/>
   </OMA>
   <OMV name="c"/>
</OMA>
```

Figur 13 – Sammenligning av Content MathML og OpenMath for samme uttrykk

Det er altså slik at både semantisk MathML og OpenMath har et generelt element for generelle funksjonsapplikasjoner (i videste forstand), henholdsvis `apply` og `OMA`. Forskjellen er at OpenMath bruker en helt generisk syntaks der attributene identifiserer operatoren, mens MathML som forklart i forrige avsnitt har egne elementer for de mest brukte operatorene, her `plus` og `times`.

OpenMath inneholder ikke en egen presentasjonsdel, men her kan man enten bruke presentasjonsrettet MathML direkte (tilsvarende som vist tidligere ved kombinasjon av content- og presentation MathML, men da med OpenMath i den semantiske delen) eller foreta en transformasjon til dette ved visning.

At MathML er en standard utgitt av W3C ser jeg også på som en fordel, særlig siden fokus primært er på web som leveringsmåte av innholdet og Inspera i tillegg benytter mest mulig W3C-standarder i sin teknologiske plattform. Settet av matematiske uttrykk som skal brukes i

systemet i første omgang vil også dekkes av de ferdigdefinerte elementene i MathML, siden det nettopp er elever i den videregående skolen som vil være første målgruppe. Verktøystøtten for MathML ser også ut til å være på god vei, se mer om dette i kapittel 5.4.

Det skal også nevnes at valget av markupspråk ikke er irreversibelt, da det finnes ulike metoder for oversetting mellom de to (bl.a. via XSLT), som jo begge er XML-formater (slik at den underliggende teknologien heller ikke ville måtte endres radikalt).

For komplette dokumenter med matematikk vil man også vurdere formater der både MathML og OpenMath kan bygges inn. OMDoc[16] (Open Mathematical Documents) er et eksempel på et slikt format med en publisert DTD, se [OPENDOCWEB]. [KOHLHASE2003] gir en beskrivelse av versjon 1.1 av DTDen og gir i tillegg både sammenligninger mellom OpenMath og MathML og tanker som minner mye om semantisk web-tankegang[17] (automatisk resonnering og bevissjekking, klassifisering og generalisering fra eksempler, samt semantiske søk i stedet for tekstbaserte).

Selve produksjonen av MathML før innleggingen i systemet var definert som å være utenfor denne oppgaven. En lisensiert en komponent fra Design Science benyttes til innlegging av uttrykkene, denne støtter både den presentasjonsrettede og den meningsbevarende versjonen av MathML. Selv om støtten for content-MathML ikke er helt komplett i denne komponenten, har ikke dette hatt noen avgjørende innflytelse på valget av lagringsformat.

WebT_EX er en variant av LaT_EX sitt matematikkformat som støttes av redigeringskomponenten fra Design Science. Dette medfører at matematisk innhold i LaT_EX-syntaks enkelt kan importeres i systemet når denne komponenten benyttes. Selv om syntaksen er svært lik på LaTEX sin, er den underliggende informasjonsmodellen basert på MathML. Dette medfører enkelte inkompabiliteter med LaT_EX, spesielt knyttet til tokenisering (hvordan et uttrykk blir delt opp når det ikke er eksplisitt gruppert) og tabeller/matriser.

Jeg har valgt å forsøke å benytte parallell semantisk- og presentasjonsrepresentasjon av de matematiske uttrykkene i min prototype. Dette for å være best mulig gardert mot fremtidige krav slik at man ikke risikerer at de innlagte matematiske uttrykk må legges inn på nytt.

Ved å velge en kombinasjon av de to delene av MathML er man også mer sikker med tanke på fremtidige bytter av verktøy. Man kan også ved visning (og ved beregning) bruke den delen som egner seg best. Ulempen er selvsagt at man ender opp med et større uttrykk som må lagres, samt at man ved oppdatering må sørge for å oppdatere begge steder. Sistnevnte argument er ikke spesielt farlig så lenge man bruker et redigeringsverktøy, det blir bare ett sted å endre for brukeren. Når det gjelder det førstnevnte tror jeg heller ikke her problemet er stort, siden XML-formatet kun benyttes på serversiden. Det er enten det ferdiggenererte bildet som sendes brukeren eller man kan, om man vil sende MathML direkte til klienten, plukke ut bare den ene halvdelen for oversending dersom det er et poeng å få ned overføringstiden.

[16] Kort fortalt tillater DTDen både bruk av MathML og OpenMath, den benytter Dublin Core til sin metadata-modul og den har også moduler for både øvinger/quiz og logikk og resonnementer. Formatet brukes blant annet aktivt på Carnegie Mellon University hvor også koordinatoren for prosjektet har sin arbeidsplass.

[17] Så vidt meg bekjent ble uttrykket presentert i [BERNERS-LEE1998], selv om det sikkert har vært diskutert i ulike fora lenge forut for dette. Det er jo også sterkt relatert til tanker innen AI (selv om det presiseres i nevnte notat at AI ikke er noen forutsetning for maskinlesbare formater for semantikk) som i alle fall har sin bakgrunn fra tidlig 80-tall. For en noe mer popularisert og inspirerende introduksjon til semantic web, se [BERNERS-LEE2001].

En siste vurdering som må gjøres ved valg av format er om man ønsker å inkludere matematiske uttrykk direkte i lagrede dokumenter (såkalt inline) eller om man ønsker å referere til matematiske uttrykk fra dokumentene, mens selve XMLen for det matematiske uttrykket lagres som en separat logisk enhet i databasen.

Ved en inline lagring lagres det matematiske uttrykket sammen med dokumentet som inneholder uttrykket. Dette gjør at man ikke kan redigere uttrykket uten å samtidig redigere dokumentet, disse er da fast sammenkoblet. Man vil heller ikke kunne bruke samme uttrykk flere steder uten å da måtte oppdatere alle ved endringer (dette kan også være en fordel, dersom man bare ønsker å endre den ene kopien). Dokumentet lagres helhetlig slik at import og eksport blir enklere.

Ved separat lagring av det matematiske objektet (med kun en logisk referanse til dette i XML-dokumentet som bruker uttrykket) vil man få noen flere innholdsforekomster i systemet. Dette kan imidlertid også være en fordel ved at det matematiske uttrykket som sådan får en logisk identitet og kan redigeres frittstående. Dette siste gjør også at det matematiske uttrykket kan vises separat, uten noen kontekst. For Insperas system vil dermed matematiske uttrykk bli helt på linje med Flash-illustrasjoner, bilder og videofiler – altså objekter man kan sette inn i dokumenter og oppdatere frittstående og uavhengig av hvor det er brukt.

At matematiske objekter får en egen identitet og kan brukes uten noe bestemt dokument er også en fordel med tanke på å bruke MathML-basert lagring av uttrykk som skal benyttes av graftegneren. Samme uttrykk kan da også inkluderes med forskjellige visning, én i symbolsk form og én i form av en graf.

Punktet om konsekvens i systemet, samt at matematiske uttrykk får sin egen identitet og kan brukes uten noe tilhørende tekstlig dokument veier for meg tyngst, og jeg har derfor valgt å lagre matematikkobjektene separat fra innholdsdokumentene som bruker dem.

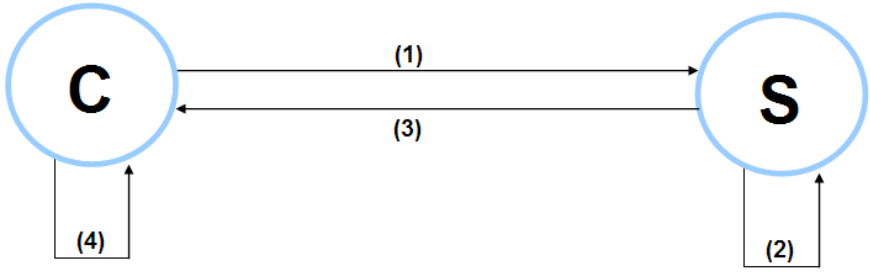
Ut fra resonnmentet og valgene gjort over, kan vi kort oppsummere følgende valg gjort for dokumentformatet:

- Å lagre både presentasjon og semantikk eksplisitt
- Å bruke MathML som format, med de to ulike delene til de to ulike behovene
- Å lagre de matematiske informasjonsobjektene adskilt fra anvendelsen i et bestemt dokument

5.3. Ansvarsfordeling mellom klient og tjener

Ut fra oppgavetekst og rammebetingelser var det allerede gitt en del retningslinjer for teknisk arkitektur. Klienten skulle være en nettleser (uavhengig av operativsystem) og på serversiden var det J2EE (Java) som var valgt. For en generell introduksjon til klient/tjener modell henvises det til for eksempel kapittel 4.4 i [COULOURIS2001]. Her går jeg bare inn på designvalgene gjort ved arbeidet med denne oppgaven og bakgrunnen for disse.

Helt generelt kan man se på prosessen ved håndtering av en forespørsel som delt opp som vist i figuren under.



Figur 14 – En forespørsel i et generelt klient/tjenersystem

Prosessen over består altså av

(1) Forespørsel fra klienten til tjeneren
(2) Prosessering på tjeneren
(3) Svar fra tjener til klienten
(4) Prosessering og visning i klienten

En slik helt overordnet beskrivelse kan synes å være lite nyttig, men jeg vil forsøke å vise hvordan valgene i arkitekturen påvirker kompleksiteten i hvert av punktene. I de følgende avsnittene vil jeg derfor referere til punktene over, samt dele opp funksjonalitet innen disse rammene. Ofte er det mulighet til å plassere deler både i punkt (2) og punkt (4) fra oversikten over og begge valgene har sine fordeler og ulemper.

To ting det kan være verdt å merke seg er at:

- En operasjon som for en bruker later til å være en operasjon/forespørsel kan innebære mange forespørsler i henhold til modellen over.
- Steg (4) kan i enkelte tilfeller være svært omfattende (selv om vi begrenser oss til webapplikasjoner, noe vi ikke har gjort i modellen så langt – men det er innenfor denne rammen prototypen skal lages) med interaksjon fra brukeren og nye visninger basert på disse. Således kan også én forespørsel i henhold til modellen oppleves som flere operasjoner sett fra brukerens ståsted.

Som et grunnleggende prinsipp har jeg holdt på at det skal legges så små begrensninger på sluttbrukerens nettleser, operativsystem og forhåndsinnstillinger som mulig. Dette stemmer også med kravene satt opp i kapittel 5.1. For å få til dette er det en fordel å gjøre så mye som mulig av prosesseringen på serveren. Ulempen med å gjøre prosesseringen på serveren er selvsagt at det setter større krav til serveren og kan gi dårligere ytelse. Dette kommer jeg tilbake til i større dybde i forbindelse med ytelsestesten senere i denne rapporten.

En annen ulempe med å gjøre prosessering på server er at mulighetene for interaktivitet blir dårligere, siden det må sendes en ny forespørsel til serveren for hver endring brukeren eventuelt gjør som skal gjenspeiles i hva som vises.

5.4. Arkitektur for visning i form av symbolske uttrykk

Systemet skal vise matematiske uttrykk på symbolform både på web og for utskrift og er derfor avhengige av å ha et format som kan inkluderes i automatisk produksjon av PDF-filer, i tillegg til å vises i en nettleser.

Et format som kunne egne seg for å støtte kravene til krysspublisering og god kvalitet i utskrift er SVG. Dette egner seg godt for trykk/papirutskrift siden det er et vektorformat (skrivere har mye større oppløsning enn skjermer, dette gir problemer for punktbaserte formater). Problemet med SVG er at ikke alle nettlesere har støtte for dette direkte og at visningsprogramvaren ikke er like velutviklet på alle plattformer.

Selve oversettingen fra MathML til symbolform i et visningsklart format på serveren gjøres av en tredjepartskomponent som integreres som en del av implementasjonen av prototypen. Når det gjelder valg av denne tredjepartskomponenten, har jeg vurdert en komponent fra IBM kalt techexplorer[18], en fra Design Science[19], en fra Schemasoft[20] og JEuclid[21] som er et prosjekt med åpen kildekode. Disse bruker litt ulike metoder for oversetting, som jeg kort vil presentere sammen med en kort vurdering av komponenten.

IBM techexplorer er et produkt som støtter visning av MathML og også andre formater som LaTeX. Det støtter avansert interaksjon og styring av komponenten (som teknisk er løst som en såkalt plugin til nettleserne). Imidlertid passer ikke produktets egenskaper med kravet om at sluttbrukerne ikke skal trenge noen stor lokal installasjon, siden installasjonen krever nedlasting av 4-10MB avhengig av operativsystem. Produktet er proprietært og har lisenskostnader per sluttbruker. For et mer spesialisert bruksområde med mer avanserte brukere ville produktet vært mer egnet, men med tanke på allemenn bruk mot elever i videregående skole er det ikke aktuelt. Det er også lite sannsynlig at krysspubliseringsaspektet støttes av komponenten, siden denne har hovedfokus på funksjonalitet i klienten via en plugin og ikke uten videre kan brukes på serveren for å produsere bilder for innebygging i dokumenter for utskrift.

Design Science sin løsning er proprietær og leverer PNG eller JPG direkte via et definert API. At den er properitær, med høye kostnader for kommersiell bruk og har lukket kildekode er en klar ulempe. Fordelen med komponenten er at den leverer gode resultater[22] og at teknologien bak er velprøvd for eksempel ved at samme firma leverer formelredigering til tekstbehandlingsprogrammet Microsoft Word. Sluttformatene den leverer (i alle fall JPG) støttes av alle relevante nettlesere. Løsningen er også fleksibel i det at den i tillegg til

[18] Produkthjemmesiden er http://www-3.ibm.com/software/network/techexplorer/.
[19] http://www.dessci.com/ er Design Science sin hjemmeside.
[20] http://www.schemasoft.com/ er Schemasoft sin hjemmeside, nevnte komponent er omtalt spesielt på adressen http://www.schemasoft.com/MathML/.
[21] http://jeuclid.sourceforge.net/ er hjemmesiden til prosjektet og informasjon om siste aktiviteter (eller mangelen på sådanne) får man oversikt over på oversiktssiden http://sourceforge.net/projects/jeuclid/.
[22] Hvorvidt resultatene er "gode" blir selvsagt noe subjektivt og det har ikke vært en vesentlig del av prosjektet å evaluere disse tredjepartskomponentene, så noen formelle kriteria for vurderingen er heller ikke satt opp. Imidlertid synes det klart at det er en kvalitetsforskjell, for eksempel ved at noen av komponentene ikke klarer å skille de ulike delene helt fra hverandre og i en del tilfeller tegner tall oppå symboler osv.

komponentene for å lage grafikk fra MathML på serveren også støtter visning av uttrykk via en applet hos klienten. I sistnevnte tilfelle trenger serveren bare levere MathML og klienten tegner opp dette selv.

Schemasoft sin er løsning er også proprietær og basert på en direkte oversetting av MathML til SVG via XSLT. Schemasoft har samme ulempe som komponenten nevnt over i at den er lukket kildekode som ikke kan endres. I tillegg er ikke resultatene per i dag gode nok, siden enkelte symboler kan havne oppå hverandre i resultatet.

Selv om det på sett og vis er elegant å oversette fra et XML-format til et annet (SVG er som nevnt også et XML-format) via XSLT blir XSL-filen ganske stor og det blir svært mange spesialbehandlinger. XSLT egner seg svært godt for oversetting mellom formater der det er en tilnærmet en-til-en mapping mellom elementene i formatene, et eksempel på dette er oversetting mellom Content-MathML og OpenMath-formatene. Man skal etter min mening likevel ikke bli fristet til å misbruke det faktum at det finnes et komplett programmeringsspråk i XSLT-standarden. Totalt sett gjør dette at jeg ikke uten videre vil anbefale metoden for bruk i en serversideløsning med mange brukere. For eksperimentering lokalt på egen PC kan likevel komponenten være god.

JEuclid gjør en tilsvarende oversetting som Schemasoft sin komponent. Imidlertid gjøres her oversettingen direkte i Java. JEuclid leverer også sluttresultatet i SVG. Dessverre er ikke resultatet godt nok per i dag (blant annet er det en del tegning av objekter oppå hverandre), men det kan være spennende å følge prosjektet fremover.

For både Schemasoft og JEuclid gjelder generelle fordeler og ulemper med SVG-formatet. Dette har jeg kort gått inn på i begynnelsen av dette kapitlet. Min konklusjon per i dag er at til tross for at SVG er bedre egnet for utskrift er det ikke utbredt nok på tvers av plattformer til å være et førstevalg blant formatene per i dag. Selv om krysspublisering er en viktig del av løsningen, er det nok slik at det er websidene som vil bli brukt klart mest og alle nettlesere (på alle operativsystemer) har altså ikke støtte for SVG per i dag. Når løsningene som leverer SVG heller ikke kan sies å være helt modne for produksjonsbruk blir konklusjonen å sitte på gjerdet med tanke på å ta i bruk disse.

Jeg har på bakgrunn av det ovenfornevnte altså valgt å benytte komponenten fra Design Science i prototypen, dette først og fremst fordi det nok er websidene som vil brukes mest og ikke alle nettlesere har SVG-støtte per i dag. Denne skal da brukes til å generere grafikk som kan vises både i dokumenter på web og innlemmes i PDF-filer.

Siden et bytte av komponent er høyst aktuelt når tilbudet blir bedre og/eller SVG-støtten i nettlesere er mer utbredt er det viktig at arkitekturen er slik at bytte av komponent er svært enkelt. Jeg presiserer hvordan jeg har løst dette sammen med klassediagrammet for Javadelen av løsningen.

På sikt er det også en mulighet for at nettleserstøtten (enten direkte eller via såkalte plugins) for MathML blir så god at oversetting til andre formater bare blir nødvendig for den delen av løsningen som skal levere resultat til PDF-produksjon. Dette vil i så fall bli svært enkelt å implementere siden det jo er MathML som er lagringsformatet som er valgt i prototypen. Hovedfordelene med en slik løsning er at det avlaster serveren i det at steg (2) i den generelle oversikten på side 27 blir enkel samt at det gir muligheter for at sluttbrukeren også kan utnytte

det faktum at han får tilgang til det matematiske uttrykket og ikke bare til en representasjon av det i form av et bilde.

5.5. Arkitektur og teknologi for graftegneren

Som nevnt tidligere har jeg for graftegneren valgt å basere meg på punktverdier for uttrykk og plotting av disse. Systemet skal vise grafer både på i nettleser og beregnet for utskrift, altså må sluttformatet (eller -formatene) egne seg både for inkludering i PDF og for visning i nettleseren, akkurat som for de symbolske uttrykkene.

Prinsipielt kan vi dele prosessen med å tegne en graf opp i følgende trinn:

a) Hente matematisk uttrykk fra database
b) Oversette uttrykk til en form som kan beregnes
c) Bergene punktverdier
d) Plotte punktverdier
e) Produsere sluttformat

Dette blir altså hva stegene (2) og (4) fra modellen for den generelle interaksjonen mellom klient og tjener til sammen må utføre. Generelt kan man anta at jo mindre som gjøres i steg (2), jo lettere vil steg (3) (overføring av resultatet) bli. Imidlertid er det ikke sikkert at det er slik i alle tilfeller, for eksempel fordi en komprimert bildefil (etter steg e)) kan være mindre enn de rå punktverdiene. Siden punkt a) er nødt til å utføres på server og innenfor praktiske rammer også punkt b)[23], blir det plasseringen av de tre siste delene av funksjonaliteten det er interessant å vurdere nærmere her.

Siden det er et krav at grafene skal kunne tas ut både innebygd direkte i PDF-filer og på web må vi tilby en metode der alle stegene a)-e) kan gjøres i tjeneren. I tillegg var det som vi så i avsnittet om brukernes karakteristika et ønske om at graftegneren skulle tilby dynamisk funksjonalitet slik at brukeren kunne endre parametere interaktivt og få direkte respons fra sine valg, dette innebærer et ønske om det motsatte – altså å flytte en del av logikken til klienten.

Som nevnt i den generelle innledningen er en klar fordel ved å gjøre prosesseringen på klienten at tjeneren avlastes. Dette er spesielt relevant for punkt c) – beregne punktverdier, da dette må antas å ta en del beregningsressurser. Dette taler for å flytte punktet til klienten. Antagelsen om at dette er et krevende punkt beregningsmessig taler også for at løsningen som velges sørger for å beregne punktverdiene så sjelden som mulig. Noe som taler i retning av å gjøre beregningen på server er at man der har kontroll over ressursene (for eksempel kapasitet i prosessor), mens det er mindre forutsigbart og større variasjoner i utstyret til sluttbrukerne.

Den svært enkle analysen av brukernes krav i kapittel 5.1.2 viste et ønske om dynamikk. Dette blir det avgjørende for plasseringen av punktene c)-e) for normalbruk via nettleser. Det er ønskelig at et valg av zoom eller et klikk for å få vite en punktverdi gir svar direkte uten noen ny forespørsel til server. Dette kan kun la seg gjøre dersom logikken i punktene c)-e) utføres av klienten.

[23] Dette fordi vi her krever en fullverdig XML-parser og denne i så fall ville måtte lastes ned til klienten, noe vi av praktiske årsaker neppe ønsker. Dette fordi nedlastingstiden vil bli urimelig lang, selv om det faktisk finnes enkelte XML-parsere som er relativt små. En reversering av valget er uansett enkelt slik den tekniske løsningen for klientdelen til slutt ble, siden også serversideløsningen er laget i Java.

Jeg har altså kommet frem til at punktene c)-e) må tilbys både på serveren (for å støtte krysspubliseringskravet) og direkte i klienten (for å støtte kravene til dynamikk). For å støtte delen i klienten mener jeg at det er fire aktuelle valg med tanke på format/teknologi: Microsoft ActiveX, Dynamisk HTML, Macromedia Flash eller en Javaapplet.

ActiveX valgte jeg tidlig å gå bort fra siden det var et krav at brukeren ikke skulle være avhengig av et bestemt operativsystem. ActiveX-kontroller er som kjent knyttet til Microsoft Windows og vil ikke fungere på andre operativsystemer. I tillegg er det ingen andre nettlesere enn Internet Explorer (sannsynligvis av politiske grunner) som støtter ActiveX selv under Windows. Dermed har heller ikke alle brukere av Microsoft Windows tilgang til slike komponenter. Det hadde heller ikke vært noen fordel å benytte ActiveX med tanke på tid som kreves for implementasjonen, siden utvikling av DLLer i C++ kan være ressurskrevende.

Jeg synes ikke det virker realistisk rent teknisk å skulle få til opptegning av fornuftige grafer i nettleseren bare ved hjelp av JavaScript og HTML, ofte kalt dynamisk HTML. Derfor falt også dette alternativet falt bort tidlig. Det er rett og slett ikke mulig å få til dette fornuftig uten bruk av spesielle komponenter i nettleseren. Dette fordi man har ikke tilgang til tegneprimitiver fra JavaScript sin objektmodell i dagens nettlesere. (Man kunne tenke seg å ”tegne” til et rutenett av etpunktsbilder ved å bytte ut disse fra JavaScript, men dette ville ikke bli verken elegant eller praktisk.)

Macromedia Flash har støtte på flere plattformer og støtter relativt kraftige tegnefunksjoner, så dette ville vært et mer aktuelt alternativ. Flash har også den fordelen at det er et vektorformat og dermed enkelt skalerbart i størrelse. Den store ulempen er at et slikt valg ville medført total duplisering av logikk for serverdel og klientdel (som skulle tilby dynamikken). Som nevnt må vi for å få til krysspubliseringsaspektet også ha logikk på serversiden som muliggjør produksjon av PDF-filer med innebygde grafer.

En Javaapplet har de samme fordelene som Flash, bortsett fra at den ikke direkte støtter endring av størrelsen på tegnefeltet (det vil si: her må programmereren gjøre slike justeringer manuelt). Det er støttet på alle de mest brukte plattformene (og de aller fleste mindre brukte også) og støtten for dynamikk i brukerens nettleser er til stede. Det kan muligens innvendes at brukerterskelen for å lage Flash er lavere, men for undertegnede (som helt klart er mer erfaren som programmerer enn designer) er det likevel lettere å forholde seg til Java hvor jeg både har bred erfaring, utviklingsmiljø[24] og tilgangen på dokumentasjon er god.

Den største fordelen med en løsning basert på en applet for opptegningen i klienten er at man ved en gjennomtenkt klassemodell da kan dele kode mellom serverkomponentene og klientkomponentene som gjør selve tegningen. Dette fordi det allerede var et premiss for løsningen at serversidekomponentene ble skrevet i Java. Ved å velge en applet slipper man således å skrive kode for samme logikk to ganger. Dette er for meg det mest tungtveiende argumentet for en løsning basert på en applet.

Valget av teknologi falt derfor på en Javaapplet for graftegneren i klienten. Dette gir de nødvendige mulighetene for å lage interaktivitet samtidig som det støttes på alle relevante plattformer[25]. I praksis blir serversidedelen et subsett av funksjonaliteten i appleten. Ulempen med valget av en applet i forhold til å gjøre alt på serversiden (som var min aller første skisse

[24] I betydning programvare for utvikling av programvare, altså en IDE (Integrated Development Environment) på engelsk.
[25] Brukeren må riktignok ha installert en såkalt plugin for å vise Javaappleter, en Java Virituell Maskin (JVM). Men dette er i alle fall tilgjengelig gratis på alle utbredte operativsystemplattformer og til alle de populære nettleserne.

til modell) er at nettleseren til brukeren må støtte det, samt at man blir avhengig av prosessorkraft på klientmaskinen. Dette siste kan dog som nevnt også være en fordel i form av at det avlaster serveren.

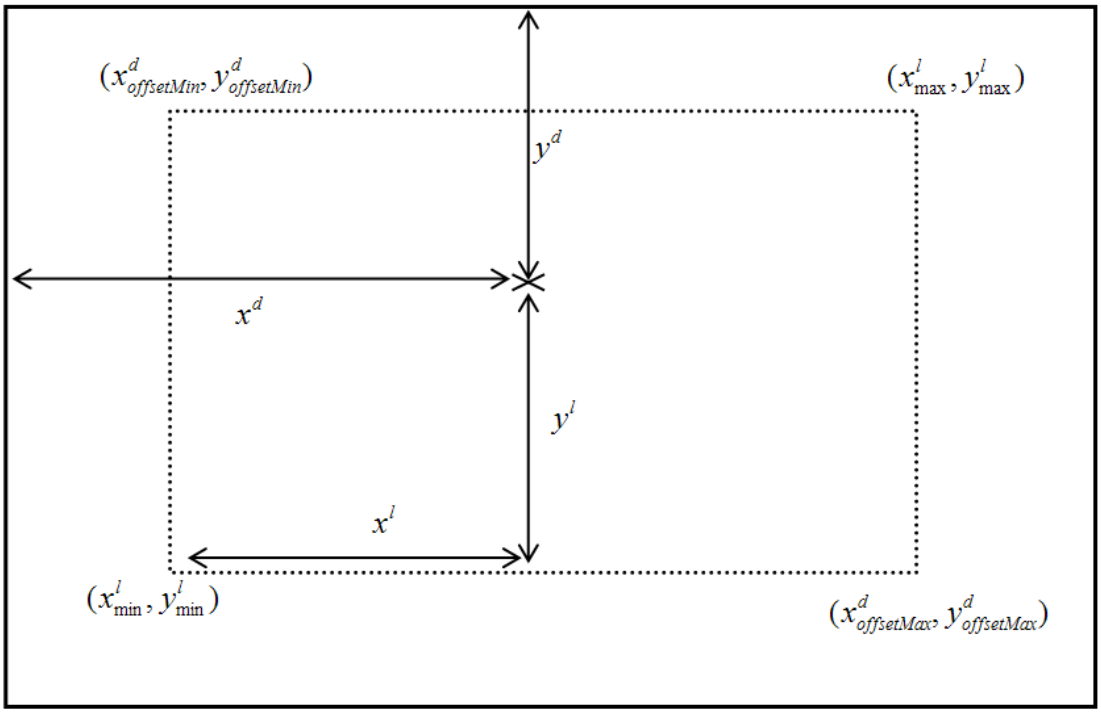
Prototypen slik den foreligger bruker MathML som grunnlag for å tegne opp grafene i graftegneren. Dette gjøres ved å traversere MathML-uttrykket for å lage en ren-tekst representasjon av uttrykket. Jeg har så benyttet tre klasser skrevet av Leigh Brookshaw (se [BROOKSHAW2003]) for å oversette fra et uttrykk i ren-tekst til en verdi. Klassene er utgitt som fri programvare med åpen kildekode, mer detaljer finnes i dokumentasjonen av prototypen, bl.a. på CD-ROMen som er omtalt i appendiks B.

Problemet blir dermed redusert til å klare å få ren tekst versjon av matematikk (i formatet som biblioteket takler) ut fra MathML-uttrykkene som er lagret i databasen. Det blir naturligvis bare et subsett av uttrykk (på formen y=<uttrykk med x>, samt x=<uttrykk med t>, y=<uttrykk med t>) som det blir mulig å tegne grafer for. Dette gjøres nå via ContentMath-klassen sin getPlainTextMath()-metode, se også senere avsnitt for ytterligere detaljer og appendiks B for full dokumentasjon (og kildekode på CD-ROM). Traverseringen bruker content-delen av MathML. Dermed brukes samme lagrete objekt til både symbolsk visning og generering av grafer.

At ren tekst benyttes til kommunikasjonen i forbindelse med beregning av verdiene for graftegning kan kanskje tolkes som at man da igjen innfører alle problemer knyttet til unik tolkning av matematiske uttrykk i ren tekst. Slik er det imidlertid ikke. Ved generering av en representasjon i ren tekst har man muligheten til å utnytte den eksplisitte semantikken som ligger i content-delen av MathML og legge på parenteser m.v. som gjør at uttrykket tolkes riktig. Dessuten lagres ikke ren-tekstversjonen noe sted, den brukes kun til akkurat dette formålet og genereres før punktverdier for et matematisk uttrykk skal beregnes i graftegneren.

Dersom man tillot seg å kreve en XML-parser på klientsiden ville samme funksjonalitet som den Brookshaw sine klasser utfører kunne gjøres enklere i klienten og man ville da kunne kvitte seg med ren-tekst formatet. Grunnen til at det ville bli enklere er at content-MathML har en struktur som eksplisitt viser den matematiske meningen og som relativt direkte ville gi et beregnbart uttrykk. Koden ville bli endret slik at getPlainTextMathFromSubTree() i stedet ville være getValueFromSubTree(), resten av koden ville bli ganske lik.

Punktenes verdier beregnes i virkelige verdier og må deretter oversettes til skjermkoordinater. Koordinatene som benyttes i Java starter med (0,0) i øverste venstre hjørne. Man kunne fått noe hjelp til oversettingen av innebygde funksjoner (som translate()), men på grunn av skaleringsfaktorer og lignende måtte man uansett ha gjort noe av beregningen manuelt. I figur 15 på neste side er omregningen fra virkelige verdier til et punkt i koordinatsystemet vist. Den tykke linjen ytterst er hele området for graftegneren, mens den prikkete linjen indikerer området selve grafen skal tegnes innenfor.



Figur 15 – Utregning av skjermkoordinater fra logiske koordinater

Utregning av skaleringsfaktor i fysiske koordinater blir lik i begge retninger, så jeg viser bare for x-retningen. Skaleringsfaktoren, her kalt *xSF* blir gitt ved:

$$xSF = \frac{x^d_{offsetMax} - x^d_{offsetMin}}{x^l_{\max} - x^l_{\min}}$$

der koordinater merket med superscript *l* er logiske koordinater (virkelige verdier), mens koordinater med superscript *d* er skjermkoordinater (devicekoordinater). Identisk utregning for y-retningen gir oss *ySF*. Dette gir x-verdien for skjermkoordinaten som skal brukes ved tegning:

$$x^d = \lfloor xSF(x^l - x^l_{\min}) + 0.5 \rfloor + x^d_{offsetMin}$$

For y-retningen blir verdien for koordinaten for tegning, y^d, gitt ved:

$$y^d = y^d_{offsetMax} - \lfloor ySF(y^l - y^l_{\min}) + 0.5 \rfloor$$

Siden beregningen av verdier krever noe ressurser er det en fordel om beregning av punktene frikobles fra tegningen slik at en gjenopptegning (for eksempel om et vindu har overlappet grafen) ikke nødvendigvis medfører en ny beregning av punktene. Dette vil kreve noe minne for lagring av resultatene mellom opptegningene, men sparer beregningsressurser. Hvordan dette er løst konkret i prototypen er dokumentert senere i rapporten, i avsnitt 5.6.1 og i mer detalj i appendiks B (og som kommentarer i kildekoden).

En ting som også kan være verdt å kommentere (som også har endret seg under arbeidet med prototypen) er hvordan intervallene for opptegning av de enkelte grafene gjøres. Det som har vært klart hele tiden var at:

- Det må kunne endres skala og visningsintervall (zoomes)
- Det må kunne være flere grafer (med ulike definisjonsintervall) i samme opptegning
- Det må kunne tegnes grafer der x-akse og definisjonsintervall for den frie variabelen ikke er den samme (altså at både x og y er parameterfremstillinger av t, for eksempel)

For å få til dette er intervallene for hvor funksjonene er definert frikoblet fra intervallene for hva som tegnes opp. Definisjonsintervallene kan godt være uten grenser (se dokumentasjonen av klassen MathInterval i appendiks B for detaljer for hvordan dette er løst i min prototype). For å finne intervallet $I_{draw,i}$ hvor funksjon i skal tegnes blir det altså:

$$I_{draw,i} = D_i \cap I_{draw}$$

I uttrykket over er D_i definisjonsmengden til funksjon i og I_{draw} det maksimale intervallet alle funksjonsuttrykkene skal tegnes for. Det er altså snittet[26] av de to intervallene som brukes. Så om tegneintervallet er [-5,5] og funksjonen er definert for [2,10] blir intervallet som beregnes og tegnes for den aktuelle funksjonen [2,5]. Antallet punktverdier i hvert intervall fordeles etter hvor stor andel av det totale tegneintervallet det utgjør.

5.6. Klassemodell og databasemodell

5.6.1. Klassemodell (Java)

Prototypen først og fremst var ment for å teste bruk av MathML som lagringsformat for matematikk i et system lignende det Inspera benytter. Fokus har således ikke vært å komme så langt som mulig med implementasjonen eller at resultatet skal være direkte nyttig. Jeg har likevel laget en modell som er mer generell en det jeg strengt tatt trenger for prototypen slik at den enkelt kan utvides for andre lignende bruksområder. Jeg vil gå kort gjennom litt av tankegangen bak klassemodellen før jeg presenterer modellen i form av et klassediagram i UML-notasjon.

Et viktig poeng var å skille logikken for utregning av punktverdier fra selve tegnelogikken. Dette er gjort ved at et grensesnitt PointValueProducer ble laget. I prototypen er det kun fra matematiske uttrykk det skal regnes ut verdier, men i fremtiden kan man godt tenke seg implementasjoner der verdiene i stedet hentes fra resultatlisten til avdelingens sekkeløp i en fil eller fra en temperaturmåler som er koblet til serieporten på maskinen. Alt som må gjøres for å støtte tegning fra slike verdikilder er å lage en ny implementasjon av en PointValueProducer og sørge for at en instans av denne returneres fra PointValueProducerFactory når innstillingene (GraphDrawerSettings) tilsier dette i fremtiden. Her benytter jeg et såkalt Factory-pattern (se for eksempel kapittel 3 i [GAMMA1994]) som i denne sammenhengen gjør at kalleren ikke trenger å vite hvilken subtype den bruker av en klasse eller forholde seg til detaljene rundt konstruksjonen av riktig objekt.

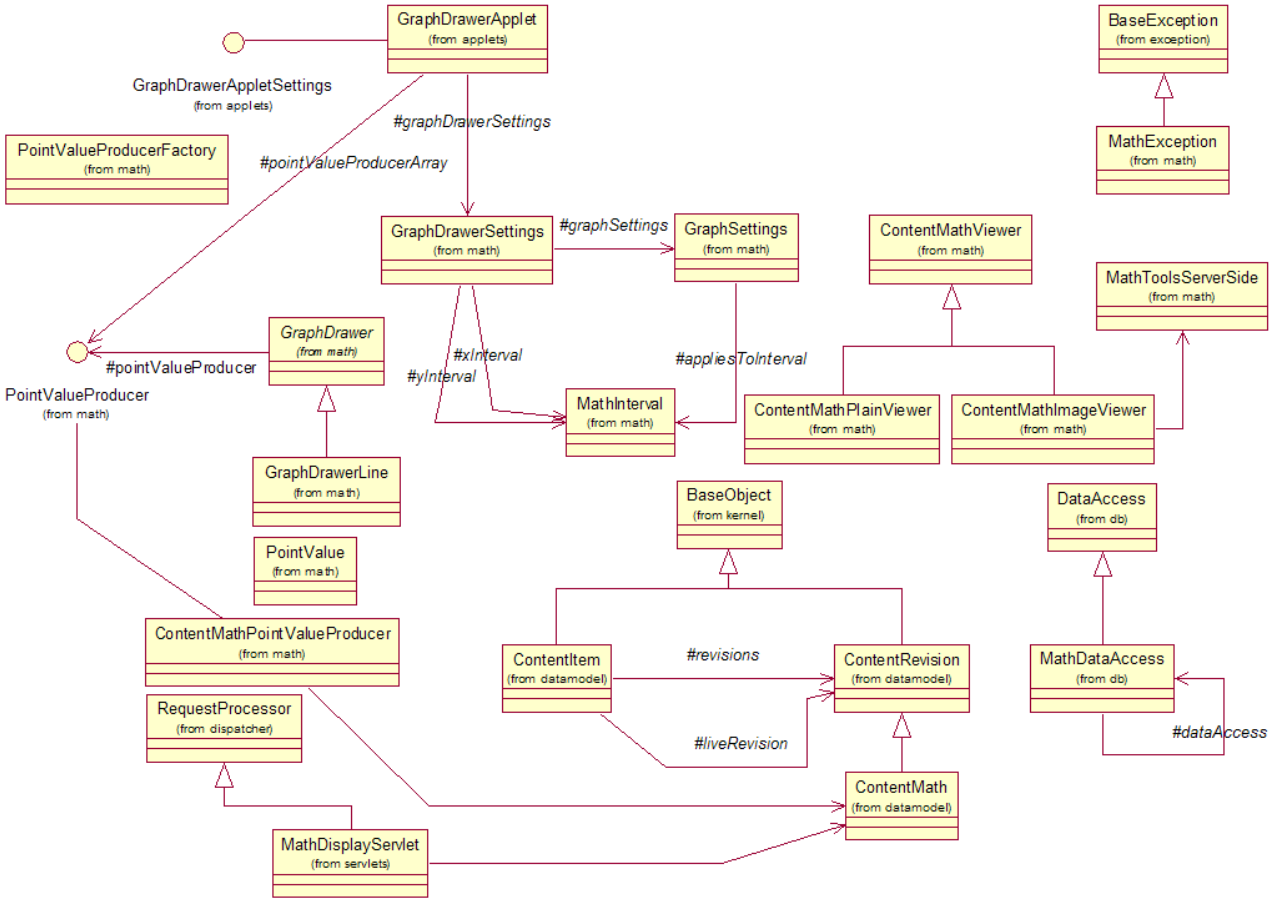
Et annet poeng som er nevnt tidligere, under diskusjonen av teknologi for graftegneren i kapittel 5.5, var at vi ikke ønsket duplisering av kode som ble brukt i graftegneren som kjører

[26] Som det går frem av koden og dokumentasjonen av denne er dog intersect() ikke strengt tatt det matematiske snittet i alle tilfeller. Den er definert slik at den returnerer snittet bare dersom begge intervallene har en eller flere grenseverdier – dersom et intervall er den tomme mengden vil det ikke resultere i at det returnerte "snittet" blir tomt automatisk, men at det andre intervallets verdier blir bestemmende. Dersom intervallklassen jeg har laget skal brukes i andre sammenhenger kan det være at dette må justeres og at disse testene i stedet må gjøres i den kallende delen av koden.

i nettleseren og det som kjører på serversiden for å kunne produsere bildefiler for inkludering i trykkformat (PDF). Dette er løst ved at ingen tegning gjøres av appleten i seg selv, men er skilt ut i en egen klasse, GraphDrawer. GraphDrawer er i sin tur en abstrakt klasse, altså en klasse det aldri skal opprettes objekter (instanser) av. Det er spesialiseringene i form av ulike graftegnere som gjør selve tegningen, i min prototype trenger jeg egentlig bare en – nemlig GraphDrawerPlot. Modellen gjør at det ikke blir en eneste linje å endre i appleten dersom innstillingene sier at det for eksempel skulle være en fremtidig GraphDrawerBarChart som returneres av GraphDrawer sin getInstance-metode (igjen et eksempel på Factory-pattern).

En detaljforklaring på hvordan kallene foregår ved tegning i appleten følger i "Figur 17 – Sekvensdiagram for kallene ved tegning av en graf i graftegneren", etter klassediagrammet er presentert og en kort tekstlig forklaring på oppgaven til de ulike klassene er gitt.

Under følger klassediagrammet for prototypen på databaselagring og visning av MathML, samt graftegner.



Figur 16 – Klassemodell for prototypen i Java

Et siste bevisst definert grensesnitt (i tillegg til de nevnt over) er det jeg tidligere snakket om rundt å legge til rette for bytte av bibliotek og implementasjon rundt produksjon av grafikk/sluttformat for bruker fra MathML-uttrykk. Dette er realisert gjennom ContentMathViewer[27]-grensesnittet og implementasjoner av dette.

[27] Rent Java-teknisk er det ikke et interface, men en abstrakt klasse (altså en klasse det aldri kan opprettes instanser av, bare av underklasser som implementerer alle de påkrevde metodene). Forskjellen er at en abstrakt klasse også kan ha

De fleste klassene ligger under pakken no.inspera.applications.math. Under følger en kort forklaring på hva de ulike klassene har av oppgaver og ansvarsområder. Mer utfyllende kommentarer finnes i kodedokumentasjonen, blant annet i form av en begrenset versjon av JavaDoc for prototypen i appendiks B der de enkelte klasser og metoder er dokumentert.

no.inspera.kernel.BaseObject
Denne klassen tilsvarer OBJECT-tabellen i databasen på Javasiden. Det er et generelt objekt som kan ha en del basisegenskaper (knyttet til for eksempel rettigheter og sporing av bruk) og mange av objektene i datamodellen arver fra dette. Denne klassen eksisterte allerede i Inspera Platform før prototypen ble utviklet.

Det er knyttet til denne objekttypen at mange av punktene jeg nevnte i kapittel 5.1.1 blir tatt hånd om i rammeverket jeg har benyttet for prototypen.

no.inspera.services.exception.BaseException
BaseException er en generell klasse for unntakshåndtering som alle definerte Exception-klasser i Inspera Platform skal være subklasser av. Dette gir sentralisert håndtering av logging og brukeren kan få feilmeldinger på sitt valgte språk. Denne klassen eksisterte allerede i Inspera Platform før prototypen ble utviklet.

no.inspera.kernel.dispatcher.RequestProcessor
RequestProcessor er en servlet med fellesfunksjonalitet som alle servlets i Inspera Platform er subklasser av. Servleten kaller subklassene sine versjoner av metodene for håndtering av HTTP GET/POST etter å ha utført tilgangssjekk, logging og eventuelt andre handlinger. Denne klassen eksisterte allerede i Inspera Platform før prototypen ble utviklet.

no.inspera.applications.publish.ContentItem
ContentItem representerer et logisk innholdselement eller informasjonsobjekt i databasen. Dette er enheten som blir forespurt av brukeren, som så får versjonen som passer best med eget språk og i henhold til hvilken revisjon som er den publiserte. ContentItem inneholder altså ikke selve innholdet, noe som muliggjør tilbakerulling til tidligere versjoner ved redigering og sporing av bruk av det logiske objektet selv etter en ny publisering av en revisjon og på tvers av ulike språkversjoner av innholdet.

ContentItem er en spesialisering av BaseObject og arver således dennes egenskaper. Denne klassen eksisterte allerede i Inspera Platform før prototypen ble utviklet.

no.inspera.applications.publish.ContentRevision
Dette er en klasse som representerer en revisjon av et informasjonsobjekt på et bestemt språk. Det er her selve det tekstlige innholdet ligger (eller binære innholdet for typer der det er aktuelt, for eksempel bilder, video og lyd).

ContentRevision er også en spesialisering av BaseObject og har derfor alle egenskapene fra denne klassen innebygd. Denne klassen eksisterte allerede i Inspera Platform før prototypen ble utviklet.

---

metodedefinisjoner som arves av subklassene, noe som er tilfellet her (bl.a. er factory-metoden lagt direkte til denne klassen i stedet for å ha en separat klasse for dette). En annen forskjell på interface og abstrakt klasse er for øvrig at Java ikke har multippel arv, men at det er tillatt for en klasse å implementere så mange interfaces som man måtte ønske.

no.inspera.applications.publish.math.ContentMath
ContentMath-klassen inneholder et matematisk objekt. Som forklart i avsnittet om valg av lagringsformat lagres disse i databasen i form av XML som lagres samlet i databasen uten videre nedbryting. Klassen er en spesialisering av ContentRevision og arver således alle egenskaper fra denne.

I praksis er det kun MathML som brukes som markeringsspråk i klassen, men strengt tatt vil klassens hovedfunksjoner fungere også med et annet XML-format. Dette kan være en fordel dersom man på et senere tidspunkt skulle komme til å revurdere valget av lagringsformat. Noen endringer vil imidlertid måtte gjøres for tolkning av uttrykket i forbindelse med graftegneren. Tredjepartskomponentene som brukes til visning av symbolske uttrykk ville også måtte byttes, dersom ikke det eventuelle nye formatet benyttet eller kunne oversettes til presentasjonsdelen av MathML.

no.inspera.applications.publish.math.PointValue
PointValue representerer en punktverdi. I forbindelse med prototypen vil alle punktverdier ha to dimensjoner, selv om det er lagt til rette for støtte av flere dimensjoner på en enkel måte.

no.inspera.applications.publish.math.PointValueProducer
Som forklart tidligere er dette et definert grensesnitt som muliggjør ryddig skille mellom tegningslogikk og produksjon av verdier for graftegning. Dette er forklart i større dybde tidligere (se innledningen i dette kapitlet), samt også kommentert relativt fyldig i de ulike deler av JavaDoc i appendiks B.

no.inspera.applications.publish.math.ContentMathPointValueProducer
Implementasjon av produksjon av verdier for graftegning basert på matematiske uttrykk. Også dette er forklart i større dybde tidligere.

no.inspera.applications.publish.math.PointValueProducerFactory
Dette er en klasse som brukes for å få tak korrekt type PointValueProducer før produksjon av verdier for graftegning begynner. For mer detaljer se forklaring over, sekvensdiagram på side 41 og JavaDoc i appendiks B.

no.inspera.applications.publish.math.GraphDrawer
Med tanke på fremtidig utvidelse til flere typer grafer ble grensesnittet mot tegnelogikken skilt ut i denne abstrakte klassen. Se også fyldigere dokumentasjon andre steder i rapporten.

no.inspera.applications.publish.math.GraphDrawerPlot
Implementasjon av tegning av grafer i form av innplottede punkter. Denne implementerer da grensesnittet definert av GraphDrawer-klassen. I tillegg til fordelen med enkel utvidelse til nye typer grafer muliggjør utskillingen som egen klasse også at klassen kan benyttes direkte av både appleten og serversidendelen av løsningen.

no.inspera.applications.publish.math.GraphDrawerSettings
Denne klassen holder innstillinger for graftegneren og brukes av både appleten og tjenerdelen av løsningen. Innstillingene inkluderer om akser skal tegnes eller ikke, skala, osv. Hver (del-)graf har sitt GraphSettings-objekt, referanser til disse er lagret her.

no.inspera.applications.publish.math.GraphSettings
Denne klassen holder alle relevante innstillinger for en graf og brukes av både appletdelen og tjenerdelen av løsningen. Innstillingene inkluderer farge på grafen og intervallet den gjelder. Hver graf har sitt GraphSettings-objekt, dermed vil det ved delt funksjonsforskrift eksistere like mange slike objekter som det er delintervaller.

no.inspera.applications.publish.math.MathToolsServerside
Klassen inneholder funksjonalitet for å opprette GraphDrawerSettings-objekt fra parameterverdiene i en HTTP-forespørsel. Dette var opprinnelig funksjonalitet som lå i GraphDrawerSettings-klassen, men siden denne klassen skulle brukes av både klientdel og tjenerdel ble det et problem med enkelte JVMer at klassen måtte være tilgjengelig i klienten (selv om metoden aldri kalles i klientdelen).

no.inspera.applications.publish.math.ContentMathViewer
Dette er et definert grensesnitt for visning av ContentMath-objekter til ulike formater. (Per i dag er dette altså i praksis et grensesnitt for å vise MathML til ulike formater, men dette vil endre seg dersom bruken av ContentMath endres – som kommenter over.)

no.inspera.applications.publish.math.ContentMathPlainViewer
Dette er en implementasjon av ContentMathViewer som gir MathML-innhold[28] direkte. Den kan altså brukes mot klienten eller programmer som har innebygd MathML-støtte.

no.inspera.applications.publish.math.ContentMathImageViewer
Dette er en implementasjon av ContentMathViewer som gir en symbolsk visningsversjon av ContentMath-objekter i form av et bilde. Den kan altså brukes mot klienter eller programmer som ikke har noen spesiell innebygd MathML-støtte. Den brukes også for produksjon av bilder som skal inkluderes i PDF-produksjon.

no.inspera.applications.publish.math.applets.GraphDrawerApplet
Dette er appleten som utgjør selve graftegneren. Mest mulig funksjonalitet er flyttet ut av appleten med tanke på at samme kode skal brukes på tjener og klient, men ting som dynamisk visning av maksimalpunkt og zoom er til dels implementert her.

Kommunikasjon med omliggende HTML-side skjer via én sentralisert metode som kalles fra javascript og navngitte meldinger/parameterverdier som sendes via denne. Dette gjør at det er enklere å endre brukergrensesnittet, siden man ikke trenger endre selve javakoden (men bare websiden den er brukt på). Dette utnyttes bl.a. ved integrasjonen av grafer i HTML-dokumenter der forfatteren kan skru mulighetene for zooming m.m. av og på. Litt mer om dette kommer i delen der selve prototypen (og bruken av denne) blir presentert.

no.inspera.applications.publish.math.applets.GraphDrawerAppletSettings
Her ligger en del konstanter som brukes av appleten samlet, for å gjøre redigering av disse enkelt.

no.inspera.applications.publish.math.MathInterval
Et intervall representeres av denne klassen. Brukes av GraphDrawerSettings for å si hvilket område grafen skal tegnes for, samt av GraphSettings for å si hviklet definisjonsområde en funksjon har. Kan være lukket eller åpent i begge ender.

[28] Igjen gjelder selvsagt at dersom lagringsformatet var et annet vil denne klassen gi dette formatet direkte uten noen prosessering, men for dagens prototype er det altså kun MathML vi snakker om.

no.inspera.db.DataAccess
Gir nødvendige metoder for å få tak i databasetilkobling og andre rene administrasjonsoppgaver med tanke på databaseinteraksjon. Inneholder ikke spørringer for applikasjonsspesifikke (som matematikk i dette tilfellet) behov. Denne klassen eksisterte allerede i Inspera Platform før prototypen ble utviklet.

no.inspera.db.MathDataAccess
Inneholder all databaseinteraksjon (spørring og kall til prosedyrer) som applikasjonen benytter. Bruker Singleton-pattern[29] med tilhørende factory-metode for å få tak i en instans for bruk.

no.inspera.applications.publish.servlets.MathDisplayServlet
Denne servleten støtter visning av matematiske uttrykk i form av ren MathML, et bilde av det symbolske uttrykket eller et bilde av en graf. Logikken i servleten er helt minimal, den bruker klassene over for å utføre dette. Servleten bruker kun klassene over og jobber ikke direkte mot databasen på noen måte.

GraphDrawerController (JSP)
En java-servlet laget i form av en JSP som ved hjelp av klassene over tegner grafer. Den er ikke hovedsakelig ment som ferdigutvilet funksjonalitetsmessig (eller brukergrensesnittsmessig) for sluttbruker, men som eksempel på hva prototypen kan gjøre. Den finnes ikke i klassediagrammet i den versjonen som ble med i den trykte versjonen av rapporten, for å gjøre diagrammet så oversiktlig som mulig.

JSPen skal bruke klassene under for tilgang til database, den forholder seg kun til dataaksess-grensesnittet og trenger ikke vite noe om underliggende lagring (som i dette tilfellet er relasjonsdatabase som benyttes via JDBC).

no.inspera.applications.publish.math.MathException
Generell klasse for unntakshåndtering innen matematikkdelen av applikasjonen, subklasse av BaseException (nevnt over) og gir dermed alle muligheter denne har.

I tillegg til klassene som er beskrevet over (som er utviklet fra grunnen av der ikke noe annet er nevnt spesielt) har følgende klasser fått et tillegg for å støtte matematiske uttrykk for innlegging og visning som del av dokumenter:

- no.inspera.applications.publish.servlets.ContentDisplayServlet (lagt til støtte for visning av matematiske uttrykk, til ulike sluttformater – i praksis å sette i gang en ContentMathViewer). ContentDisplayServlet støtter visning av kun ett innholdselement, siden vi også ønsket å støtte visning av grafer med mange objekter ble servleten MathDisplayServlet (beskrevet over) laget i tillegg. Denne støtter å bruke flere innholdselementer i samme visning.
- no.inspera.applications.publish.servlets.CMSAdminServlet og tilhørende CMSService (lagt til mulighet for å legge inn og redigere matematiske uttrykk, bruker felles funksjonalitet for revisjonskontroll og språk)
- no.inspera.utils.XSLTools (lagt til støtte for at en XSL-mal kan kalle opp et matematisk uttrykk for å lage dynamisk grensesnitt rundt en applet-basert graftegner inne i visningen av et dokument)

[29] Det vil altså eksistere kun ett objekt av denne klassen, for detaljer om dette designpatternet se for eksempel http://www.castle-cadenza.demon.co.uk/single.htm.

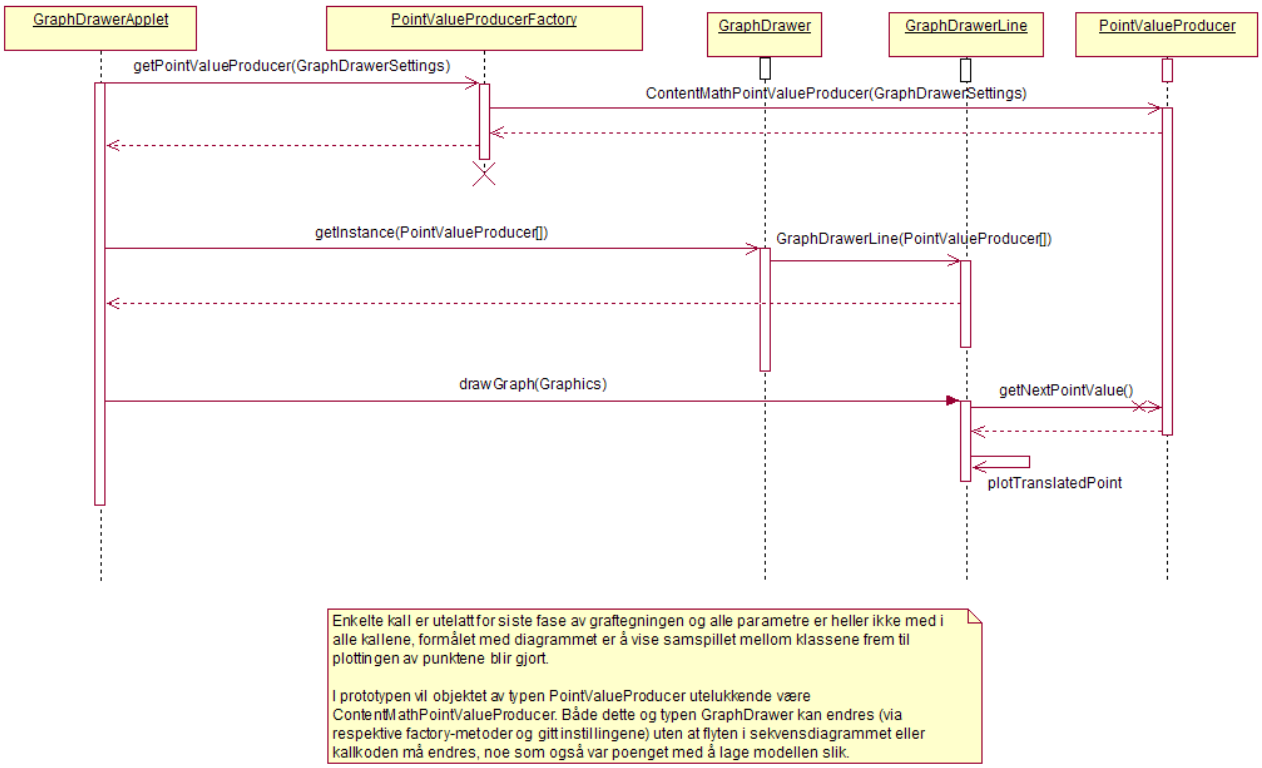
For disse klassene har jeg for ryddighets skyld bare tatt med kildekoden for de delene jeg nå har nyutviklet på vedlagte CD-ROM, se appendix B for detaljer om katalogstruktur på denne.

Senere, i kapittel 7.3 av denne rapporten, vil jeg vise noen enkle linjediagrammer i forbindelse med ytelsesvurderingen av prototypen. I den sammenheng ville jeg finne ut om modellen over virkelig var så enkel å tilpasse til litt andre bruksområder som jeg mente i teorien. Derfor har jeg i tillegg til selve løsningen for prototypen laget tre klasser som gjorde at jeg kunne benytte prototypen for å lage disse diagrammene:

- GraphDrawerLine – en implementasjon av GraphDrawer for linjebaserte grafer, der punktverdiene sammenkobles med linjer fra et punkt til det neste.
- PlaintextFilePointValueProducer – som navnet sier en PointValueProducer der verdiene finnes ved lesing av en tekstfil i stedet for ved beregning av matematiske uttrykk, slik tilfellet var for ContentMathPointValueProducer (som forklart over).
- GraphCmdLine – et enkelt kommandolinjeverktøy for å oppgi innfil (ren tekst) og utfil (bildefil) for å lage en graf basert på statistiske data.

Disse tre klassene er ikke med på klassediagrammet over, da de ikke er en del av prototypen som sådan. Men også for disse klassene er kildekoden på vedlagt CD-ROM og grensesnittet dokumentert på CD-ROMen omtalt i appendiks B. Det tok ikke lengre tid enn ventet å lage disse klassene (heller tvert i mot!). Dette må derfor kunne sees på som en indikasjon om at modellen kan anvendes generelt, slik det var argumentert for teoretisk i forbindelse med utformingen av ansvarsfordelingen mellom klassene i prototypen, i begynnelsen av dette kapitlet.

På neste side følger et sekvensdiagram som viser i noe mer detalj hvordan interaksjonen foregår ved tegning av en graf i applet-delen av prototypen.



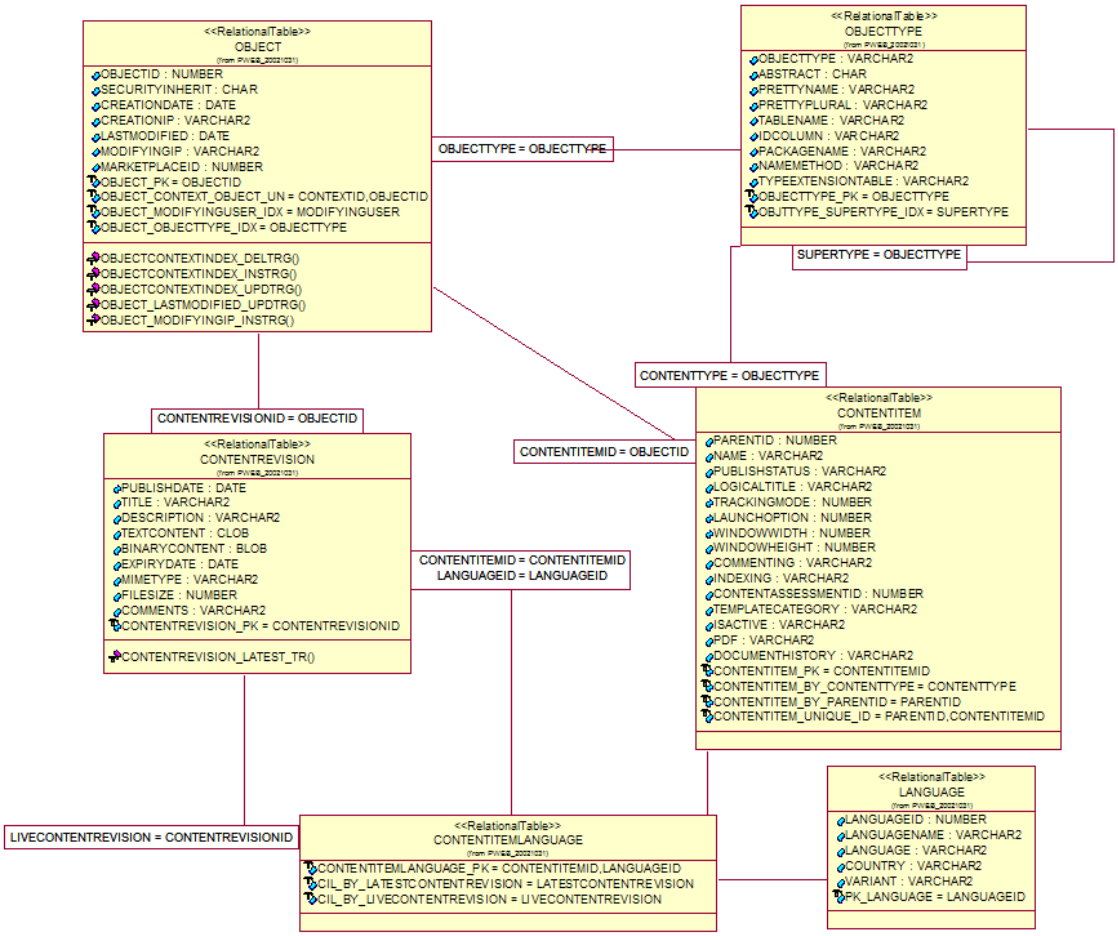
Figur 17 – Sekvensdiagram for kallene ved tegning av en graf i graftegneren

5.6.2. Databasemodell - Oracle

Databasemodelleringen slik jeg valgte å utforme løsningen (med lagring av MathML-uttrykket direkte i databasen) er relativt triviell. Selve MathML-uttrykket lagres som en samlet datamengde uten nedbrytning i databasen altså som en kolonne av typen CLOB (eller BLOB). Dersom man benytter Oracle9i har denne også datatypen XMLType som med fordel kan brukes. Sistnevnte datatype gir en del muligheter for indeksering og spørring på dokumentfragmenter (via XPath), selv om lagringen av dokumentet skjer samlet.

Noe som imidlertid er verdt en kommentar er at modellen støtter versjonering med tilbakerulling til tidligere versjoner og flere språkversjoner av samme logiske innholdselement. Det er også mulig å legge til nye versjoner på kun ett språk slik modellen er, mens det for andre språk benyttes en eldre versjon.

Tabellene som benyttes i databasen er vist på neste side.



Figur 18 – Databasemodell for lagring av MathML i en Oracle-database

Et logisk innholdselement (omtrent tilsvarende et informasjonsobjekt om man vil følge terminologi fra digitale biblioteker) modelleres i databasen som et CONTENTITEM. Her finnes en forekomst/rad for ethvert innholdselement, uavhengig av typen innhold (for eksempel lyd, video, tekstlig dokument og nå også matematisk objekt). Det faktiske innholdet lagres i tabellen CONTENTREVISION og tabellen CONTENTITEMLANGUAGE holder styr på hvilken revisjon som er den aktive for hvert språk. Dette er måten databasemodellen i løsningen forsøker å ta høyde for de generelle retningslinjene fra kapittel 5.1.1.

Det er altså ingen egen tabell for lagring av MathML, jeg bruker kolonnen CONTENTREVISION.Textcontent for denne lagringen. Det legges imidlertid også inn metadata for objekttypen i tabellen OBJECTTYPE. Dette gjør at dynamisk sletting av rader med MathML vil fungere (SQL genereres da på grunnlag av foreldreinformasjonen i typetabellen).

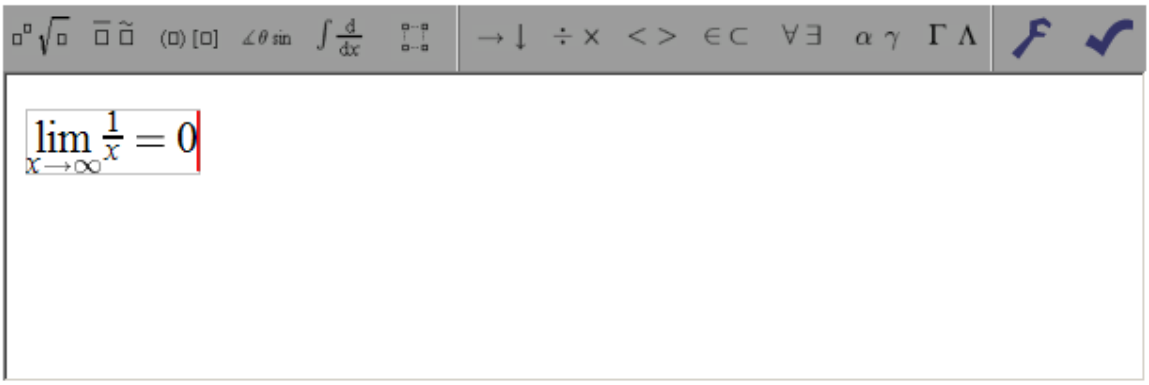
Det kan også være verdt å merke seg at både CONTENTREVISION og CONTENTITEM er OBJECTer. Modelleringen her benytter arv slik at det ved oppretting av en revisjon også alltid vil eksistere en tilhørende rad i OBJECT, siden en revisjon også er et objekt i databasemodellen til Inspera. Dette gjør at en del felles egenskaper (rettigheter, sporing av bruk, m.v.) kan benyttes. Opprettelsen av radene i tabellene i dette hierarkiet blir tatt hånd om av PL/SQL-pakkene som kjører på databasesiden (bl.a. via tidligere nevnte data i OBJECTTYPE). Dette gjør også at koden i Java slipper å bruke eksplisitt transaksjonshåndtering for å sørge for at opprettingen går riktig for seg. Et kall til

PKG_CONTENTREVISION.New vil altså gi en rad i OBJECT og en i CONTENTREVISION som har samme verdi for primærnøkkelen. Tilsvarende gjelder for CONTENTITEM ved oppretting av det logiske innholdselementet.

6. Presentasjon av utviklet prototype

6.1. Innlegging og visning av matematiske uttrykk

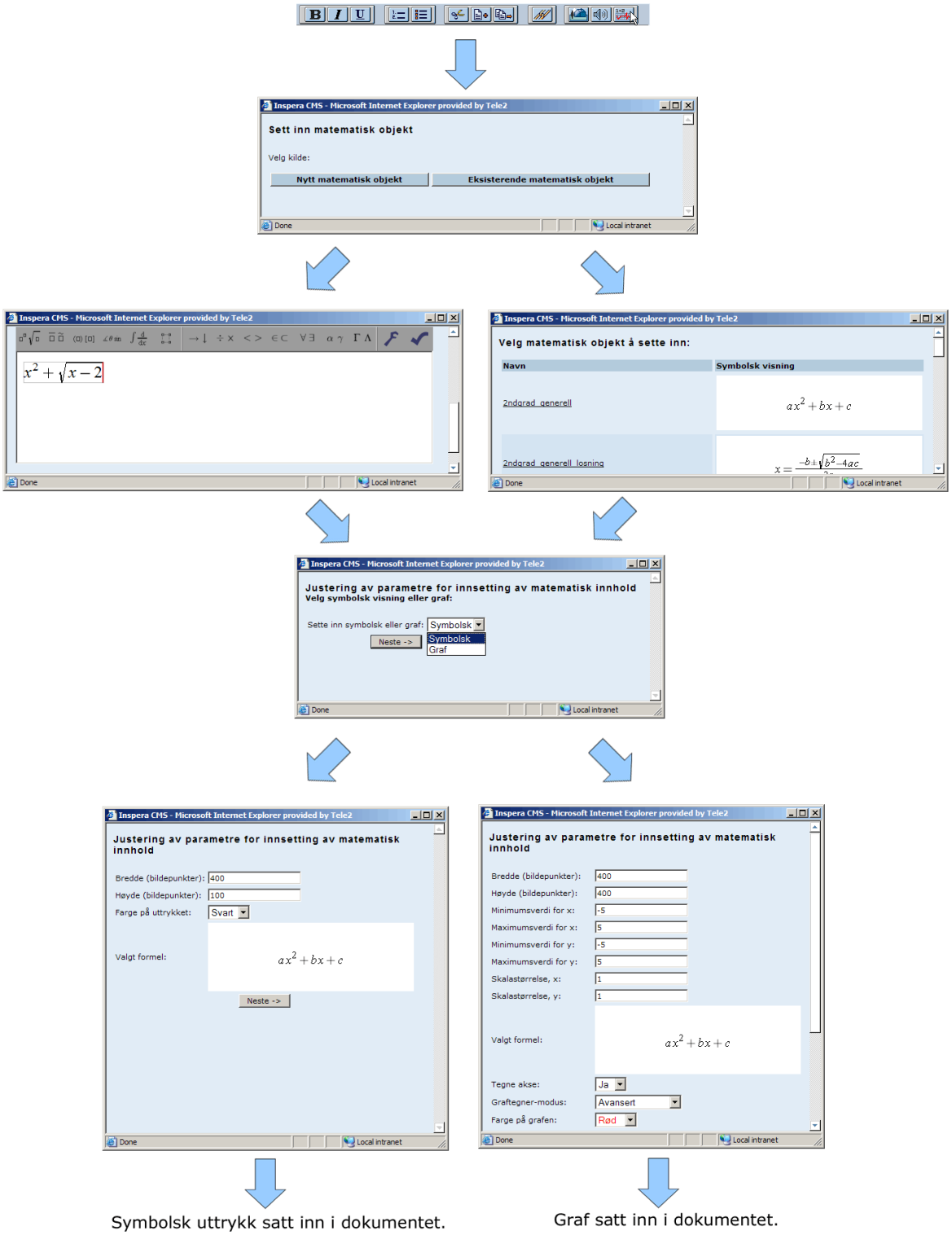
Som nevnt i kapittel 5.1 var produksjon av MathML før innleggingen i systemet definert som å være utenfor denne oppgaven. Jeg har likevel benyttet den nevnte formelredigeringskomponenten fra Design Science ved innlegging av MathML i systemet. Da kombineres content- og presentation-MathML ved å hente ut de to versjonene via et JavaScript-API og slå sammen før oversending til server. Koden for dette finnes på vedlagte CD-ROM, se appendiks B. Under følger et skjermbilde av editoren i bruk for å lage et uttrykk for lagring i systemet.



Figur 19 – Redigering av matematiske uttrykk med Design Science WebEQ

De matematiske objektene er i systemet selvstendige informasjonsobjekter med egen identitet, revisjonskontroll og mulighet for språkversjoner. I prinsippet redigeres altså disse helt uavhengig av anvendelsen i ulike dokumenter, selv om det ikke trenger å være slik brukergrensesnittmessig.

Brukeren kan velge å sette inn eksisterende matematiske uttrykk i et dokument eller opprette et nytt for innsetting. Flyten i veiviseren som lar brukeren sette inn et matematisk objekt som graf eller med symbolsk visning er vist i figuren på neste side.



Figur 20 – Veiviser for innsetting av matematikk i symbolsk form eller graf i et dokument

De matematiske objektene settes inn i de XML-baserte dokumentene ved en referanse. Hvilken type visning man ønsker styres per innsetting ved at XMLen for de to er forskjellig. Under følger et utsnitt fra utvidelsen av DTDen for å støtte innsetting av matematiske objekter i XML-baserte dokumenter. Det er verdt å legge merke til hvordan formen blir med referanse

til et separat[30] innholdsobjekt, som forklart og begrunnet tidligere. Ellers er noen parametre relatert til å kunne lage mer pedagogiske animasjoner i formatet, et lite eksempel på denne bruken følger senere.

```
<!ELEMENT symbolicMath (mathObject)>
<!ATTLIST symbolicMath
   width CDATA #REQUIRED
   height CDATA #REQUIRED
>

<!ELEMENT graphMath (backgroundImage?,mathObject+,userParameters?)>
<!ATTLIST graphMath
   guiMode (advanced | animation | none) "advanced"
   drawAxis (true|false) "true"
   width CDATA #REQUIRED
   height CDATA #REQUIRED
   minY CDATA #IMPLIED
   maxY CDATA #IMPLIED
   minX CDATA #REQUIRED
   maxX CDATA #REQUIRED
   scaleIndicatorSizeX CDATA #IMPLIED
   scaleIndicatorSizeY CDATA #IMPLIED
   animationImageLogicalTitle CDATA #IMPLIED
>

<!ELEMENT mathObject EMPTY>
<!ATTLIST mathObject
   logicalTitle CDATA #IMPLIED
   color CDATA #IMPLIED
   freeVarStart CDATA #IMPLIED
   freeVarEnd CDATA #IMPLIED
>
<!-- freeVarStart and freeVarEnd are only used in the setting
where mathObjects is children of a graphMath-element

The color attribute should be given in RRGGBB format.
-->

<!ELEMENT userParameters (userParameter+) >

<!ELEMENT userParameter EMPTY>
<!ATTLIST userParameter
   name CDATA #REQUIRED
   displayName CDATA #REQUIRED
   defaultValue CDATA #REQUIRED
>

<!ELEMENT backgroundImage EMPTY>
<!ATTLIST backgroundImage
   logicalTitle CDATA #REQUIRED
   tiled (true|false) "false"
>
```

Figur 21 – Utvidelsen av DTDen for Insperas dokumenter for bruk av matematiske objekter

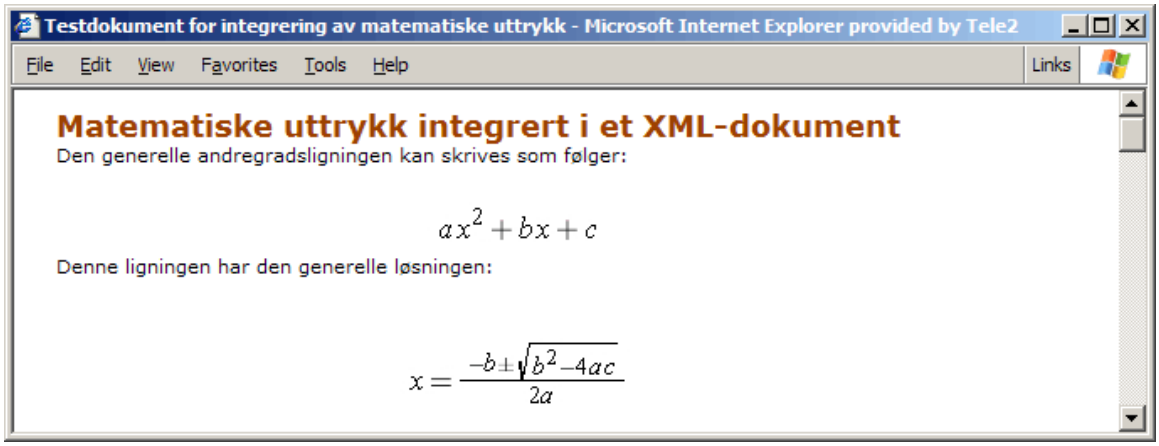
I tillegg til elementene dokumentert over gjorde jeg en liten endring på elementet `contentSection` som er grunnsteinen i Insperas dokumentformat, slik at denne kan

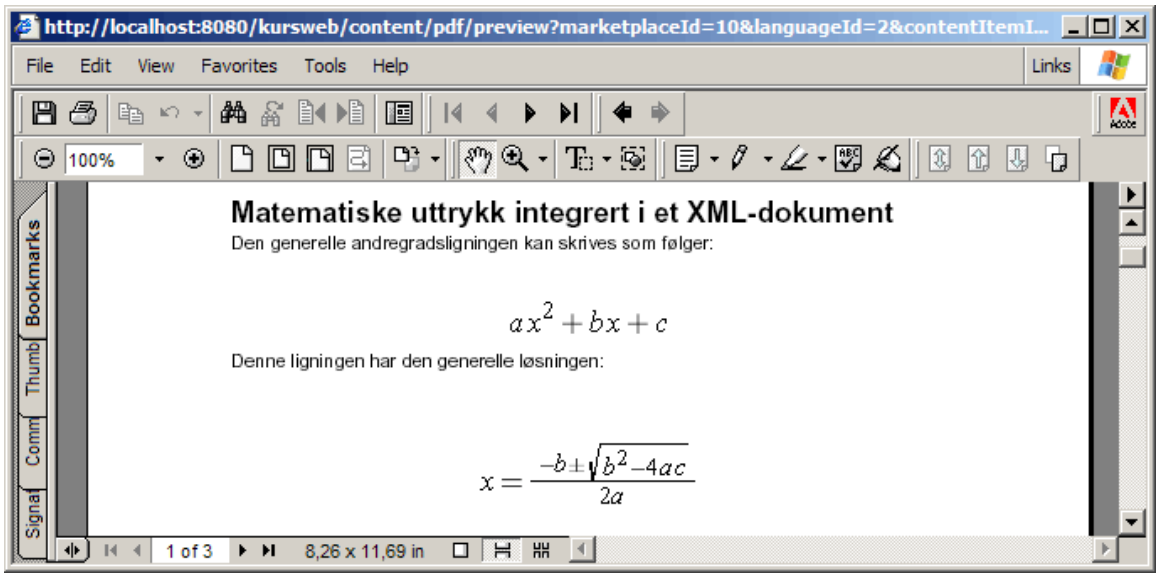
[30] Ved eksport til andre systemer er det sannsynlig at man vil gå fra denne representasjonen til enten å ha MathML-innholdet direkte i dokumentet eller å referere til et generert bilde. Dette er svært enkelt å implementere utfra løsningen som er brukt i prototypen, men som tidligere forklart ble valget at matematikkobjektene *i systemet* skulle være selvstendige innholdsobjekter som kunne anvendes i ulike sammenhenger og redigeres uavhengig av dokumentene det er brukt i.

inneholde `graphMath` og `symbolicMath` for å sette inn henholdsvis grafer og symbolsk visning av matematiske objekter.

Legge merke til at et `symbolicMath`-element skal ha ett og bare ett `mathObject`-element som barn, mens det for `graphMath` kan være ett eller flere. Man kunne vurdert å innføre et `graphPart`-element som hadde `mathObject` som barn i stedet for å legge attributtene for definisjonsmengde direkte på `mathObject`. I så fall ville man sluppet forholdet som er kommentert i utdraget fra DTDen, at enkelte parametre bare gjelder noen bruksområder. (Et annet alternativ ville være å ha et helt separat element, men jeg synes det er ryddig å bruke `mathObject` i begge tilfeller siden det er referanser til samme typen informasjonsobjekter).

Samme lagrede kilde kan bruke til produksjon av ulike sluttformater, i prototypen er det HTML og PDF som er benyttet. Ved produksjon av et dokument for utskrift i PDF-format gjøres hele prosessen på tjeneren før resultatet oversendes klienten som en ferdig fil med innebygde objekter. For HTML-formatet sin del lages det XHTML med et bildeelement for hvert matematiske objekt. Under er et skjermbilde fra et ferdiggenerert dokument fra et XML-basert innholdselement i Inspera sin løsning der to matematikkobjekter fra prototypen var satt inn, HTML er øverst og PDF nederst.





Figur 22 – Demonstrasjon av krysspublisering for et dokument med matematiske uttrykk

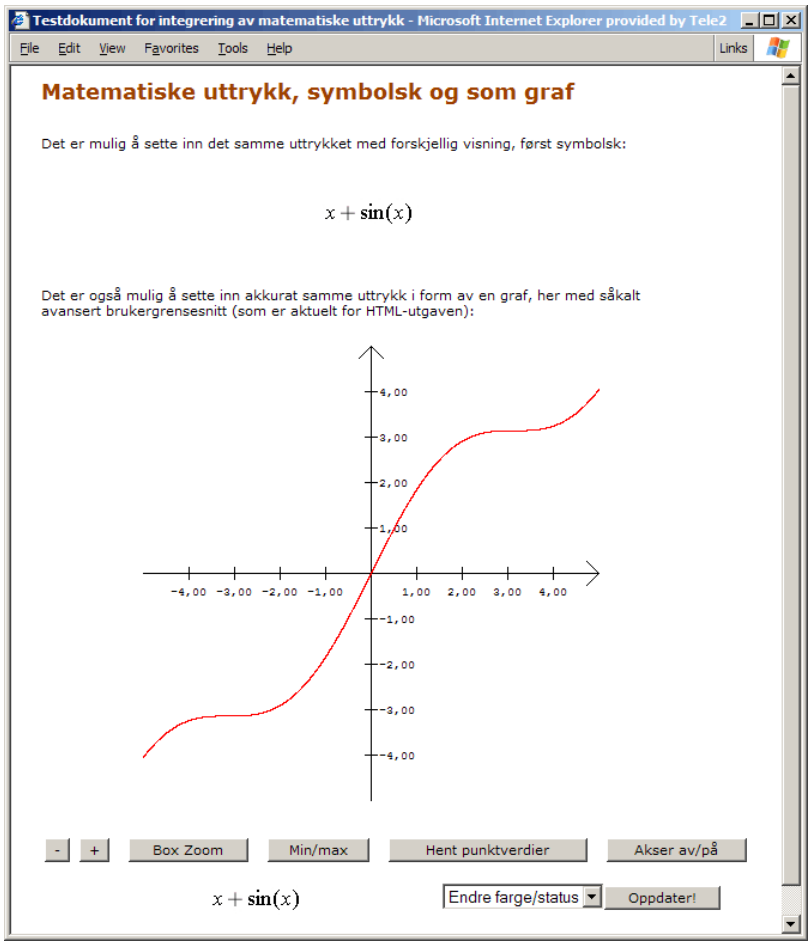
Siden det er samme XML-dokument som brukes som grunnlag for alle visningsformater blir det kun ett sted å oppdatere og alle sluttformater holdes automatisk oppdatert uten at forfatteren behøver å foreta seg noe.

6.2. Graftegner

Graftegneren baserer seg som nevnt i kapittel 5.5 på en traversering av content-delen av MathML-uttrykket etter lagring og plotting av punktverdier. Et og samme utrykk kan også brukes i to sammenhenger med forskjellig visning (graf et sted og symbolsk visning et annet sted).

Eksemplene som blir vist her i rapporten vil gi et brukbart inntrykk av hvordan graftegneren virker, men det finnes også noen eksporterte eksempler fra systemet på den vedlagte CD-ROMen. Dette er nærmere omtalt i appendiks C. Eksemplene på CD-ROMen er i hovedsak tilsvarende som de som blir vist her i teksten og også XML-utdraget som resulterte i visningen i eksporten finnes på CDen.

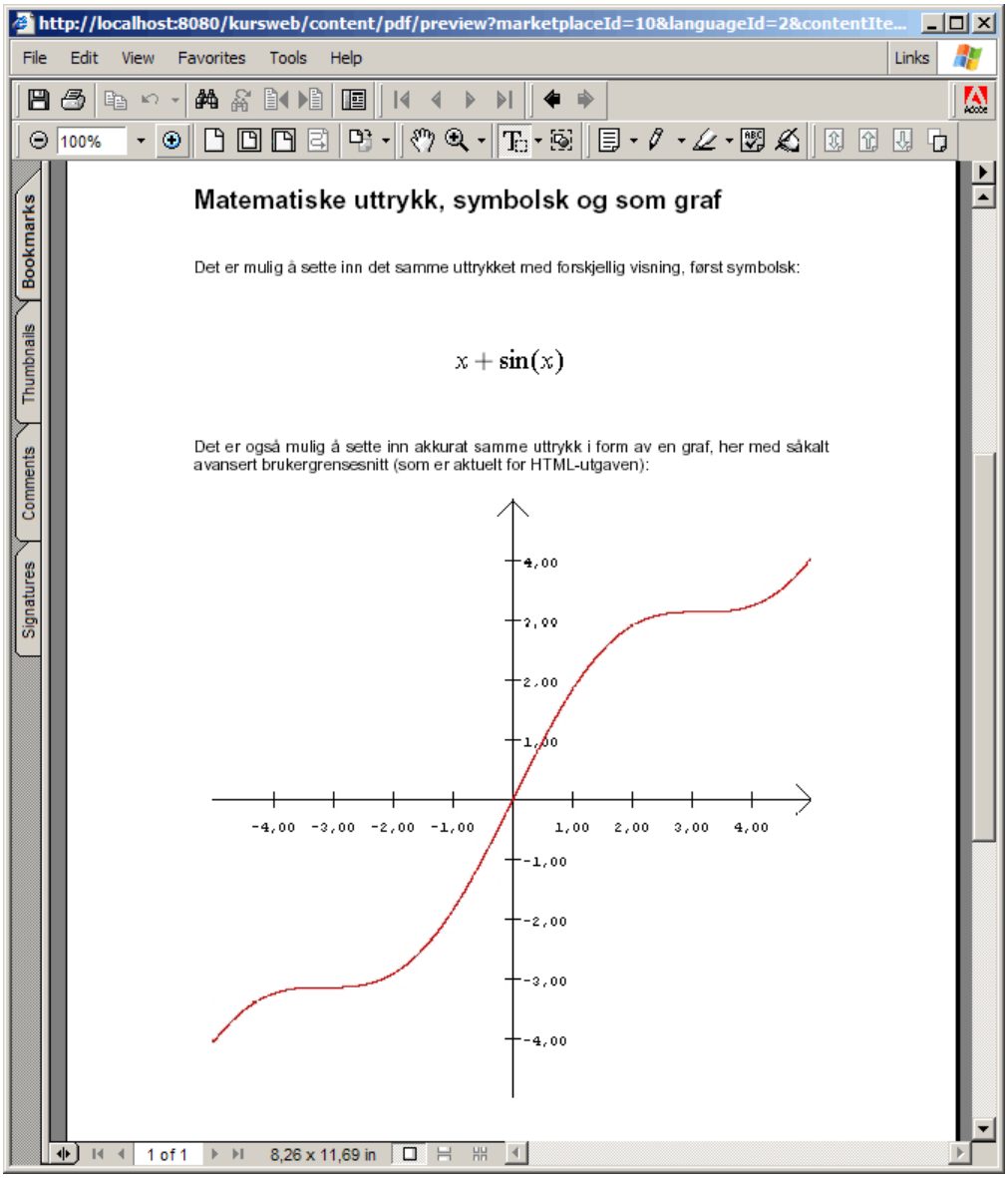
Under følger et skjermbilde fra HTML-versjonen av et dokument der samme matematiske uttrykk er satt inn, med forskjellige valg for visning. Altså er det et `symbolicMath`- og et `graphMath`-element i dokumentet, begge med et `mathObject`-element (som viser til det samme matematiske objektet) som barn.



Figur 23 – Matematiske uttrykk i form av graf og symbolsk visning i et dokument i HTML-format

I skjermbildet over er grafen tegnet opp i nettleseren på brukerens maskin. Dette gir muligheter til å utføre operasjoner som å zoome inn direkte i klienten. Brukergrensesnitt for dette kan skrus av og på via innstillinger i XML-koden for innsettingen av det matematiske uttrykket. Jeg kommer tilbake til dette lengre ned, men den observante leser vil kanskje allerede ved å se på DTDen, som ble vist i forrige kapittel, skjønne hvilke parametre dette gjelder.

Samme matematiske uttrykk kan altså brukes gjentatte ganger i ulike former. Som nevnt er det også slik at samme dokument kan tas ut i ulike formater. Dette fungerer også for dokumentet over som inneholder samme uttrykk både som graf og i symbolsk form. I prototypen gjelder dette HTML og PDF. Under følger et skjermbilde fra det samme dokumentet som over, men da i PDF-format produsert av prototypen fra samme lagrete kilde (både for dokument og innsatte matematiske uttrykk).

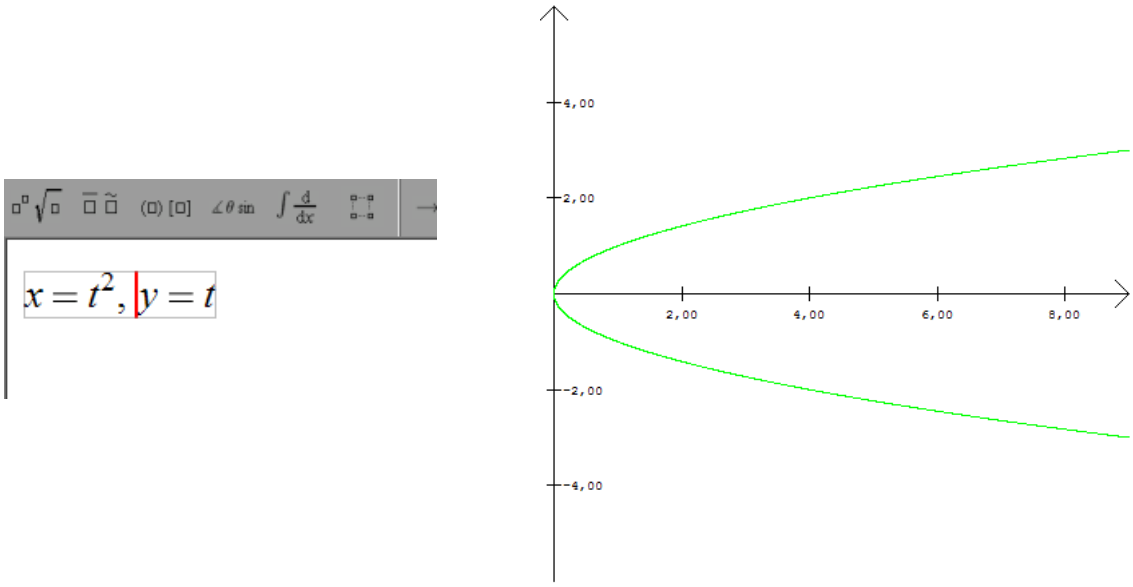


Figur 24 - Matematiske uttrykk i form av graf og symbolsk visning i et dokument i PDF-format

Grafen i bildet over er produsert ferdig til en grafikkfil på serversiden før den ble bygd inn i PDF-filen og oversendt klienten i ferdig stand. Klienten gjør her altså ikke noe annet arbeid

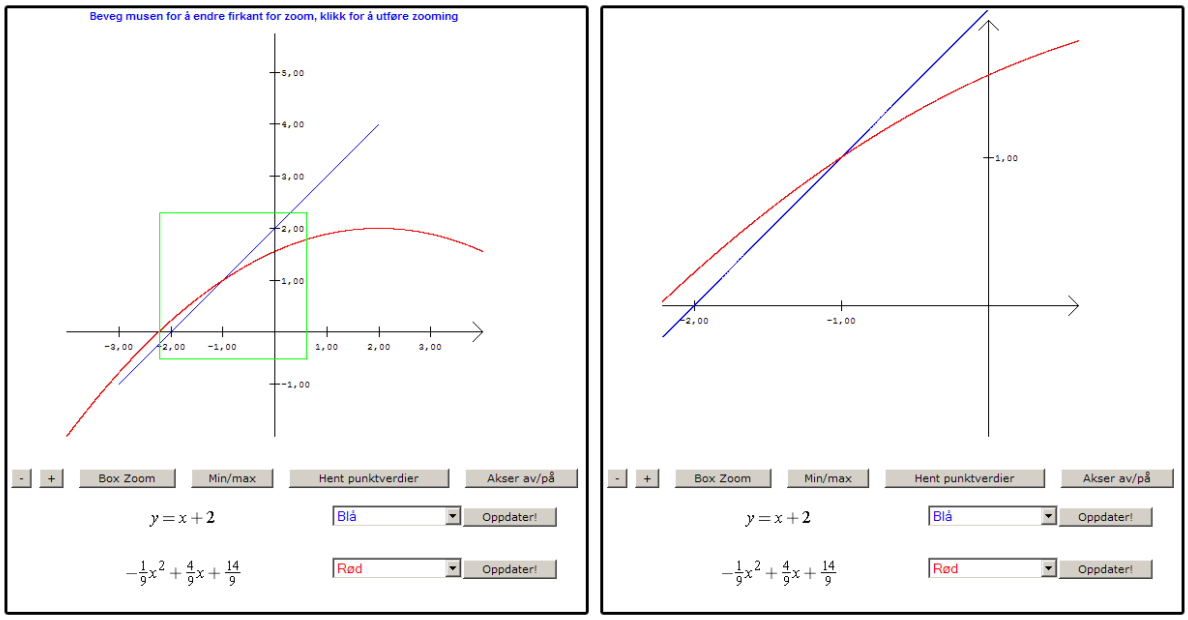
enn å ta i mot den ferdige PDF-filen og vise denne i et passende program. Det er verdt å merke seg at HTML-versjonen utnytter mulighetene for direkte interaksjon og dermed lar brukeren få mange ekstra muligheter, sammenlignet med hva som er mulig i PDF-versjonen (som har andre sterke sider). Dette gjøres altså fra eksakt samme XML-kilde, lagret i databasen.

Graftegneren er som nevnt i avsnittet i kapittel 1 om avgrensning av oppgaven begrenset til todimensjonale plottinger av funksjoner. I tillegg har jeg lagt til en mulighet for at uttrykket som skal plottes ikke er slik at det kan fremstilles som en enkel funksjon, men som en parameterfremstilling. Under er et enkelt eksempel på dette, med uttrykket til venstre i editoren og opptegningen av dette til høyre.



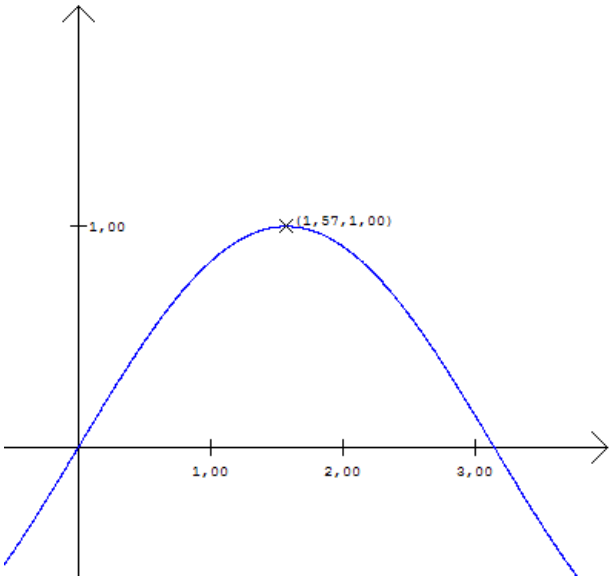
Figur 25 – Parameterbasert uttrykk tegnet i graftegnerprototypen

Graftegneren i klienten skal kunne tilby en del dynamikk. Siden dette i hovedsak er snakk om implementasjonsjobb uten mange interessante prinsipielle problemstillinger har jeg valgt å ikke bruke for mye tid på dette i forbindelse med utviklingen av prototypen. Jeg har likevel implementert enkelte eksempler på funksjonalitet som gjøres mulig av den valgte arkitekturen, blant annet innzooming ved hjelp av å tegne inn en firkant og markering av maksimalpunkt på grafen. Dette er vist i figurene på neste side.



Figur 26 – Zoom til valgt rektangel i prototypens graftegnerdel

Endringen i skjermbildet over skjer altså uten noen ny forespørsel til tjeneren, slik det ble diskutert i forbindelse med valget av arkitektur. Maksimums- og minimumspunkter kan vises i prototypen ved at brukeren velger dette, ved å bruke knappen som er synlig i nedre del av figuren overfor. Når brukeren klikker gir dette opptegning av punktene i grafen, som vist under.

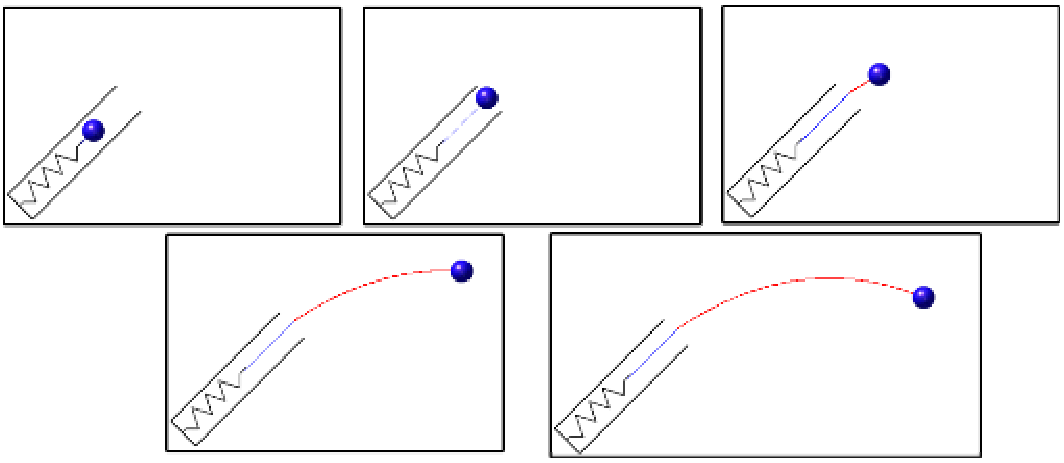


Figur 27 – Markering av lokal ekstremalverdi i prototypens graftegnerdel

Det er også i prototypen lagt til mulighet for at brukeren klikker på et punkt på grafen, for å få vite punktverdiene i det punktet. For eksempel på dette og funksjonaliteten som er vist over henvises det til eksemplene som er eksportert til vedlagt CD-ROM. For detaljer om filnavn og lignende, se appendiks C.

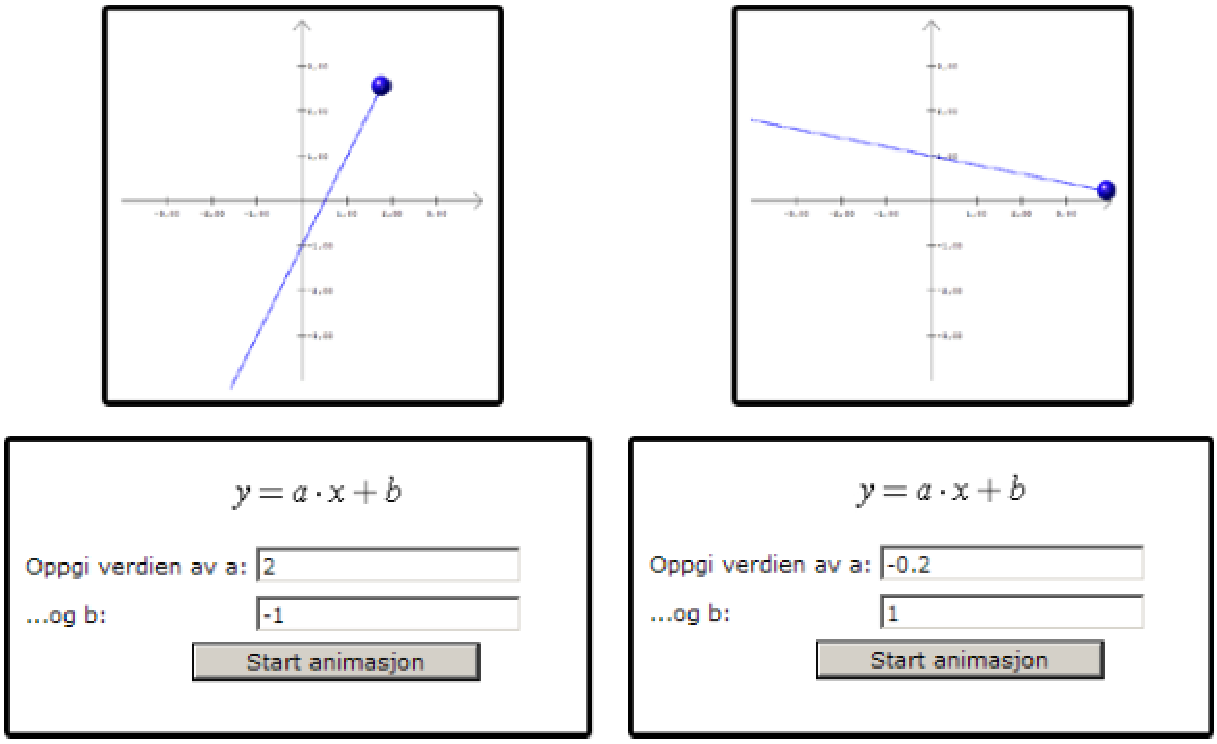
For bruk i pedagogisk sammenheng kan det være ønskelig å visualisere hva grafen representerer med mer enn bare en vanlig graf. For eksempel kan man ved å bruke bilder visualisere et kast ved hjelp av en ball der ballen faktisk følger banen (grafen) den ville gjort. Siden bruk mot videregående skole vil være blant de mest aktuelle for en slik graftegner, har jeg implementert støtte for slik bruk.

Når komponenten brukes i animasjonsmodus vil sluttbrukeren ha en knapp for å starte animasjonen, samt to knapper for å endre hastigheten. Når animasjonen startes tegnes grafen opp fra venstre mot høyre (i stedet for å tegnes komplett direkte). Under følger et skjermbilde for en animasjon der det i tillegg til å tegnes opp en graf med en gang tegnes en ball som følger banen til grafen. Jeg har også lagt inn støtte for bakgrunnsbilde, slik at ballen for eksempel kan ”sprette ut fra en fjær” rent visuelt.



Figur 28 – Bildeserie fra bruk av graftegneren i animasjonsmodus

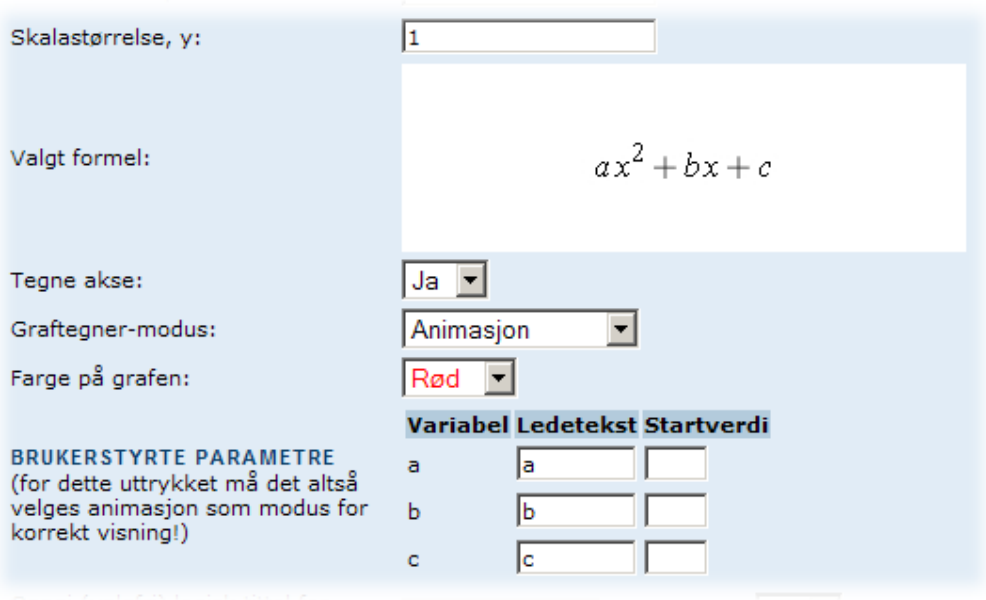
Et annet pedagogisk poeng er at brukeren kan lære gjennom å prøve seg frem med slike animasjoner og styre ulike parametre. For eksemplet med kast er vel vinkelen og utgangsfarten sannsynlige kandidater til brukertstyrte parametre. Dette støttes i prototypen gjennom at man legger inn uttrykk med variabler (utover x, y og t) som så byttes ut med verdier satt av brukeren før opptegning. Under har jeg laget et enkelt eksempel med brukerstyrt animasjon, ved at den rette linjen $y=ax+b$ kan animeres mens brukeren endrer verdiene for a og b.



Figur 29 – Brukerstyrte parametre med graftegneren i animasjonsmodus

En eksportert fil tilsvarende eksemplet under finnes også på vedlagte CD-ROM, se appendiks C for mer detaljer om filplassering, kildefiler i XML og lignende. I skjermbildene over er grafdelen forminsket betydelig mer enn grensesnittet der parametrene legges inn, dette for å få et bilde som, uten å ta alt for stor plass, viser funksjonaliteten på trykk.

Under vises hvordan skjermbildet for forfatteren av dokumentet blir for et uttrykk som inneholder parametre som skal være brukerstyrte. Disse parametrene detekteres automatisk og forfatteren får oppgi ledetekster og startverdier. Figuren under er et utsnitt fra tilsvarende grensesnitt som det som er vist i veiviseren for innsetting i kapittel 6.1.



Figur 30 – Spesifisering av ledetekst og startverdi for brukerstyrte parametre

Ledetekst og startverdi lagres i XML-dokumentet (låses altså ikke per uttrykk, men *per anvendelse*) i henhold til dokumentformatet som er dokumentert i "Figur 21 – Utvidelsen av DTDen for Insperas dokumenter for bruk av matematiske objekter".

6.3. Oppsummering av funksjonalitet i prototypen

Jeg vil i det følgende kort oppsummere funksjonaliteten i prototypen og sammeholde denne med kravene som ble satt opp innledningsvis. Hvis vi sammenligner funksjonaliteten i prototypen med kravene satt opp i kapittel 5.1 ser vi at alle kravene er oppfylt:

- Matematiske uttrykk kan vises i symbolsk form og som graf, fra samme lagrede kilde
- Graftegneren tilbyr brukeren rikelig med dynamikk og muligheter for interaksjon
    - Mulighet for å endre farge på grafer, skru visning av grafer av/på, sette skalastørrelse og styre om akser skal tegnes
    - Mulighet for å markere punktverdier på grafer ved å klikke med musen direkte i grafen eller bruke piltaster
    - Mulighet for å finne lokale minmum og maximum
    - Mulighet for zoom, både stegvis og til et egendefinert rektangel
    - Mulighet for å spille av en animasjon som følger grafbanen
    - Mulighet for brukerstyrte parametre til en animasjon for interaktiv læring
    - Forfatter kan styre grensesnittet eller velge å ikke ha noe grensesnitt for manipulering av grafen eller animasjon
- Matematiske uttrykk kan tas ut i versjoner både for utskrift og web i form av at sluttformatene PDF og XHTML støttes, dette gjelder også både symbolsk visning og visning som graf
- Prototypen kjører på Insperas plattform og er fullt integrert. Det eneste spesielle kravet til programvare er at nettleseren må[31] ha en javamotor, og det finnes for alle populære nettlesere og operativsystem.
- Bytte av tredjepartskomponenter kan skje relativt enkelt ved at det er klart definerte grensesnitt mellom de ulike komponentene, dette er forklart i detalj i kapittel 5.6.1.

I tillegg støttes både uttrykk der y er en funksjon x og uttrykk der x og y er parameterfremstillinger av t.

På flere av punktene overgår prototypens funksjonalitet selv de mer optimistiske anslagene fra starten av arbeidet med oppgaven. Selv om mye av funksjonaliteten er kjent og finnes i flere kommersielle produkter, er kombinasjonen av støtte for krysspublisering og mulighet for interaksjon, læring og animasjoner meg bekjent ikke tidligere presentert. I tillegg bygger prototypen på et standardisert format og mulighetene for å utnytte semantikken i det lagrede innholdet i andre programmer i fremtiden er stort.

I det følgende vurderes et annet aspekt med prototypen i tillegg til det rent funksjonelle, nemlig en nærmere vurdering av ytelse og skalerbarhet.

[31] Dersom de ikke har det vil de kunne få grafene i form av bilder i XHTML, men ingen av de avanserte mulighetene med interaksjon og lignende.

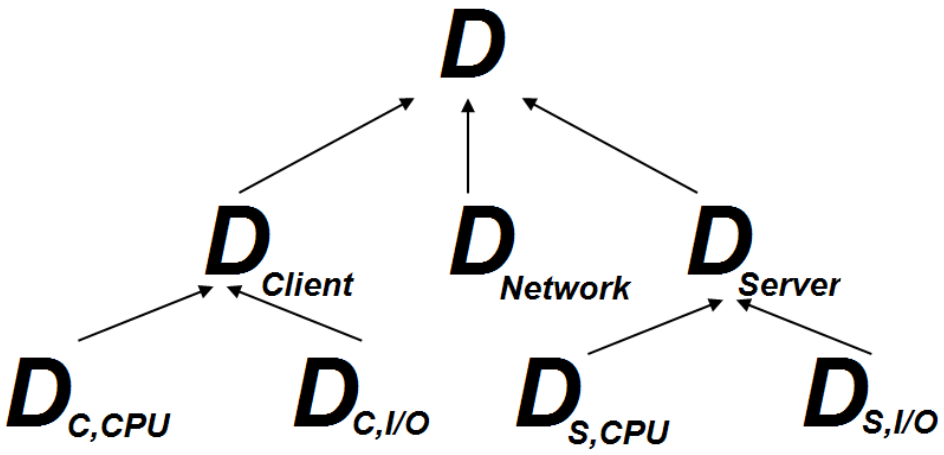
7. Vurdering av ytelse og skalerbarhet

Hovedmålet med ytelsestesten er å sammenligne ytelsen ved å utføre hele beregningen på serversiden med å flytte en del av ansvaret til klienten (som tidligere begrunnet må prototypen uansett støtte begge metoder av funksjonelle grunner for å kunne støtte krysspublisering og dynamikk i graftegneren). Med dette vil forskjellen som vi har antydet i forbindelse med valg av arkitektur blir kvantifisert. Et sekundært mål for vurderingen er å få et konkret måltall for antall samtidige brukere som er mulig per server med prototypen slik den foreligger (også for å besvare det konkrete spørsmålet som var stilt i oppgaveteksten).

Fokus er på graftegnerdelen, da den er den mest ressursintensive. Det er også lettere å få frem det prinsipielle ved å ikke benytte mange typer forespørsler samtidig.

7.1. Teoretisk modell for ytelsesvurdering

En mulig angrepsvinkel for å vurdere ytelsen til prototypen (og den underliggende modellen som er valgt i denne) er å dekomponere arbeidet som utføres ved en forespørsel. Som jeg tidligere har vært inne på blir lasten på de ulike ressursene relativt forskjellig for de to ulike løsningene jeg har skissert over. Under følger en skisse over dekomponeringen av påkrevet arbeid (såkalt service demand), der D_i er arbeidet krevet på/av del i i systemet. Her er også klientdelen tatt med, selv om vi senere kommer til å se nærmere bare på en utvalgt del. Dette gjør det lettere å belyse forskjellene mellom de to modellene for løsningen (ren serverside og distribuert, som forklart tidligere).



Figur 31 – Dekomponering av totalarbeidet krevet av en forespørsel (service demand)

D_i inkluderer ikke køtiden og er dermed uavhengig av lasten. Fra [MENASCÉ1998] har vi at maksimal gjennomstrømning λ er gitt ved:

$$\lambda \leq \frac{1}{\max_{i=1}^{K} D_i}$$

Altså vil gjennomstrømningen være begrenset av ressursen med størst verdi for arbeidsmengden. Dette stemmer også godt med hva vi ville antatt uten utledning; flaskehalsen bestemmer total gjennomstrømning.

I en webapplikasjon er bruken kjennetegnet ved at brukeren kommer med en forespørsel, før han studerer resultatet (leser innholdet på en webside, for eksempel) og deretter følger en ny forespørsel. Tiden der brukeren ikke har noen aktiv forespørsel til serveren kalles gjerne *think time* og vi bruker symbolet Z for gjennomsnittet av denne. Den inverse av denne gir oss da et tall for antall forespørsler per tidsenhet generert av én bruker.

Ved hjelp av verktøy[32] for optimering av kode kan man se tilnærmelsesvis hvor lang tid de ulike delene tok i min prototype og dermed få en indikasjon på hva tallene for de enkelte arbeidsmengdene (service demands) er. Tallene i seg selv er ikke så interessante, men (for så vidt som ventet) viser det seg at beregningen av tallverdier og plotting av disse i de ulike punktene tar en svært stor del av tiden. Altså er det henholdsvis $D_{C,CPU}$ og $D_{S,CPU}$ som blir flaskehals avhengig av hvor denne beregningen gjøres. Typiske tall for andelen av tid brukt til beregningen av punktene var 85%, men siden både beregningstiden og de andre tallene vil variere med fysisk utstyr må dette bare sees på som en grov indikasjon. Imidlertid gir den oss en pekepinn på at valget av arkitektur er svært vesentlig for hvor skalerbar en ferdig løsning basert på arkitekturen vil ha mulighet for å bli. I tillegg vil også kodingen av selve bildefilen for de fleste bildeformater være CPU-intensiv.

$D_{Network}$ vil også bli påvirket av plasseringen av beregningen. Selve overføringstiden knyttet til hver enkelt graf blir kuttet i størrelsesorden til en femtendedel[33] ved å overføre beregningen til klienten. Dette skyldes at det ikke blir overført noe bilde, men bare en ren-tekst representasjon i form av matematisk uttrykk og parametre.

Imidlertid er oppstartskostnadene større ved at man for førstegangs tegning av graf må overføre programkoden[34] for appleten som skal kjøres i klienten. Etter såkalt obfuskering[35] er denne er i størrelsesorden 20 ganger større enn kostnaden per graf i modellen med beregning i klienten (og dermed altså større enn kostnaden ved overføring av en enkelt graf i en ren servermodell). Forskjellen er at koden bare overføres første gang en bruker skal se en graf, så dersom brukeren ser på mange dokumenter med grafer vil gjennomsnittlig båndbredde per graf gå kraftig ned. For en ren serverside-modell vil nettverksoverføringen ved å sende over en graf etter en brukerinteraksjon være like stor som den initielle overføringen, mens denne vil være 0 for den distribuerte modellen.

[32] Verktøyet som ble benyttet heter JProbe, det kan prøves uten kostnad en viss periode. Produkthjemmesiden hos produsenten og forhandleren Quest Sofware er http://java.quest.com/jprobe/jprobe.shtml.

[33] Siden dette ikke er noe stort poeng har jeg ikke satt opp noe fullverdig testsett for å finne et eksakt tall, men for to eksempelgrafer var tallene 15640B og 27804B i form av bildefil i JPEG-format med 100% kvalitet og 1402B og 1434B i ukomprimert tekstformat ved oversending. De eksakte størrelsene vil avhenge av bildeformat og komprimeringsalgortime, komprimert med LZH-algortimen (GIF) vil størrelsen for bildefilen kunne komme ned mot 5000B, men også ren-tekst representasjonen ville kunne komprimeres kraftig ved oversending over nettet, noe jeg ikke har tatt hensyn til over – så alt i alt vil jeg tro anslagene er relativt realistiske. GIF (eller mer presist: LZH-komprimeringen) kunne jeg ikke bruke i prototypen på grunn av patentrestriksjoner. Jeg synes ikke det er naturlig å inkludere protokolloverhead og andre detaljer i en slik omtrentlig beregning jeg her har gjort. PNG-formatet egner seg godt, men siden ikke PDF-biblioteket per i dag støtter innebygging av PNG-filer kunne ikke dette brukes til tross for at det som nevnt ville kunne gi mindre filer uten å gi lisensproblemene til GIF-formatet.

[34] Helt konkret er dette snakk om jar-filen ”graphdrawer.jar”. Denne blir også brukt på CD-ROMen til praktisk demonstrasjon av graftegneren, se katalogen ”graphdrawer_samples” for størrelsen på den aller siste versjonen av filen.

[35] En prosess der metode- og variabelnavn byttes ut med korte (typisk enbokstavers) navn for å gjøre bytecoden (instruksjonene for den virtuelle maskinen) i tillegg til at (biblioteks-)funksjoner som ikke brukes strippes vekk. Dette gjør at det resulterende arkivets filstørrelse blir mindre, i dette tilfellet var effekten om lag 35% reduksjon i filstørrelsen.

Siden $D_{Network}$ er direkte proporsjonal med overført datamengde kan vi ved å se nærmere på sistnevnte finne verdier for når de ulike modellene er best med tanke på å minimere $D_{Network}$. For en bruker som ser på totalt n grafer i systemet og for hver grafvisning gjør m interaktive operasjoner (zoom, markering av punkter, max/min, m.v.) kan denne datamengden S skrives:

$$S = S_{startup} + n \cdot S_{graph} + m \cdot n \cdot S_{interaction}$$

Dersom vi vil regne med relative tall basert på de estimerte størrelsene i forrige avsnitt kan vi sette størrelsen på en graf som overføres til måltall 1. Vi får dermed følgende uttrykk for datamengden som overføres totalt, for den distribuerte modellen:

$$S_{distributed} = S_{startup,distributed} + n \cdot S_{graph,distributed} + m \cdot n \cdot S_{interaction,distributed} = 20 + n \cdot 1 + m \cdot n \cdot 0 = 20 + n$$

Tilsvarende for den sentraliserte modellen blir:

$$S_{centralized} = S_{startup,centralized} + n \cdot S_{graph,centralized} + m \cdot n \cdot S_{interaction,centralized} = 0 + n \cdot 15 + m \cdot n \cdot 15$$
$$S_{centralized} = 15 \cdot n \cdot (m+1)$$

Dermed kan man bestemme hvor lenge S_{server} vil være mindre enn S_{client}:

$$\begin{aligned} S_{centralized} &< S_{distributed} \\ 15 \cdot n \cdot (m+1) &< 20 + n \\ n(15m+14) &< 20 \\ n &< \frac{20}{15m+14} \end{aligned}$$

For m=0 (brukeren gjør altså ingen interaksjon etter førstegangs visning av grafen) blir da n=1,43. Altså vil modellen med beregning på klientsiden være overlegen med tanke på overføringstid så snart brukerne i snitt ser flere dokumenter med grafer per besøk, gitt forutsetningene jeg har forklart over med filstørrelser. For $m > 0$ vil klientsidemodellen gi lavest $D_{Network}$ for alle verdier av n og modellen er således å foretrekke med tanke på nettverkslast selv om brukerne kun ser på ett dokument som inneholder en graf.

$D_{S,I/O}$ vil ikke påvirkes av valg av arkitektur. Denne består i hovedsak av databaseprosessens lesing mot disk på serveren og er like stor uansett hvor beregningen gjøres.

Ut fra målte verdier for service demands ved lav last har man mulighet til å bruke en iterativ algoritme for å se på hvordan gjennomstrømning og responstid utvikler seg med flere brukere, såkalt MVA (mean value analysis). Dette er bl.a. beskrevet i kapittel 9.4 av [MENASCÉ1998]. (D_i vil ikke endre seg med lasten, men siden tiden i kø for den enkelte request endres vil responstiden endres.)

For en videre vurdering av ytelsen har jeg gjort en praktisk test ved hjelp av en syntetisk lastgenerator. Dokumentasjon av oppsettet for denne testen finnes under neste overskrift.

7.2. Praktisk gjennomføring av ytelsesmålinger

I denne ytelsesmålingen er det D_{Server} som vurderes. D_{Client} holdes utenfor og anses som en svart boks. Dette fordi det er skalerbarhet og krav til tjeneren som skal vurderes her. Som nevnt vil ikke $D_{S,I/O}$ påvirkes av valg av arkitektur. Det er altså i praksis en sammenligning av hvordan endringen i $D_{S,CPU}$ for de to modellene slår ut som gjøres i denne ytelsesmålingen.

Ytelsesmålingen ble gjort for et utvalg verdier for N, antall brukere. Hver måling ble typisk kjørt i 90 sekunder.

I ekseperimentoppsettet ble Z=30s satt (programmet operer ikke med ”think time”, men ”request delay” kan settes til et intervall som det velges tilfeldig blant, dermed blir effekten den samme). Simuleringen ved hjelp av en syntetisk lastgenerator skjer uten at det er lagt på annen trafikk, slik at tallene gjelder en server som er dedikert til kun denne oppgaven.

Maskinoppsettet for testen var som følger:

- Applikasjonsserver med Intel P4 Mobile 1.6GHz, 512MB RAM og 5400rpm IDE harddisk
- Databaseserver med Intel Celeron 800Mhz, 512MB RAM og 7200rpm IDE harddisk
- 7 klient-PCer som simulerte trafikk fra mange brukere, hver med ca. 1GHz CPU og mer enn nok minne til denne oppgaven

Programmet som ble benyttet for å simulere lasten var Microsoft Web Application Stress Tool versjon 1.1. Testoppsettet er lagret i filformatet til dette programmet (i tillegg til en forklaring i ren tekst) sammen med resulatene på vedlagte CD-ROM. Programmet støtter distribuering av lastgenereringen og innsamling av dataene til én PC som styrer testen.

Det viste seg ved testkjøring at det ved for mange brukere (som måtte til i det distribuerte tilfellet) ble klient-PCene en flaskehals på grunn av for mange tråder. Dette ble løst til dels med å øke antallet klienter til syv (startet med to) og til dels gjennom å korte ned Z og kutte N tilsvarende.

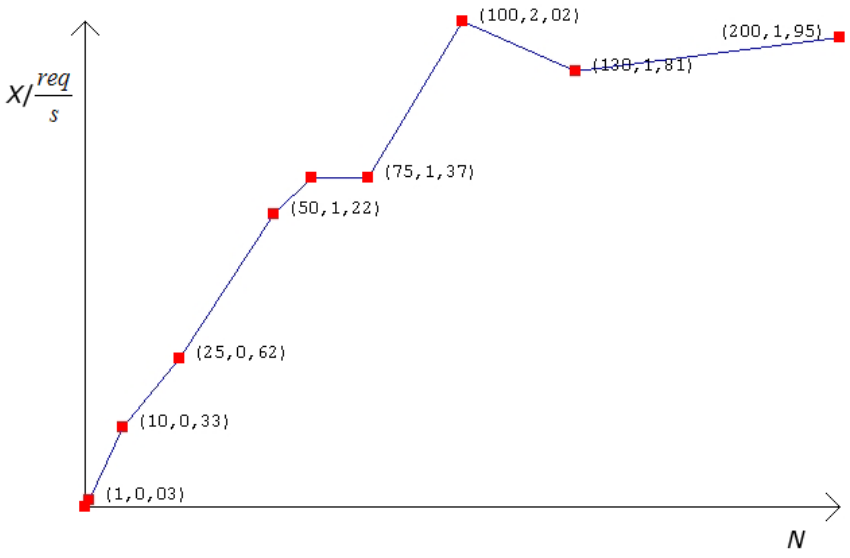
Applikasjonsserveren som ble brukt var Jakarta Tomcat, kjørt på MS Windows XP Professional. Denne applikasjonsserveren er ikke kjent for å være blant de aller raskeste, men som presisert tidligere er ikke de absolutte tallene som kommer ut av dette eksperimentet det viktigste – hovedpoenget er å få sammenlignet de to modellene. Dessuten er det bedre at tallet for maksimalt antall samtidige brukere er et underestimat enn motsatt.

Jeg valgte for oversiktens skyld å skille ut funksjonaliteten jeg ville gjøre den praktiske ytelsestesten mot i filen perfeval.jsp. Dermed er det lett å sammenligne hva som faktisk gjøres for de to modellene, rent teknisk, ved å inspisere denne filen. Filen er delt i to blokker for de to ulike modellene som velges med en parameter i URLen.

Testoppsettet med adresser som er benyttet er dokumentert fullt ut på CD-ROMen som er vedlagt, se appendiks D for filnavn og detaljer.

7.3. Testresultater

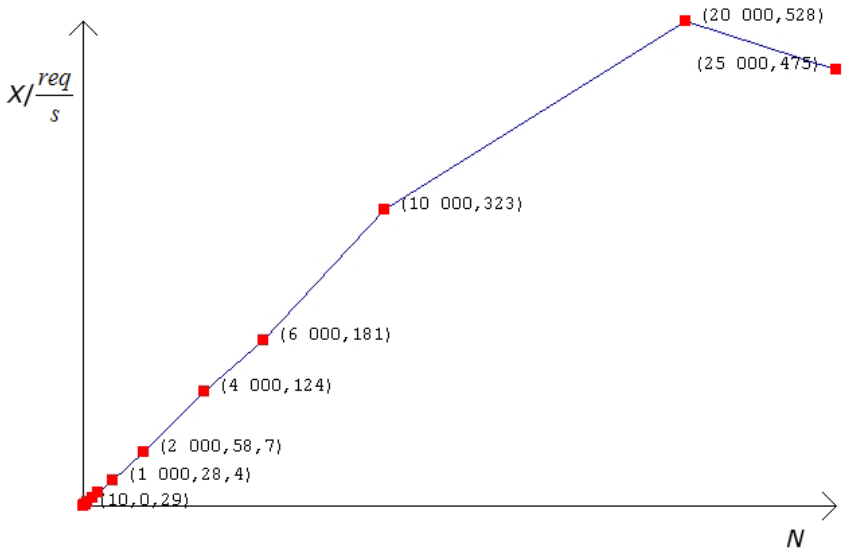
Følgende graf viser hvordan gjennomstrømningen (throughput) varierer med antall brukere når vi bruker en ren serverside prosessering:



Figur 32 – Gjennomstrømning som funksjon av antall brukere med sentralisert modell

Grafene i dette kapittelet er for øvrig laget ved hjelp av programvaren jeg har laget i forbindelse med denne oppgaven, som omtalt i slutten av kapitlet om klassemodellen. (Det var ikke noe stort poeng i å lage dette, men det ble gjort som en enkel test på at klassemodellens grunntanker faktisk lot seg gjennomføre i praksis.)

Tilsvarende for den distribuerte modellen gir følgende graf:

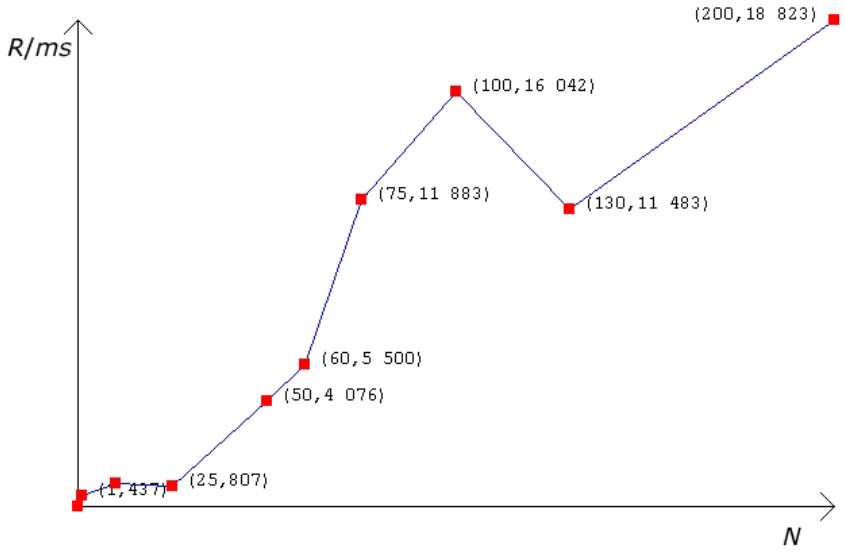


Figur 33 – Gjennomstrømning som funksjon av antall brukere med distribuert modell

Vi ser at gjennomstrømningen for ren serversidebasert arkitektur når et tak på om lag 2 forespørsler per sekund, dette skjer når det er 100 samtidige brukere. For arkitekturen der

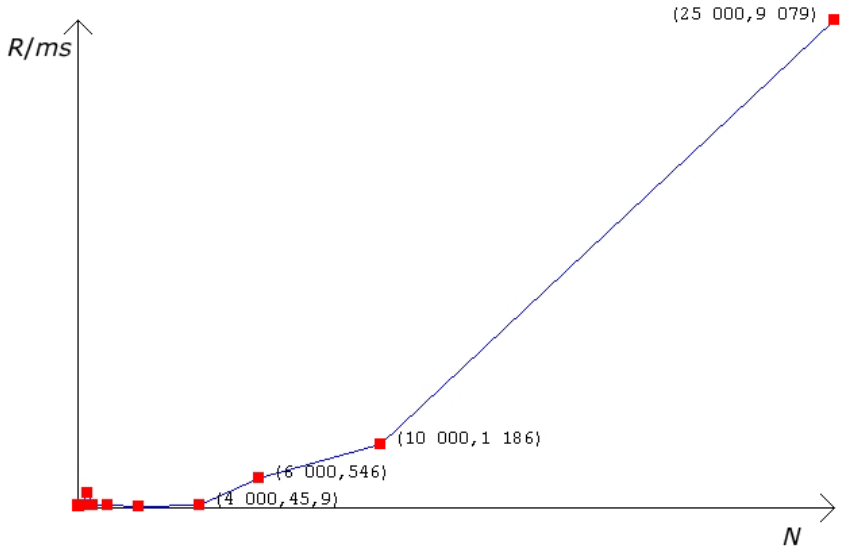
hoveddelen av beregningen gjøres i klienten er tilsvarende tall noe vanskeligere å si entydig. Dette fordi gjennomstrømningen fortsetter å øke også etter at feilene er for mange til at brukerne vil oppleve tjenesten som fullt fungerende. Men ved 6000 samtidige brukere er tjenestekvaliteten i hovedsak tilfredsstillende og gjennomstrømningen 546 forespørsler per sekund.

Under følger en graf der responstiden vises som funksjon av antall samtidige brukere i modellen der all beregning gjøres på serversiden.



Figur 34 – Responstid som funksjon av antall brukere med sentralisert modell

Tilsvarende graf for den distribuerte modellen ble som vist i figuren under.



Figur 35 – Responstid som funksjon av antall brukere med distribuert modell

Vi ser at responstiden for ren serversidebasert arkitektur passerer 5 sekunder når det er 60 samtidige brukere. I arkitekturen der hoveddelen av beregningen gjøres i klienten er tilsvarende tall godt over 10000. Imidlertid må vi her (som for gjennomstrømning) se på feilraten og ikke blindt godta maksimaltiden (for en bruker som ikke får svar hjelper det ikke at alle som får svar får det tilstrekkelig raskt). Det kan se ut som om dette er et sted mellom 4000 og 10000.

Det kan tenkes at man ved tuning av nettverksinnstillingene for applikasjonsserveren kunne bøtet på noe av problemene ved svært høy frekvens på forespørslene og dermed fått høyere aksepterte tall. Hovedmønsteret (ved at selve applikasjonsserveren blir flaskehals og ikke programkoden) er uansett at koden som kjører på serveren i den distribuerte modellen ikke er krevende.

På vedlagte CD-ROM finnes alle detaljer i testresultatene, i tillegg har jeg i appendiks D oppsummert tallene som ligger til grunn for grafene som er gjengitt over.

Feilkilder og andre ting å være spesielt oppmerksom på i forbindelse med den praktiske ytelsestesten kan kort sies å være:

- Overføring av programkoden er ikke med i den distribuerte modellen. Dette fordi denne statiske filen uansett ville blitt oversendt av webserveren og aldri blitt videresendt som en forespørsel til applikasjonsserveren. Det kunne vært et alternativ å sette opp en requestmix som også tok med denne, men tallene for throughput ville blitt kunstig høye ved at ren oversending av den statiske filen ble talt med.
- Databaseserveren vil i praksis bare levere data fra minnet siden det er gjentatte forespørsler om de samme dataene og disse derfor er cachet. Dette er imidlertid samme effekt som man vil ha i et reelt produksjonsmiljø i stor grad. Dersom man hadde hatt et svært stort datasett ville effekten blitt mindre. Et siste poeng er at databasetilgangen ikke var flaskehalsen.
- Klientkallene kan bli flaskehalsen når serverlasten per kall er lav. Dette detekterte jeg imidlertid og håndterte som forklart over.
- Tallene om maksimal responstid fra testverktøyet kan være noe villedende, siden en del forespørsler kan mislykkes i å gjennomføres (kommer under nettverksfeil). Tilsvarende gjelder for gjennomstrømning. Man bør derfor sjekke at ikke feilraten er for høy før man godtar en observasjon av denne typen.
- Det er viktig å være klar over at responstiden som måles i de to tilfellene ikke er direkte sammenlignbare (sett fra brukerens ståsted) fordi det i det distribuerte tilfellet ikke er tatt med tid til beregning og opptegning, som gjøres i klienten. Dette er for så vidt også tilfelle for ren tjenerside modell – det er ikke medregnet tiden nettleseren bruker på å dekode og vise bildefilen, men det er rimelig å anta at visningen av et ferdig bilde krever ressurser av en helt annen, og mindre, størrelsesorden enn opptegningen.
- Det kan se ut som testverktøyet har problemer med å gi presise tall der tallene har lav absolutt verdi (som er tilfellet i det distribuerte tilfellet). Målt gjennomsnittlig responstid varierer for eksempel tilsynelatende tilfeldig mellom de ulike innstillingene for antall brukere i den lavere enden (mens tjeneren har minimalt med last).
- Rene tall for gjennomstrømning gir liten verdi uten å samtidig vurdere tjenestekvalitet, dette henger også sammen med punktet om feilraten over.

7.4. Konklusjoner på basis av utført ytelsestest

Det var ventet at det skulle være forskjell på de to modellene. Å flytte prosessering fra tjener til klient vil nødvendigvis frigjøre ressurser på tjeneren, nettopp dette var jo også et av argumentene i forbindelse med utforming av arkitekturen for graftegneren. Undertegnende ble riktignok litt overrasket over størrelsen på forskjellen og slik sett var det nyttig å få kvantifisert den.

Ytelsestesten og forutgående resonnement viser altså følgende effekter ved å flytte deler av prosesseringen til klienten:

- Serveren kan håndtere inntil hundre ganger flere brukere
- Kravet til båndbredde er svært avhengig av bruksmønster, men i de fleste tilfeller vil båndbreddekravet være mindre for den distribuerte modellen. For utdypning vises det til utregningene over.

I tillegg kommer de funksjonelle effektene (mulighetene for å lage dynamiske effekter m.m.), dette er blitt behandlet i full bredde tidligere, teknisk i kapittel 5.5 og sett fra brukerens ståsted i kapittel 6.2.

Det skal også påpekes at dersom vi i den praktiske testen hadde sette på et scenario der brukeren først fikk tegnet opp grafen (som nå) for deretter å zoome inn på en del av grafen (altså tilsvarende $m > 0$ i resonnementet om nettverkslast), ville forskjellen i ytelse mellom de to arkitekturene blitt enda større. Dette fordi steg to med distribuert modell ikke ville ført til last på serveren i det hele tatt, mens det med sentralisert modell ville vært like mye jobb for serveren en gang til – forskjellen ville altså grovt sett blitt dobbelt så stor. Slik kunne vi fortsette resonnementet med flere interaksjonsrunder (markering av maksimum, skru av/på visning av en graf, osv.).

Vi noterer oss at kravene til dynamikk som vi satte opp innledningsvis er ivaretatt. Oppgaveteksten ber også om en vurdering av om hvorvidt prototypen egner seg til bruk i større skala. På dette spørsmålet må svaret bli at dersom man skal skalere til svært mange samtidige brukere må man velge den distribuerte modellen. Ekstensiv bruk av den sentraliserte modellen vil gi uforholdsmessig store krav til maskinvare på serveren, dersom det er svært mange samtidige brukere.

Som dokumentert i denne rapporten er modellen for prototypen vurdert relativt nøye. Det er imidlertid ikke gjort optimalisering av koden på et finkornet granualitetsnivå for å få den til å kjøre så raskt som mulig. Noen håndgrep er gjort (blant annet bruker jeg som nevnt en variant av ”lazy evaluation” som gjør at punktverdier bare beregnes på nytt når de må), men det vil helt sikkert være noe å hente på å ta en gjennomgang av koden for å optimere den for høy ytelse.

8. Avslutning og oppsummering

8.1. Nytteverdi av oppgaven

Jeg har i oppgaven presentert og vurdert alternativer ved valg av markupbasert lagring av matematiske uttrykk for bruk i digitale biblioteksystemer eller på web. Mitt valg av format vil kanskje ikke passe alle, men ved å kombinere presentasjonsrettet og semantisk markup kombinerer man de to fundamentale behovene i forbindelse med matematiske uttrykk på elektronisk form. Siden det stadig er utvikling både i standarder og ikke i minst i adopsjonen av disse, vil over tid en gjentatt vurdering av hva som er mest formålstjenelig uansett være klokt.

Prototypen som er utviklet mangler ikke mye på å kunne nyttiggjøres i praksis, for eksempel som et hjelpemiddel i den videregående skolen. I tillegg til den faktiske funksjonaliteten håper jeg at diskusjonen rundt arkitektur og valgene jeg har gjort kan være nyttig for andre som skal implementere lignende funksjonalitet – selv om de ikke nødvendigvis gjør de samme prioriteringer og valg.

Ytelsestesten og resonnementene rundt denne viste som ventet at det er forskjell på den distribuerte og sentraliserte modellen med tanke på ytelse. Det var likevel interessant *hvor* stor forskjellen mellom de to modellene ble. Dette kan være en tankevekker for andre som er i analysefasen i utformingen av sammenlignbare systemer. Reversering av slike fundamentale valg blir gjerne kostbare (i alle fall i form av tid, om ikke alltid penger), dersom de ikke gjøres allerede på tegnebrettet.

Prototypen som sådan har ikke et gjennomtenkt brukergrensesnitt. Dette var heller ikke meningen, da den er ment mer eller mindre som en teknologidemonstrasjon og ikke et produkt for sluttbruker. Imidlertid har (som tidligere forklart) enkelte deler av prototypen direkte nytteverdi.

Jeg mener også at deler klassemodellen jeg har utformet har bruksområder utover ren opptegning av grafer til matematiske uttrykk, noe jeg fikk en indikasjon på ved at jeg kunne lage programvaren for å tegne grafene til ytelsestesten enkelt og greit i rammeverket. Jeg håper at jeg har klart å forklare tankegangen bak slik at andre også kan benytte konstruksjonsmønstrene i andre sammenhenger hvor de kan være nyttige.

8.2. Videre arbeid

Adopteringen av markupspråkene for matematikk er som nevnt ikke fullført og det kan tenkes at trender og muligheter endres slik at utvekslingsverdien blir større med andre metoder enn de jeg har valgt. Å overvåke internasjonalt standardiseringsarbeid og ikke minst utberedelsen av de ulike formatene (som kjent øker potensielle utvekslinger eksponentielt med antallet som bruker et format) vil derfor fortsatt være viktig.

Som en del av denne oppgaven har jeg laget en enkel tekstbasert versjon av de matematiske uttrykkene. Denne kan om ønskelig vises sluttbrukere med mobile terminaler og enkelte håndholdte terminaler vil kunne bruke sluttformatene som dagens prototype lager mer eller mindre direkte. En mer omfattende analyse av behovene for bruk av matematiske uttrykk fra

mobile og håndholdte enheter faller imidlertid utenfor denne oppgaven, men bør være et interessant område både for teoretisk arbeid og implementasjon.

For graftegner-delen av arbeidet er det rikelig med oppgaver som kan gjøres, men de fleste er mindre prinsipielt interessante (mer en ren implementasjonsjobb). Dette inkluderer:

- Regresjon, det vil si å tilpasse kurver (finne funksjonsuttrykket) til gitte punkter
- Utregning av bestemte integral, gjerne med markering av areal i tegnekomponenten
- Inntegning av tangenter i punkter, samt finne funksjonsuttrykk for disse

En mer omfattende utvidelse vil være en utvidelse med støtte for tegning av romlegemer i tillegg til todimensjonale grafer. Her vil man også komme innpå mer arkitekturmessige utfordringer, selv om det (som kommentert i dokumentasjonen) for så vidt er forsøkt å legge noe til rette for slik utvidelse.

Utvidelsen jeg gjorde som gjorde at jeg kunne bruke min egen prototype til å lage grafene i kapittel 7.3 fungerte godt. Imidlertid er det noen mindre ting som burde vært endret rent teknisk hvis man skulle gjøre mange lignende utvidelser (for å gjøre vedlikehold og feilrettinger enklere ved at man bare endrer ett sted). Dette ville bli en felles superklasse for GraphDrawerLine og GraphDrawerPlot siden disse har ganske mye felles (siden funksjonaliteten ikke deles med tegneklasser av helt andre typer, slik som den tenkte GraphDrawerBarChart, bør implementasjonen ikke flyttes til GraphDrawer).

Tilsvarende ville man kunne unngå duplisering av kode mellom de ulike implementasjonene av PointValueProducer ved å innføre et mellomnivå for familier av lignende klasser (en ListbasedPointValueProducer vil for eksempel kunne samle relativt mye av det som i dag er duplisert). Dette bør gjøres dersom systemet skal utvides kraftig i de retninger som det her er forsøkt tatt høyde for, men det er ikke vesentlig i prototypen som er laget i dette prosjektet.

Prototypen gir kvalitet i visning av symbolske uttrykk som er et klart fremskritt i forhold til mye av det som publiseres på nett i dag. Den er også god nok til å brukes til generering av PDF for elevoppgaver og lignende. Imidlertid gir den ikke kvalitet på et nivå som gjør det mulig å publisere profesjonelle bøker direkte fra systemet. Her vil jeg tro at de vektorbaserte komponentene jeg har sett på, med SVG som sluttformat, har stort potensial så snart de blir mer modne. En enklere utvidelse vil være å generere svært store punktbaserte bilder, for så å sette de inn i mindre og nedskalerte versjoner (for å gi bedre oppløsning). Dette vil ikke kreve noe ny funksjonalitet på javasiden, men bare en endring i selve XSL-malen som brukes for oversetting til XSL/FO og dermed PDF.

For grafene vil en enkel utvidelse være å legge til støtte for PNG-formatet, som har mange av de gode egenskapene jeg tidligere har nevnt[36] med GIF, men ikke er beheftet med samme begrensninger i form av patentrettigheter. Dette vil som nevnt kreve en mindre endring av biblioteksfunksjonaliteten for generering av PDF, siden dagens versjon ikke støtter innebygging av PNG-bilder.

På sikt er det et mål at sluttbrukere skal kunne nyttiggjøre seg representasjonen i MathML (eller annet markupformat med støtte for matematisk semantikk for den del) direkte. Dette krever som forklart tidligere ingen større endringer i arkitekturen eller løsningen, men bare en mindre endring av implementasjonen for visning.

[36] Muligens relativt godt skjult. Det er omtalt i fotnote 33.

8.3. Konklusjon og avslutning

Jeg har i oppgaven gjort et valg av et format for å støtte begge behovene (semantikk og presentasjon) for representasjon av matematikk. Formatet var en kombinasjon av content- og presentation MathML. Jeg har også valgt og dokumentert en arkitektur for bruk av dette formatet til symbolsk visning av matematiske uttrykk og tegning av grafer.

Jeg har videre laget en fullverdig prototype som bruker det valgte formatet og viser hvordan dette kan brukes både til å vise uttrykket symbolsk og utnytte semantikken i det lagrede uttrykket for å tegne opp en graf. Krysspublisering fra samme lagrede matematikk-kilde til ulike sluttformater støttes både i form av grafer og symbolske uttrykk. For graftegerens del tilbyr prototypen dynamikk blant annet i form av animasjoner, zoom og muligheter for markering av punkter på graf.

Prototypen er fullt fungerende og alle mål som var satt til denne ved prosjektstart er nådd.

Ytelsestesten viste klart at potensialet for skalering til bruk fra svært mange brukere er størst ved å bruke arkitekturen der beregningen blir distribuert ved at klienten selv beregner og plotter punktverdiene. Forskjellen var stor, med en faktor på om lag 100. Dette er i seg selv svært lite overraskende, men sammen med mulighetene for dynamikk og direkte tilbakemelding i nettleseren gjør dette at jeg konkluderer med at valget om å bruke den mest distribuerte modellen var klokt.

Jeg håper mine betraktninger om format og behov vil være nyttig for andre, samt at mine ideer og løsninger kan gjøre at flere får realisert mulighetene ved publisering av matematisk innhold på web.

9. Kilder og referanser

[ABRAMOWITZ1972]
Milton Abramowitz, Irene A. Stegun – Handbook of Mathematical Functions With Formulas, Graphs, and Mathematical Tables (10th edition)
National Bureau of Standards Applied Mathematics Series No. 55, ASIN 0160002028

[BERNERS-LEE1998]
Tim Berners Lee – The semantic web (W3C Architecture Note)
http://www.w3.org/DesignIssues/Semantic.html

[BERNERS-LEE2001]
Tim Berners-Lee, James Hendler, Ora Lassila – The Semantic Web
Scientific American 17. mai 2002

[BJØRNEBEKK2001]
Sondre S. Bjørnebekk, Kjell O. Skarbø – Krysspubliseringssystem for Internett basert på XML, Oracle og Java
Hovedprosjekt Høgskolen i Bergen 2001, tilgjengelig fra biblioteket ved HiB

[BROOKSHAW2003]
Leigh Brookshaw sin webside
http://www.sci.usq.edu.au/staff/leighb/

[CAPROTTI1999]
O. Caprotti, D. Carlisle – OpenMath and MathML: Semantic Mark Up for Mathematics
ACM Crossroads, http://www.acm.org/crossroads/xrds6-2/openmath.html

[CARLISLE2001]
D. Carlisle, J. Davenport, M.Dewar, N.Hur og W. Naylor - Conversion between MathML and OpenMath
http://www.openmath.org/cocoon/openmath/documents/om-mml.pdf

[CLARK1997]
James Clark – Comparison of SGML and XML (W3C Note)
http://www.w3.org/TR/NOTE-sgml-xml

[COCOON2003]
Understanding Apache Cocoon, fra websiden til Apache Cocoon-prosjektet
http://xml.apache.org/cocoon/userdocs/concepts/index.html

[COULOURIS2001]
G. Coulouris, J. Dollimore, T. Kindberg – Distributed Systems – Concepts and Design (3rd edition)
Addison-Wesley, ISBN 0201-619-180

[DESSCI-MOW]
Design Science Inc. – Strategies for Math on the Web
http://www.dessci.com/en/reference/webmath/strategies.htm

[DOCBOOK4.2]
DocBook XML 4.2, websiden for siste versjon av DocBook-DTDen
http://www.docbook.org/xml/4.2/index.html

[DOCBOOK4.2]
OASIS DocBook TC, websiden til komiteen som vedlikeholder DocBook-DTDen
http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=docbook

[FRONTER2003]
Pressemelding fra Fronter AS, 12.02.2003: ”8 av 10 elever mener skolearbeidet er mer interessant når de bruker dataverktøy i undervisningen”
http://fronter.com/fronter/personal/index.phtml?show_news_all=&expand=4904&user=web_news2_no

[GAMMA1994]
Erich Gamma, Richard Helm, Ralph Johnson, og John Vlissides - Design Patterns
Addison-Wesley, ISBN 0-201-63361-2

[GURARI2000]
Eitan M. Gurari og Sebastian Rahtz – From LaTeX to MathML and Back with TeX4ht and PassiveTeX
http://www.cis.ohio-state.edu/~gurari/docs/mml-00/mml-00.html

[HAROLD2002]
Elliotte Rusty Harold og W. Scott Means – XML in a Nutshell, Second Edition
O'Reilly & Associates, ISBN 0-596-00292-0

[INSPERA2003]
Inspera AS, Medie-uavhengig krysspublisering – webpresentasjon
http://www.inspera.no/kursweb/content/?contentItemId=154835&marketplaceId=7&languageId=1

[OPENMATH2002]
O.Caprotti, D.P.Carlisle og A.M.Cohen (red.) – The OpenMath Standard (version 1.1b)
http://www.openmath.org/cocoon/openmath//standard/om11/omstd11.pdf

[OPENDOCWEB]
OMDoc: A Standard for Open Mathematical Documents, websiden for OMDoc-DTDen
http://www.mathweb.org/omdoc/index.html

[KNUTH1986]
Donald E. Knuth – The $T_EX$book.
Addison-Wesley, ISBN 0-201-13448-9.

[KOHLHASE2003]
Michael Kohlhase – OMDoc: An Open Markup Format for Mathematical Documents (Version 1.1)
http://www.mathweb.org/omdoc/omdoc.ps

[MENASCÉ1998]
Daniel A. Menascé og Virgilio A. F. Alemeida – Capacity Planning for Web Performance: Metrics, models and Methods
Prentice Hall, ISBN 0-13-693822-1

[RAYMOND2001]
Eric S. Raymond – The Cathedral & the Bazaar, Musings on Linux and Open Source by an Accidental Revolutionary
O'Reilly & Associates, ISBN 0-596-00131-2

[W3C-MATHML2]
W3C – Mathematical Markup Language (MathML) Version 2.0
http://www.w3.org/TR/MathML2/

[W3C-MATHMLOV]
W3C – Putting mathematics on the Web with MathML
http://www.w3.org/Math/XSL/Overview.html

[W3C-MIMP2001]
Patrick Ion (red.) – MathML 2.0 Implementation and Interoperability Report
http://www.w3.org/Math/iandi/

[W3C-MIMPWEB]
W3C – MathML Software, webside med oversikt over tilgjengelig programvare for matematikk på web
http://www.w3.org/Math/implementations.html

[W3C-XHTML]
W3C – XHTML™ 1.0 The Extensible HyperText Markup Language (Second Edition)
http://www.w3.org/TR/xhtml1/

[W3C-XLINK]
Steve DeRose, Eve Maler og David Orchard (red.) – XML Linking Language v. 1.0
http://www.w3.org/TR/xlink/

[WILKINS1995]
David R. Wilkins – Getting Started with LaTeX
http://www.maths.tcd.ie/~dwilkins/LaTeXPrimer/

[WOLFRAM2003A]
About Steven Wolfram, fra www.stephenwolfram.com, Stephen Wolframs websider
http://www.stephenwolfram.com/about-sw/

[WOLFRAM2003B]
What is Mathematica?, fra www.wolfram.com, Wolfram Research sine websider
http://www.wolfram.com/products/mathematica/introduction.html

Appendiks A – Oppgavetekst

Matematiske uttrykk i digitale biblioteksystemer

I forbindelse med nettbasert læring og andre former for publisering på nett er det ofte behov for å *lagre* og *visualisere* matematiske uttrykk. Denne diplomoppgaven skal vurdere ulike alternativer for lagring av matematiske uttrykk og visualisering av disse. Det skal også lages en prototype-implementasjon av lagring og visualisering basert på MathML og gjøres en vurdering av om denne metoden er brukbar ved stor trafikk.

Oppgaven går ut på å:

1. Redegjøre for alternative representasjonsmåter for matematiske uttrykk i datamaskiner
2. Lage en prototype på lagring og visning av slike uttrykk basert på W3C sin MathML
3. Lage en prototype på visualisering lagrete matematiske uttrykk i form av grafer.
4. Vurdere prototypenes egnethet for bruk i større skala, herunder med kvantifiserte tester av ytelse og skalerbarhet.

Implementasjon og databasemodellering (Oracle) gjøres i Inspera Platform rammeverket (J2EE).

Appendiks B – Kildekode og dokumentasjon av løsning

Vedlagt CD-ROM

Utskrift av kildekode for prototypene i Java og database ville bli alt for omfangsrikt å ta med som vedlegg her. På vedlagte CD-ROM (se bakerste perm) finnes all kildekode for prototypen i Java og databasescript for å lage datamodellen slik den er beskrevet i denne rapporten. Det finnes også dokumentasjon av kildekoden i HTML-format for løsningen. Katalogstrukturen på CD-ROMen er som følger:

report/	Inneholder denne rapporten i ulike format, samt figurene og eksemplene som separate filer, annen dokumentasjon og datamodellen i Rational Rose-format.
javadoc/	Inneholder dokumentasjon av all kode (Java) i HTML-format. Dette tilsvarer utskriften under, men er noe mer detaljert og er i tillegg krysslinket og egner seg derfor bedre for navigasjon.
src/java	Inneholder all Java kildekode produsert i prosjektet og dokumentert i denne rapporten. Som nevnt deler serversidedelen og appletdelen mye kode og appletkoden er plassert i samme kodetre som serverdelen.
src/sql	Inneholder SQL-scripts (Oracle) for opprettelse av tabeller, sekvenser og prosedyrer (pakker) på databaseserveren.
src/misc	Inneholder deler av løsningen som ikke faller under punktene over, dette inkluderer javascript- og XSL-filer
stresstest/	Inneholder filene for oppsett av den praktiske delen av ytelsestesten. Informasjon om kjøringene og resultatene ligger også inne i filene. For fullt utbytte bør man nok ha installert nevnte stresstest-program, men jeg har også trukket ut informasjon til rene tekstfiler med tanke på lesbarhet uten dette.
graphdrawer_samples/	Inneholder eksporterte eksempler på bruk av graftegneren, disse er omtalt i et eget appendiks

En del av løsningen forutsetter en installert versjon av Inspera sin plattform for å fungere 100% direkte (da databasetabellene referer til tabeller der og tilsvarende for Java-klasser). Dersom en praktisk demonstrasjon er ønsket er jeg bare svært glad for å få vise frem prototypen for interesserte. Ta i så fall kontakt, for eksempel via e-postadressen sondresb@idi.ntnu.no.

Utskrift av forenklet JavaDoc

Kun klassene som er nyutviklet for prototypen i dette prosjektet er tatt med her. Den korte beskrivelsen under avsnittet i hoveddelen av rapporten antas å være tilstrekkelig for basisklassene som er benyttet, da disse ikke er direkte relatert til løsningen som sådan, men kun benyttet for å få elegant integrasjon med løsningen ved testkjøring.

På CD-ROMen er derimot også flere av klassene som allerede eksisterte før prototypen tatt med.

Jeg vil samtidig anbefale å bruke versjonen av JavaDoc som finnes på CD-ROMen, siden denne versjonen er krysslenket med ”Se også”-referanser og indeksert og derfor mye lettere å navigere i enn denne trykte versjonen/utdraget. Grunnen til at jeg inkluderer den trykte versjonen er at rapporten i seg selv skal gi en så god dokumentasjon av prototypen som mulig.

Dersom du leser en elektronisk utgave av rapporten og ikke har tilgang til nevnte CD kan du også se den elektroniske utgaven på følgende webadresse:

http://www.stud.ntnu.no/~sondresb/diplom/javadoc

Hvis rapporten leses om såpass lang tid at denne adressen ikke lengre er gyldig, sender jeg gjerne kopi av JavaDoc til interesserte. Ta i så fall kontakt (om ikke kontaktinformasjonen her skulle stemme om noen år er jeg sikker på at Google finner meg)!

Rekkefølgen av klassene er (tallene i parentes er sidetallet):

ContentMath (76), PointValue (80), PointValueProducer (83),
ContentMathPointValueProducer (86), PointValueProducerFactory (91),
GraphDrawer (92), GraphDrawerPlot (97), GraphDrawerSettings (105),
GraphSettings (113), MathToolsServerside (118), ContentMathViewer (120),
ContentMathPlainViewer (122), ContentMathImageViewer (123),
GraphDrawerApplet (125), GraphDrawerAppletSettings (138), MathInterval (141),
MathDataAccess (146), MathDisplayServlet (150), MathException (152)

no.inspera.applications.publish.datamodel

Class ContentMath

```
java.lang.Object
  |
  +--no.inspera.kernel.BaseObject
        |
        +--no.inspera.applications.publish.datamodel.ContentRevision
              |
              +--no.inspera.applications.publish.datamodel.ContentMath
```

public class **ContentMath**
extends no.inspera.applications.publish.datamodel.ContentRevision
implements no.inspera.services.search.Searchable,
no.inspera.applications.publish.util.PublishConstants

This class represents a revision of a mathematical object. It provides ways to access the underlying XML, currently MathML, both presentational and semantic parts.

It also contains a method for getting a plaintext-representation of the expression in the content-part (semantic part) of the MathML. It does this by traversing the expression tree.

The class is used only on the server side.

Version:
$Revision: 1.5 $ $Date: 2003/05/18 20:21:48 $
Author:
Sondre Skaug Bjørnebekk

Field Summary

private static org.apache.log4j.Category	**logCat** The logger for writing messages to the log, provides central control on the level of logging as well as display of the originator of the log entry.

Constructor Summary

ContentMath() Standard ctor which creates a new ContentMath-object without any initial data.

Method Summary	
long	**getContentMathId**() Gets the contentMathId by getting the objectId.
private java.lang.String	**getContentMathML**() Gets the content-MathML from this mathematical object.
org.apache.lucene.document.Document	**getDocument**() Implementation to facilitate indexing (and thereby searching) via Jakarta Lucene.
private java.util.ArrayList	**getElementsFromList**(org.w3c.dom.NodeList l) Helper method that returns only the Element-type Nodes from the given list.
java.lang.String	**getPlainTextMath**() Gets a plaintext-representation of the mathematical expression, suitable for the plaintextparser used to compute the values.
private java.lang.String	**getPlainTextMathFromSubTree**(org.w3c.dom.Element e) Helper method that gets a plaintext-representation of the mathematical expression subtree starting with e.
java.lang.String	**getPresentationMathML**() Gets the math expression in the form of presentation-based MathML, used for editing and creating images of the symbolic expression via third-party components that supports this part of the MathML standard.
private java.util.ArrayList	**getUserParameterNames**() Helper method that returns the user configurable parameters from the content-MathML of the object.
java.lang.String	**getUserParametersXML**() Returns an XML of the user configurable parameters from the content-MathML of the object.
java.lang.String	**getXML**(boolean includeEncodingInstruction) Gets a XML-representation of the ContentDocument as a String.
void	**setContentMathId**(long contentMathId) Sets the contentMathId by setting the objectId.

Field Detail

logCat

```
private static org.apache.log4j.Category logCat
```

The logger for writing messages to the log, provides central control on the level of logging as well as display of the originator of the log entry.

Constructor Detail

ContentMath

```
public ContentMath()
```

Standard ctor which creates a new ContentMath-object without any initial data.

Method Detail

getContentMathId

```
public long getContentMathId()
```

Gets the contentMathId by getting the objectId.

---

setContentMathId

```
public void setContentMathId(long contentMathId)
```

Sets the contentMathId by setting the objectId.

---

getXML

```
public java.lang.String getXML(boolean includeEncodingInstruction)
```

Gets a XML-representation of the ContentDocument as a String.

Overrides:

`getXML` in class `no.inspera.applications.publish.datamodel.ContentRevision`

Parameters:

`includeEncodingInstruction` - If `true`, include XML processing instruction setting encoding to ISO-8859-1.

---

getDocument

```
public org.apache.lucene.document.Document getDocument()
```

Implementation to facilitate indexing (and thereby searching) via Jakarta Lucene.

Specified by:

`getDocument` in interface `no.inspera.services.search.Searchable`

Overrides:

`getDocument` in class `no.inspera.applications.publish.datamodel.ContentRevision`

---

getPresentationMathML

```
public java.lang.String getPresentationMathML()
```

Gets the math expression in the form of presentation-based MathML, used for editing and creating images of the symbolic expression via third-party components that supports this part of the MathML standard.

---

getContentMathML

```
private java.lang.String getContentMathML()
```

Gets the content-MathML from this mathematical object. This contains the semantics (i.e. the meaning) of the mathematical expression.

---

getPlainTextMath

```
public java.lang.String getPlainTextMath()
```

Gets a plaintext-representation of the mathematical expression, suitable for the plaintextparser used to compute the values. Does this by traversing the content MathML tree.

---

getPlainTextMathFromSubTree

```
private java.lang.String getPlainTextMathFromSubTree(org.w3c.dom.Element e)
```

Helper method that gets a plaintext-representation of the mathematical expression subtree starting with `e`. Does this by traversing the content MathML tree which `e` should be a node in.

Parameters:

`e` - The node to start from.

---

getElementsFromList

```
private java.util.ArrayList getElementsFromList(org.w3c.dom.NodeList l)
```

Helper method that returns only the `Element`-type `Node`s from the given list.

---

getUserParametersXML

```
public java.lang.String getUserParametersXML()
```

Returns an XML of the user configurable parameters from the content-MathML of the object. This is decided to be all identificators that are not y, x or t.

---

getUserParameterNames

```
private java.util.ArrayList getUserParameterNames()
```

Helper method that returns the user configurable parameters from the content-MathML of the object. This is decided to be all identificators that are not y, x or t. Will never return null, an empty list is returned if now such parameters exist.

no.inspera.applications.publish.math

Class PointValue

```
java.lang.Object
  |
  +--no.inspera.applications.publish.math.PointValue
```

public class **PointValue**
extends java.lang.Object

This class represents a simple point value with decimal numbers.

Currently it is used solely for (x,y) (2-dimensional values), but extending it for multidimensional (including 3D) should be fairly easy.

The class is used both by the server side part and by the graph drawer applet.

Version:
$Revision: 1.4 $ $Date: 2003/05/18 20:21:48 $

Field Summary

double	**x** The x-value of the point, made public on intention to reduce unecessary function calls (it will be MANY such objects for a graph drawing, for instance).
double	**y** The y-value of the point, made public on intention to reduce unecessary function calls (it will be MANY such objects for a graph drawing, for instance).

Constructor Summary

PointValue() Standard constructor for creating a object with no values set.
PointValue(double x, double y) Parameterized constructor for creating a object with values already set.

Method Summary

boolean	**equals**(java.lang.Object o) Override Object.equals() to provide specific behaviour, i.e. to be equal all values must be equal.
double	**getDimensionValue**(int dim) Method for generic access to the Nth-dimension value of the point.
java.lang.String	**toString**() String representation of the object, overrides Object.toString() for more useful debugging info.

Field Detail

x

```
public double x
```

The x-value of the point, made public on intention to reduce unecessary function calls (it will be MANY such objects for a graph drawing, for instance).

---

y

```
public double y
```

The y-value of the point, made public on intention to reduce unecessary function calls (it will be MANY such objects for a graph drawing, for instance).

Constructor Detail

PointValue

```
public PointValue()
```

Standard constructor for creating a object with no values set.

PointValue

```
public PointValue(double x,
                  double y)
```

Parameterized constructor for creating a object with values already set.

Method Detail

getDimensionValue

```
public double getDimensionValue(int dim)
```

Method for generic access to the Nth-dimension value of the point. For now this is just a way of "futureproofing" some of the code, since loops etc. of the callers can be written a bit more general. For now only dimensions 1 (x) and 2 (y) is used.

Parameters:

`dim` - The natural number of the dimension to get, i.e. a 1-based index.

---

equals

```
public boolean equals(java.lang.Object o)
```

Override Object.equals() to provide specific behaviour, i.e. to be equal all values must be equal.

Overrides:

`equals` in class `java.lang.Object`

---

toString

```
public java.lang.String toString()
```

String representation of the object, overrides Object.toString() for more useful debugging info.

Overrides: `toString` in class `java.lang.Object`

no.inspera.applications.publish.math

Interface PointValueProducer

All Known Implementing Classes:

PlaintextFilePointValueProducer, ContentMathPointValueProducer

---

public interface **PointValueProducer**

This interface defines a producer of PointValues, i.e. a source of data for display (for instance for a graph).

This way different sources such as statistical data and mathematical functions can implement different kinds of PVPs and thereby use the same drawing functions via a clean API.

The class is used both by the server side part and by the graph drawer applet.

Version:

$Revision: 1.6 $ $Date: 2003/05/18 20:21:48 $

Method Summary

GraphSettings	**getGraphSettings**() Gets the settings for the graph associated with this PVP.
double	**getMaxValueForDimension**(int dimNumber) Gets the maximum value of all the points for the given dimension this producer will deliver with the current settings.
double	**getMinValueForDimension**(int dimNumber) Gets the minimum value of all the points for the given dimension this producer will deliver with the current settings.
PointValue	**getNextPointValue**() Gets the next point value from the underlying producer.
int	**getNumberOfValues**() Gets the number of values this PVP will provide.
PointValue	**getPointValueForXVal**(double xVal, double precision) Gets the point value for a point by the given x-value (to the precision given by `precision`, implementing classes should terminate the search when the x-value is within `precision` from `xVal`).
PointValue[]	**getPointValuesInMaximum**() Gets the point values in local maximum values (where y have a top) for the intervals currently computed.
PointValue[]	**getPointValuesInMinimum**() Gets the point values in local minimum values (where y have a bottom) for the intervals currently computed.
PointValue	**getValueNumber**(int index) Gets pointvalue number `index` (at offset index, zero-based).

void	**markDirty**() Marks the values as dirty, i.e. they must be re-computed before getNextPointValue() can be called.
void	**reset**() Resets the "iterator" of the PVP so that the next call to getNextPointValue() will give the first value
void	**setGraphSettings**(GraphSettings graphSettings) Sets the settings for the graph associated with this PVP.

Method Detail

getNextPointValue

```
public PointValue getNextPointValue()
```

Gets the next point value from the underlying producer. Should return `null` if no more values are available, this way we can avoid a separate call to hasNext() or similar (which might be significant as it will probably be exessive calls to the method).

---

getMaxValueForDimension

```
public double getMaxValueForDimension(int dimNumber)
                                 throws java.lang.IllegalArgumentException
```

Gets the maximum value of all the points for the given dimension this producer will deliver with the current settings. If the max value is not available, an IllegalArgumentException is thrown.

---

getMinValueForDimension

```
public double getMinValueForDimension(int dimNumber)
                                 throws java.lang.IllegalArgumentException
```

Gets the minimum value of all the points for the given dimension this producer will deliver with the current settings. If the min value is not available, an IllegalArgumentException is thrown.

---

getGraphSettings

```
public GraphSettings getGraphSettings()
```

Gets the settings for the graph associated with this PVP.

---

setGraphSettings

```
public void setGraphSettings(GraphSettings graphSettings)
```

Sets the settings for the graph associated with this PVP.

---

reset

```
public void reset()
```

Resets the "iterator" of the PVP so that the next call to getNextPointValue() will give the first value

---

markDirty

```
public void markDirty()
```

Marks the values as dirty, i.e. they must be re-computed before getNextPointValue() can be called.

---

getNumberOfValues

```
public int getNumberOfValues()
```

Gets the number of values this PVP will provide.

---

getValueNumber

```
public PointValue getValueNumber(int index)
```

Gets pointvalue number `index` (at offset index, zero-based).

---

getPointValuesInMaximum

```
public PointValue[] getPointValuesInMaximum()
```

Gets the point values in local maximum values (where y have a top) for the intervals currently computed. Should never return null (but an array with length 0 if no maximum exists).

---

getPointValuesInMinimum

```
public PointValue[] getPointValuesInMinimum()
```

Gets the point values in local minimum values (where y have a bottom) for the intervals currently computed. Should never return null (but an array with length 0 if no minimum exists).

---

getPointValueForXVal

```
public PointValue getPointValueForXVal(double xVal,
                                       double precision)
```

Gets the point value for a point by the given x-value (to the precision given by `precision`, implementing classes should terminate the search when the x-value is within `precision` from `xVal`). Should return null if the PVP is not defined for this value.

no.inspera.applications.publish.math

Class ContentMathPointValueProducer

```
java.lang.Object
  |
  +--no.inspera.applications.publish.math.ContentMathPointValueProducer
```

All Implemented Interfaces:

PointValueProducer

---

public class **ContentMathPointValueProducer**
extends java.lang.Object
implements PointValueProducer

This is a implementation for a producer of PointValues, i.e. a source of data for display (for instance for a graph) based on a plaintext math object. I.e. it will use a plaintext-version of a ContentMath-object from the DB.

The strategy used by this class is to first compute all values in the given dimension and then return values from the cached pointvalues as long as possible to avoid recomputing values at each call.

The class is used both by the server side part and by the graph drawer applet.

Each caller will use a separate object for getting values and the getNextPointValue is not thread safe (in the sense that the "current index" is shared among threads). But this will no be a problem as long as each request/thread uses its own CMPVP.

Version:

$Revision: 1.8 $ $Date: 2003/05/18 20:21:48 $

Field Summary

private int	**currentIndex** The current index of the values to return from getNextPointValue.
private GraphSettings	**graphSettings** Some of the seettings are relevant for the computation, such as intervals etc.
private boolean	**needsComputing** Indicator to show if a new computing if needed.
private java.util.Vector	**values** The computed values to use for drawing.

Constructor Summary

(package private)	**ContentMathPointValueProducer**(GraphSettings gs) Creates the PointValueProducer based on the given settings.

Method Summary	
private void	**doComputation**() Does the actual computation of point values.
GraphSettings	**getGraphSettings**() Gets the settings for the graph associated with this PVP.
double	**getMaxValueForDimension**(int dimNumber) Gets the max value of all the points for the given dimension this producer will deliver with the current settings.
double	**getMinValueForDimension**(int dimNumber) Gets the min value of all the points for the given dimension this producer will deliver with the current settings.
PointValue	**getNextPointValue**() Gets the next point value from the underlying producer.
int	**getNumberOfValues**() Gets the number of values this PVP will provide.
PointValue	**getPointValueForXVal**(double xVal, double precision) Gets the point value for a point by the given x-value (to the precision given by precision, will terminate the search when the x-value is within precision from xVal).
PointValue[]	**getPointValuesInMaximum**() Gets the point values in local maximum values (where y have a top) for the intervals currently computed.
PointValue[]	**getPointValuesInMinimum**() Gets the point values in local minimum values (where y have a bottom) for the intervals currently computed.
PointValue	**getValueNumber**(int index) Gets pointvalue number index (at offset index, zero-based).
void	**markDirty**() Marks the values as dirty, i.e. they must be re-computed before getNextPointValue() can be called.
void	**reset**() Resets the "iterator" of the PVP so that the next call to getNextPointValue() will give the first value
void	**setGraphSettings**(GraphSettings graphSettings) Sets the settings for the graph associated with this PVP.
private java.lang.String	**substituteUserParams**(java.lang.String s) Helper method that substitutes any user parameters in the plaintext expression before computing.

Field Detail

needsComputing

```
private boolean needsComputing
```

Indicator to show if a new computing if needed.

---

values

```
private java.util.Vector values
```

The computed values to use for drawing. A Vector of PointValue-objects. Since this class is intended to be used in an applet as well, the use of the "old" Collection-API is intended.

---

graphSettings

```
private GraphSettings graphSettings
```

Some of the seettings are relevant for the computation, such as intervals etc.

---

currentIndex

```
private int currentIndex
```

The current index of the values to return from getNextPointValue.

Constructor Detail

ContentMathPointValueProducer

```
ContentMathPointValueProducer(GraphSettings gs)
```

Creates the PointValueProducer based on the given settings.

Method Detail

getNextPointValue

```
public PointValue getNextPointValue()
```

Gets the next point value from the underlying producer. Should return `null` if no more values are available, this way we can avoid a separate call to hasNext() or similar (which might be significant as it will probably be exessive calls to the method).

Specified by:

`getNextPointValue` in interface `PointValueProducer`

---

reset

```
public void reset()
```

Resets the "iterator" of the PVP so that the next call to getNextPointValue() will give the first value

Specified by:

`reset` in interface `PointValueProducer`

---

doComputation

```
private void doComputation()
```

Does the actual computation of point values.

---

getMaxValueForDimension

```
public double getMaxValueForDimension(int dimNumber)
                                 throws java.lang.IllegalArgumentException
```

Gets the max value of all the points for the given dimension this producer will deliver with the current settings. If the max value is not available for the given dimension, an IllegalArgumentException is thrown.

Specified by:

`getMaxValueForDimension` in interface `PointValueProducer`

---

getMinValueForDimension

```
public double getMinValueForDimension(int dimNumber)
                                 throws java.lang.IllegalArgumentException
```

Gets the min value of all the points for the given dimension this producer will deliver with the current settings. If the min value is not available for the given dimension, an IllegalArgumentException is thrown.

Specified by:

`getMinValueForDimension` in interface `PointValueProducer`

---

getGraphSettings

```
public GraphSettings getGraphSettings()
```

Gets the settings for the graph associated with this PVP.

Specified by:

`getGraphSettings` in interface `PointValueProducer`

---

setGraphSettings

```
public void setGraphSettings(GraphSettings graphSettings)
```

Sets the settings for the graph associated with this PVP.

Specified by:

`setGraphSettings` in interface `PointValueProducer`

---

markDirty

```
public void markDirty()
```

Marks the values as dirty, i.e. they must be re-computed before getNextPointValue() can be called.

Specified by:

`markDirty` in interface `PointValueProducer`

---

getNumberOfValues

```
public int getNumberOfValues()
```

Gets the number of values this PVP will provide.

Specified by:

`getNumberOfValues` in interface `PointValueProducer`

---

getValueNumber

```
public PointValue getValueNumber(int index)
```

Gets pointvalue number `index` (at offset index, zero-based).

Specified by:

`getValueNumber` in interface `PointValueProducer`

---

getPointValuesInMaximum

```
public PointValue[] getPointValuesInMaximum()
```

Gets the point values in local maximum values (where y have a top) for the intervals currently computed. Will never return null (but an array with length 0 if no maximum exists).

Specified by:

`getPointValuesInMaximum` in interface `PointValueProducer`

---

getPointValuesInMinimum

```
public PointValue[] getPointValuesInMinimum()
```

Gets the point values in local minimum values (where y have a bottom) for the intervals currently computed. Will never return null (but an array with length 0 if no minimum exists).

Specified by:

`getPointValuesInMinimum` in interface `PointValueProducer`

---

getPointValueForXVal

```
public PointValue getPointValueForXVal(double xVal,
                                       double precision)
```

Gets the point value for a point by the given x-value (to the precision given by `precision`, will terminate the search when the x-value is within `precision` from `xVal`). Returns null if the PVP is not defined for this value. The logic uses binary search.

Specified by:

`getPointValueForXVal` in interface `PointValueProducer`

---

substituteUserParams

```
private java.lang.String substituteUserParams(java.lang.String s)
```

Helper method that substitutes any user parameters in the plaintext expression before computing.

no.inspera.applications.publish.math

Class PointValueProducerFactory

```
java.lang.Object
  |
  +--no.inspera.applications.publish.math.PointValueProducerFactory
```

public class **PointValueProducerFactory**
extends java.lang.Object

This class provides a way to get a instance of a PointValueProducer, i.e. get a source of data for display (for instance for a graph).

By using a PointValueProducer different sources (such as statistical data and mathematical functions) can use the same drawing functions via a clean API.

The class is used both by the server side part and by the graph drawer applet.

Version:
$Revision: 1.4 $ $Date: 2003/05/18 20:21:48 $

Constructor Summary

private	**PointValueProducerFactory**() "Hides" the constructor (the default implementation if not provided is public), as it should never be used -- only the static factory method is of any relevance to callers.

Method Summary

static PointValueProducer	**getPointValueProducer**(GraphSettings gds) Gets a PointValueProducer-instance that will provide PointValues to the caller.

no.inspera.applications.publish.math

Class GraphDrawer

```
java.lang.Object
  |
  +--no.inspera.applications.publish.math.GraphDrawer
```

Direct Known Subclasses:

GraphDrawerLine, GraphDrawerPlot

---

public abstract class **GraphDrawer**
extends java.lang.Object

This class provides the interface that has to be implemented by the classes doing the actual implementation of the drawing of graphs.

It will do this by drawing on a `Graphics`-object using a PointValueProducer which will provide the values to draw.

Subclasses of this class will provide drawing of the differents types of graphs, such as line or bar charts. This class provides a factory-method to get an instance of such a subclass and common functionality among the different graph types might also be implemented here.

The class is used both by the server side part and by the graph drawer applet.

Version:

$Revision: 1.7 $ $Date: 2003/05/15 20:22:04 $

Field Summary

protected GraphDrawerSettings	**graphDrawerSettings** The settings for this graph drawer, such as background color etc.
protected PointValueProducer[]	**pointValueProducers** The source for values for this GraphDrawer.

Method Summary

abstract void	**drawGraph**(java.awt.Graphics g, int topLeftX, int topLeftY, int bottomRightX, int bottomRightY) Draws the graph to the given graphics-object, the actual drawing is required to be implemented by subclasses.
abstract void	**drawGraphToPoint**(java.awt.Graphics g, int topLeftX, int topLeftY, int bottomRightX, int bottomRightY, PointValue stopPoint) Draws the graph up to the value given to the given graphics-object, the actual drawing is required to be implemented by subclasses.

abstract void	**drawImageCenteredAtPoint**(java.awt.Graphics g, java.awt.Image image, PointValue pv, java.awt.image.ImageObserver imgObs, int topLeftX, int topLeftY, int bottomRightX, int bottomRightY) Draws the Image-object image at the location indicated by the real world values in pv.
abstract void	**drawMarkerAtPoint**(java.awt.Graphics g, PointValue pv, int topLeftX, int topLeftY, int bottomRightX, int bottomRightY) Draws a marker (cross) at the location indicated by the real world values in pv.
GraphDrawerSettings	**getGraphDrawerSettings**() Gets the settings for this graph drawer.
static GraphDrawer	**getInstance**(PointValueProducer[] pvps, GraphDrawerSettings gds) Factory-method for getting an instance of a subclass of GraphDrawer.
static GraphDrawer	**getInstance**(PointValueProducer pvp, GraphDrawerSettings gds) Factory-method for getting an instance of a subclass of GraphDrawer.
abstract double	**getMaxX**() Gets the maximum value in the x-direction, from all the PVPs used to draw a graph.
abstract double	**getMaxY**() Gets the maximum value in the y-direction, from all the PVPs used to draw a graph.
abstract double	**getMinX**() Gets the minimum value in the x-direction, from all the PVPs used to draw a graph.
abstract double	**getMinY**() Gets the minimum value in the y-direction, from all the PVPs used to draw a graph.
PointValueProducer[]	**getPointValueProducers**() Gets the source for values for this GraphDrawer.
abstract void	**resetExtremalValues**() Resets max/min values in all (both for now!)
void	**setGraphDrawerSettings**(GraphDrawerSettings graphDrawerSettings) Sets the settings for this graph drawer.
void	**setPointValueProducers**(PointValueProducer[] pointValueProducers) Sets the source for values for this GraphDrawer.

Field Detail

pointValueProducers

```
protected PointValueProducer[] pointValueProducers
```

The source for values for this GraphDrawer.

---

graphDrawerSettings

```
protected GraphDrawerSettings graphDrawerSettings
```

The settings for this graph drawer, such as background color etc.

Method Detail

drawGraph

```
public abstract void drawGraph(java.awt.Graphics g,
                               int topLeftX,
                               int topLeftY,
                               int bottomRightX,
                               int bottomRightY)
```

Draws the graph to the given graphics-object, the actual drawing is required to be implemented by subclasses. If yInterval of the associated settings-object is null, the yInterval is computed from min/max values if needed for that type of graph (i.e. one does not have to send yInterval if one does not want to override this default).

---

drawGraphToPoint

```
public abstract void drawGraphToPoint(java.awt.Graphics g,
                                      int topLeftX,
                                      int topLeftY,
                                      int bottomRightX,
                                      int bottomRightY,
                                      PointValue stopPoint)
```

Draws the graph up to the value given to the given graphics-object, the actual drawing is required to be implemented by subclasses. If yInterval of the associated settings-object is null, the yInterval is computed from min/max values if needed for that type of graph (i.e. one does not have to send yInterval if one does not want to override this default).

---

getMaxX

```
public abstract double getMaxX()
```

Gets the maximum value in the x-direction, from all the PVPs used to draw a graph.

---

getMinX

```
public abstract double getMinX()
```

Gets the minimum value in the x-direction, from all the PVPs used to draw a graph.

---

getMaxY

```
public abstract double getMaxY()
```

Gets the maximum value in the y-direction, from all the PVPs used to draw a graph.

---

getMinY

```
public abstract double getMinY()
```

Gets the minimum value in the y-direction, from all the PVPs used to draw a graph.

---

resetExtremalValues

```
public abstract void resetExtremalValues()
```

Resets max/min values in all (both for now!) dimensions. Can also be called from outside after zooming or other changes of the underlying PointValueProucers, made public for that reason.

---

drawImageCenteredAtPoint

```
public abstract void drawImageCenteredAtPoint(java.awt.Graphics g,
                                              java.awt.Image image,
                                              PointValue pv,

java.awt.image.ImageObserver imgObs,
                                              int topLeftX,
                                              int topLeftY,
                                              int bottomRightX,
                                              int bottomRightY)
```

Draws the Image-object `image at the location indicated by the real world values in pv. The image will be places so its center is in that location and drawn on the Graphics-object g. The four last parameters gives the rectangle of the Graphics object to use.`

---

drawMarkerAtPoint

```
public abstract void drawMarkerAtPoint(java.awt.Graphics g,
                                       PointValue pv,
                                       int topLeftX,
                                       int topLeftY,
                                       int bottomRightX,
                                       int bottomRightY)
```

`Draws a marker (cross) at the location indicated by the real world values in pv. A String indicating the value pair will also be drawn on the Graphics-object g.`

---

getInstance

```
public static GraphDrawer getInstance(PointValueProducer pvp,
                                      GraphDrawerSettings gds)
```

`Factory-method for getting an instance of a subclass of GraphDrawer. This is just a convenience-method to avoid having callers make arrays.`

---

getInstance

```
public static GraphDrawer getInstance(PointValueProducer[] pvps,
                                      GraphDrawerSettings gds)
```

Factory-method for getting an instance of a subclass of GraphDrawer. The array of PVPs (which may contain only one element, for support of this see the convenience-method with approriate type) will give the values for the drawing.

---

getPointValueProducers

```
public PointValueProducer[] getPointValueProducers()
```

Gets the source for values for this GraphDrawer.

---

setPointValueProducers

```
public void
setPointValueProducers(PointValueProducer[] pointValueProducers)
```

Sets the source for values for this GraphDrawer.

---

getGraphDrawerSettings

```
public GraphDrawerSettings getGraphDrawerSettings()
```

Gets the settings for this graph drawer.

---

setGraphDrawerSettings

```
public void setGraphDrawerSettings(GraphDrawerSettings graphDrawerSettings)
```

Sets the settings for this graph drawer.

no.inspera.applications.publish.math

Class GraphDrawerPlot

```
java.lang.Object
  |
  +--no.inspera.applications.publish.math.GraphDrawer
        |
        +--no.inspera.applications.publish.math.GraphDrawerPlot
```

public class **GraphDrawerPlot**
extends GraphDrawer

This class provides the implementation of GraphDrawer for plots of graphs (typically mathematical functions).

It will do this by drawing on a `Graphics`-object using a PointValueProducer which will provide the values to draw.

The class is used both by the server side part and by the graph drawer applet.

Version:
$Revision: 1.3 $ $Date: 2003/05/18 20:21:48 $

Field Summary

private int	**AXIS_ARROWHEAD_HEIGHT** A constant that decides the height of the arrowhead on the axis (in pixels).
private int	**AXIS_ARROWHEAD_WIDTH** A constant that decides the width of the arrowhead on the axis (in pixels).
private int	**AXIS_SCALEINDICATOR_LENGTH** A constant that decides the length of scale indicator lines.
private int	**MARKER_LINELENGTH** A constant that decides the length of the diagonals of the marker used to mark a point.
private double	**maxX** The maximum x-value to draw (computed), in real-world values
private double	**maxY** The maximum y-value to draw (computed), in real-world values
private double	**minX** The minimum x-value to draw (computed), in real-world values
private double	**minY** The minimum y-value to draw (computed), in real-world values

private int	**PADDING POINTVALUE** A constant that decides the size of the space from the marker to the text with the value.
private int	**PADDING SCALEINDICATOR** A constant that decides the size of the space around the scale indicators.
private java.text.DecimalFormat	**pointTextFormater** A formatter used to format the values for display next to a marker for a point.
private java.awt.Font	**scaleFont** The font used for printing the values for the scale indicators.
private java.awt.Font	**valueFont** The font used for printing the values for points next to a marker/cross.

Method Summary

private void	**d**(java.lang.String s) Used for writing a debug-string to the console.
private void	**drawAxisAndScale**(java.awt.Graphics g, double xScaleFactor, double yScaleFactor, int topLeftX, int bottomRightY, java.text.DecimalFormat scaleFormater) Helper method that draws the axis and (if set) the scale indicators to the given Graphics-object.
void	**drawGraph**(java.awt.Graphics g, int topLeftX, int topLeftY, int bottomRightX, int bottomRightY) Draws the graph up to the value number given to the given graphics-object, the actual drawing is required to be implemented by subclasses.
void	**drawGraphToPoint**(java.awt.Graphics g, int topLeftX, int topLeftY, int bottomRightX, int bottomRightY, PointValue stopPoint) Draws the graph to the given graphics-object, by plotting a great number of points.
private void	**drawHorizCenteredString**(java.lang.String s, double x, double y, double xScaleFactor, double yScaleFactor, int offsetX, int maxOffsetY, java.awt.Graphics g) Helper method that will draw a string centered horizontally to the point (real world values that will be translated), and the top of the text placed to the point.
void	**drawImageCenteredAtPoint**(java.awt.Graphics g, java.awt.Image image, PointValue pv, java.awt.image.ImageObserver imgObs, int topLeftX, int topLeftY, int bottomRightX, int bottomRightY) Draws the Image-object image at the location indicated by the real world values in pv.
void	**drawMarkerAtPoint**(java.awt.Graphics g, PointValue pv, int topLeftX, int topLeftY, int bottomRightX, int bottomRightY) Draws a marker (cross) at the location indicated by the real world values in pv.

private void	**drawTranslatedLine**(double xFrom, double yFrom, double xTo, double yTo, double xScaleFactor, double yScaleFactor, int offsetX, int maxOffsetY, java.awt.Graphics g) Helper method that will draw a line to the given graphics object.
private void	**drawVertCenteredString**(java.lang.String s, double x, double y, double xScaleFactor, double yScaleFactor, int offsetX, int maxOffsetY, java.awt.Graphics g) Helper method that will draw a string centered vertically to the point (real world values that will be translated), and the top of the text placed to the point.
double	**getMaxX**() Gets the maximum value in the x-direction, from all the PVPs used to draw a graph.
double	**getMaxY**() Gets the maximum value in the y-direction, from all the PVPs used to draw a graph.
double	**getMinX**() Gets the minimum value in the x-direction, from all the PVPs used to draw a graph.
double	**getMinY**() Gets the minimum value in the y-direction, from all the PVPs used to draw a graph.
private void	**plotTranslatedPoint**(PointValue p, double xScaleFactor, double yScaleFactor, int offsetX, int maxOffsetY, java.awt.Graphics g) Helper method that will plot a point to the given graphics object.
void	**resetExtremalValues**() Helper method that reset max/min values in all (both for now!)

Field Detail

maxX

```
private double maxX
```

The maximum x-value to draw (computed), in real-world values

---

maxY

```
private double maxY
```

The maximum y-value to draw (computed), in real-world values

---

minX

```
private double minX
```

The minimum x-value to draw (computed), in real-world values

---

minY

```
private double minY
```

The minimum y-value to draw (computed), in real-world values

AXIS_ARROWHEAD_WIDTH

```
private final int AXIS_ARROWHEAD_WIDTH
```

A constant that decides the width of the arrowhead on the axis (in pixels).

AXIS_ARROWHEAD_HEIGHT

```
private final int AXIS_ARROWHEAD_HEIGHT
```

A constant that decides the height of the arrowhead on the axis (in pixels).

AXIS_SCALEINDICATOR_LENGTH

```
private final int AXIS_SCALEINDICATOR_LENGTH
```

A constant that decides the length of scale indicator lines.

PADDING_SCALEINDICATOR

```
private final int PADDING_SCALEINDICATOR
```

A constant that decides the size of the space around the scale indicators.

---

scaleFont

```
private final java.awt.Font scaleFont
```

The font used for printing the values for the scale indicators.

MARKER_LINELENGTH

```
private final int MARKER_LINELENGTH
```

A constant that decides the length of the diagonals of the marker used to mark a point.

---

pointTextFormater

```
private final java.text.DecimalFormat pointTextFormater
```

A formatter used to format the values for display next to a marker for a point.

PADDING_POINTVALUE

```
private final int PADDING_POINTVALUE
```

A constant that decides the size of the space from the marker to the text with the value.

---

valueFont

```
private final java.awt.Font valueFont
```

The font used for printing the values for points next to a marker/cross.

Method Detail

resetExtremalValues

```
public void resetExtremalValues()
```

Helper method that reset max/min values in all (both for now!) dimensions. Can also be called from outside after zooming or other changes of the underlying PointValueProucers, made public for that reason.

Overrides:

`resetExtremalValues` in class `GraphDrawer`

---

drawGraph

```
public void drawGraph(java.awt.Graphics g,
                      int topLeftX,
                      int topLeftY,
                      int bottomRightX,
                      int bottomRightY)
```

Draws the graph up to the value number given to the given graphics-object, the actual drawing is required to be implemented by subclasses. If yInterval of the associated settings-object is null, the yInterval is computed from min/max values if needed for that type of graph (i.e. one does not have to send yInterval if one does not want to override this default).

Overrides:

`drawGraph` in class `GraphDrawer`

---

drawGraphToPoint

```
public void drawGraphToPoint(java.awt.Graphics g,
                             int topLeftX,
                             int topLeftY,
                             int bottomRightX,
                             int bottomRightY,
                             PointValue stopPoint)
```

Draws the graph to the given graphics-object, by plotting a great number of points. If yInterval is null, the yInterval is computed from min/max values (i.e. one does not have to send yInterval if one does not want to override this default).

Overrides:

`drawGraphToPoint` in class `GraphDrawer`

---

drawAxisAndScale

```
private void drawAxisAndScale(java.awt.Graphics g,
                              double xScaleFactor,
                              double yScaleFactor,
                              int topLeftX,
                              int bottomRightY,
                              java.text.DecimalFormat scaleFormater)
```

Helper method that draws the axis and (if set) the scale indicators to the given Graphics-object.

---

drawHorizCenteredString

```
private void drawHorizCenteredString(java.lang.String s,
                                     double x,
                                     double y,
                                     double xScaleFactor,
                                     double yScaleFactor,
                                     int offsetX,
                                     int maxOffsetY,
                                     java.awt.Graphics g)
```

Helper method that will draw a string centered horizontally to the point (real world values that will be translated), and the top of the text placed to the point. Will draw to the given graphics object. The font will be reset to the previous font after drawing.

---

drawVertCenteredString

```
private void drawVertCenteredString(java.lang.String s,
                                    double x,
                                    double y,
                                    double xScaleFactor,
                                    double yScaleFactor,
                                    int offsetX,
                                    int maxOffsetY,
                                    java.awt.Graphics g)
```

Helper method that will draw a string centered vertically to the point (real world values that will be translated), and the top of the text placed to the point. Will draw to the given graphics object. The font will be reset to the previous font after drawing.

---

drawTranslatedLine

```
private void drawTranslatedLine(double xFrom,
                                double yFrom,
                                double xTo,
                                double yTo,
                                double xScaleFactor,
                                double yScaleFactor,
                                int offsetX,
                                int maxOffsetY,
                                java.awt.Graphics g)
```

Helper method that will draw a line to the given graphics object. The four coordinate-values will have values in real-world values, this method translates this to screen coordinates before drawing.

---

plotTranslatedPoint

```
private void plotTranslatedPoint(PointValue p,
                                 double xScaleFactor,
                                 double yScaleFactor,
                                 int offsetX,
                                 int maxOffsetY,
                                 java.awt.Graphics g)
```

Helper method that will plot a point to the given graphics object. A point is plotted in java by using a line ending in the same point that it is starting. The point `p` will have values in real-world values, this method translates this to screen coordinates before drawing.

---

drawImageCenteredAtPoint

```
public void drawImageCenteredAtPoint(java.awt.Graphics g,
                                     java.awt.Image image,
                                     PointValue pv,
                                     java.awt.image.ImageObserver imgObs,
                                     int topLeftX,
                                     int topLeftY,
                                     int bottomRightX,
                                     int bottomRightY)
```

Draws the Image-object image at the location indicated by the real world values in pv. The image will be places so its center is in that location and drawn on the Graphics-object g.

Overrides:

drawImageCenteredAtPoint in class GraphDrawer

---

drawMarkerAtPoint

```
public void drawMarkerAtPoint(java.awt.Graphics g,
                              PointValue pv,
                              int topLeftX,
                              int topLeftY,
                              int bottomRightX,
                              int bottomRightY)
```

Draws a marker (cross) at the location indicated by the real world values in pv. A String indicating the value pair will also be drawn on the Graphics-object g.

Overrides:

drawMarkerAtPoint in class GraphDrawer

---

getMaxX

```
public double getMaxX()
```

Gets the maximum value in the x-direction, from all the PVPs used to draw a graph.

Overrides:

getMaxX in class GraphDrawer

---

getMaxY

```
public double getMaxY()
```

Gets the maximum value in the y-direction, from all the PVPs used to draw a graph.

Overrides:

getMaxY in class GraphDrawer

---

getMinX

```
public double getMinX()
```

Gets the minimum value in the x-direction, from all the PVPs used to draw a graph.

Overrides:

getMinX in class GraphDrawer

---

getMinY

```
public double getMinY()
```

Gets the minimum value in the y-direction, from all the PVPs used to draw a graph.

Overrides:

getMinY in class GraphDrawer

---

d

```
private void d(java.lang.String s)
```

Used for writing a debug-string to the console. Since this class is used both by the server side and the by the applet we cannot simply use the standard log4j-logging on the server side.

no.inspera.applications.publish.math

Class GraphDrawerSettings

```
java.lang.Object
  |
  +--no.inspera.applications.publish.math.GraphDrawerSettings
```

public class **GraphDrawerSettings**
extends java.lang.Object

This class is an collection of settings for the graph drawer, such as the background color, to include axis or not, and so on. The class GraphSettings contains the settings for one particular graph, so it can be many GraphSettings-objects for one GraphDrawerSettings-object. The class is used both by the server side part and by the graph drawer applet.

Version:
$Revision: 1.7 $ $Date: 2003/05/18 20:21:48 $

Field Summary

private java.awt.Color	**backgroundColor** The color used as a background for the drawing-area for the graph.
private static int	**DEFAULT_NUMBER_OF_STEPS** A constant that gives the default number of steps used for computing pointvalues for a graph.
private boolean	**drawAxis** If `true`, a GraphDrawer based on these settings will draw the axis.
private GraphSettings[]	**graphSettings** Settings for the indvidual graphs drawed using the general settings in this object.
private double	**scaleIndicatorSizeX** The size of (distance between) scale indicators to use for the x-axis, if applicable for the type of graph object these settings regard.
private double	**scaleIndicatorSizeY** The size of (distance between) scale indicators to use for the y-axis, if applicable for the type of graph object these settings regard.
private java.util.Hashtable	**userParameters** The parameters that can be set by the user, in the form name => value (i.e. the names of the variables are used as keys when hashing).
private MathInterval	**xInterval** The interval (first dimension values) these settings applies to (in real world values), i.e. this and yInterval defines the area to draw.

private MathInterval	**yInterval** The interval (second dimension values) these settings applies to (in real world values), i.e. this and xInterval defines the area to draw.

Constructor Summary

GraphDrawerSettings() Creates empty settings object with default settings.

Method Summary

void	**adjustStepSize**() Adjusts the stepsize, using the default number of steps, so that the graphs will get a correct number of samples according to the size of their definition interval.
void	**adjustStepSize**(int numberOfSteps) Adjusts the stepsize so that the graphs will get a correct number of samples according to the size of their definition interval.
java.awt.Color	**getBackgroundColor**() Gets the color used as a background for the drawing-area for the graph.
boolean	**getDrawAxis**() Gets the axis on/off parameter.
GraphSettings[]	**getGraphSettings**() Gets the settings for the indvidual graphs drawed using the general settings in this object.
GraphSettings	**getGraphSettingsAtIndex**(int index) Gets the graph setting object with the given (0-based) index, if it exists.
int	**getNumberOfGraphs**() Returns the number of graphs these settings contain, i.e. how many GraphSettings-objects that are present
double	**getScaleIndicatorSizeX**() Gets the size of (distance between) scale indicators to use for the x-axis, if applicable for the type of graph object these settings regard.
double	**getScaleIndicatorSizeY**() Gets the size of (distance between) scale indicators to use for the y-axis, if applicable for the type of graph object these settings regard.

java.util.Hashtable	**getUserParameters**() Gets the parameters that can be set by the user, in the form name => value.
MathInterval	**getXInterval**() Gets the interval (first dimension values) these settings applies to (in real world values), i.e. this and yInterval defines the area to draw.
MathInterval	**getYInterval**() Sets the interval (second dimension values) these settings applies to (in real world values), i.e. this and xInterval defines the area to draw.
boolean	**hasGraphs**() Returns true if this object includes settings for any graphs as well.
void	**setBackgroundColor**(java.awt.Color backgroundColor) Sets the color used as a background for the drawing-area for the graph.
void	**setDrawAxis**(boolean drawAxis) Sets the axis on/off parameter.
void	**setGraphSettings**(GraphSettings[] graphSettings) Sets the settings for the indvidual graphs drawed using the general settings in this object.
void	**setPlainTextExpressionForGraph**(int index, java.lang.String plainTextMath) Wrapper method to let the caller manipulate the plaintext math expression of one of the graph setting objects directly.
void	**setScaleIndicatorSizeX**(double scaleIndicatorSizeX) Sets the size of (distance between) scale indicators to use for the x-axis, if applicable for the type of graph object these settings regard.
void	**setScaleIndicatorSizeY**(double scaleIndicatorSizeY) Sets the size of (distance between) scale indicators to use for the y-axis, if applicable for the type of graph object these settings regard.
void	**setUserParameters**(java.util.Hashtable userParameters) Sets the parameters that can be set by the user, in the form name => value.
void	**setXInterval**(MathInterval xInterval) Sets the interval (first dimension values) these settings applies to (in real world values), i.e. this and yInterval defines the area to draw.
void	**setYInterval**(MathInterval yInterval) Sets the interval (second dimension values) these settings applies to (in real world values), i.e. this and xInterval defines the area to draw.

void	**toggleAxis**() Toggles the status for the drawAxis-setting.
java.lang.String	**toString**() Returns a string-representation of the object.

Field Detail

backgroundColor

```
private java.awt.Color backgroundColor
```

The color used as a background for the drawing-area for the graph.

---

drawAxis

```
private boolean drawAxis
```

If `true`, a GraphDrawer based on these settings will draw the axis.

---

scaleIndicatorSizeX

```
private double scaleIndicatorSizeX
```

The size of (distance between) scale indicators to use for the x-axis, if applicable for the type of graph object these settings regard. Default value is 0 (i.e. no scale indicators).

---

scaleIndicatorSizeY

```
private double scaleIndicatorSizeY
```

The size of (distance between) scale indicators to use for the y-axis, if applicable for the type of graph object these settings regard. Default value is 0 (i.e. no scale indicators).

---

xInterval

```
private MathInterval xInterval
```

The interval (first dimension values) these settings applies to (in real world values), i.e. this and yInterval defines the area to draw.

---

yInterval

```
private MathInterval yInterval
```

The interval (second dimension values) these settings applies to (in real world values), i.e. this and xInterval defines the area to draw. Does not need to be set (if not set, will default to using min/max computed values).

---

graphSettings

```
private GraphSettings[] graphSettings
```

Settings for the indvidual graphs drawed using the general settings in this object.

---

userParameters

```
private java.util.Hashtable userParameters
```

The parameters that can be set by the user, in the form name => value (i.e. the names of the variables are used as keys when hashing). Semantics: When null no user parameters exists. The value corresponding to each key should be parseable as a Double. If there is multiple expressions drawn, the same substitution is done in all of the expressions.

DEFAULT_NUMBER_OF_STEPS

```
private static final int DEFAULT_NUMBER_OF_STEPS
```

A constant that gives the default number of steps used for computing pointvalues for a graph.

Constructor Detail

GraphDrawerSettings

```
public GraphDrawerSettings()
```

Creates empty settings object with default settings.

Method Detail

getDrawAxis

```
public boolean getDrawAxis()
```

Gets the axis on/off parameter. If `true`, a GraphDrawer based on these settings will draw the axis.

---

setDrawAxis

```
public void setDrawAxis(boolean drawAxis)
```

Sets the axis on/off parameter. If `true`, a GraphDrawer based on these settings will draw the axis.

---

getScaleIndicatorSizeX

```
public double getScaleIndicatorSizeX()
```

Gets the size of (distance between) scale indicators to use for the x-axis, if applicable for the type of graph object these settings regard.

---

setScaleIndicatorSizeX

```
public void setScaleIndicatorSizeX(double scaleIndicatorSizeX)
```

Sets the size of (distance between) scale indicators to use for the x-axis, if applicable for the type of graph object these settings regard. Set to 0 to remove scale indicators.

---

getScaleIndicatorSizeY

```
public double getScaleIndicatorSizeY()
```

Gets the size of (distance between) scale indicators to use for the y-axis, if applicable for the type of graph object these settings regard.

---

setScaleIndicatorSizeY

```
public void setScaleIndicatorSizeY(double scaleIndicatorSizeY)
```

Sets the size of (distance between) scale indicators to use for the y-axis, if applicable for the type of graph object these settings regard. Set to 0 to remove scale indicators.

---

getBackgroundColor

```
public java.awt.Color getBackgroundColor()
```

Gets the color used as a background for the drawing-area for the graph.

---

setBackgroundColor

```
public void setBackgroundColor(java.awt.Color backgroundColor)
```

Sets the color used as a background for the drawing-area for the graph.

---

getGraphSettings

```
public GraphSettings[] getGraphSettings()
```

Gets the settings for the indvidual graphs drawed using the general settings in this object.

---

setGraphSettings

```
public void setGraphSettings(GraphSettings[] graphSettings)
```

Sets the settings for the indvidual graphs drawed using the general settings in this object.

---

getNumberOfGraphs

```
public int getNumberOfGraphs()
```

Returns the number of graphs these settings contain, i.e. how many GraphSettings-objects that are present

---

hasGraphs

```
public boolean hasGraphs()
```

Returns `true` if this object includes settings for any graphs as well.

---

getGraphSettingsAtIndex

```
public GraphSettings getGraphSettingsAtIndex(int index)
```

Gets the graph setting object with the given (0-based) index, if it exists. If it doesn't an IllegalArgumentException is thrown.

---

setPlainTextExpressionForGraph

```
public void setPlainTextExpressionForGraph(int index,
                                           java.lang.String plainTextMath)
```

Wrapper method to let the caller manipulate the plaintext math expression of one of the graph setting objects directly. If the index is invalid an IllegalArgumentException is thrown.

---

getXInterval

```
public MathInterval getXInterval()
```

Gets the interval (first dimension values) these settings applies to (in real world values), i.e. this and yInterval defines the area to draw.

---

setXInterval

```
public void setXInterval(MathInterval xInterval)
```

Sets the interval (first dimension values) these settings applies to (in real world values), i.e. this and yInterval defines the area to draw. Should always be set to a starting value before attempting to draw a graph using the settings.

See Also:
`setYInterval(no.inspera.applications.publish.math.MathInterval)`

---

getYInterval

```
public MathInterval getYInterval()
```

Sets the interval (second dimension values) these settings applies to (in real world values), i.e. this and xInterval defines the area to draw. If not set, the drawing code should default to using min/max computed values.

---

setYInterval

```
public void setYInterval(MathInterval yInterval)
```

Sets the interval (second dimension values) these settings applies to (in real world values), i.e. this and xInterval defines the area to draw. Does not need to be set (if not set, will default to using min/max computed values).

See Also:
`setXInterval(no.inspera.applications.publish.math.MathInterval)`

---

adjustStepSize

```
public void adjustStepSize()
```

Adjusts the stepsize, using the default number of steps, so that the graphs will get a correct number of samples according to the size of their definition interval.

---

adjustStepSize

```
public void adjustStepSize(int numberOfSteps)
```

Adjusts the stepsize so that the graphs will get a correct number of samples according to the size of their definition interval.

---

getUserParameters

```
public java.util.Hashtable getUserParameters()
```

Gets the parameters that can be set by the user, in the form name => value.

---

setUserParameters

```
public void setUserParameters(java.util.Hashtable userParameters)
```

Sets the parameters that can be set by the user, in the form name => value.

---

toggleAxis

```
public void toggleAxis()
```

Toggles the status for the drawAxis-setting. (I.e. if it was on before the call it is off after and vice versa.)

---

toString

```
public java.lang.String toString()
```

Returns a string-representation of the object. This method is mostly useful for debugging.

Overrides:

`toString` in class `java.lang.Object`

no.inspera.applications.publish.math

Class GraphSettings

```
java.lang.Object
  |
  +--no.inspera.applications.publish.math.GraphSettings
```

public class **GraphSettings**
extends java.lang.Object

This class is an collection of settings for one specific graph in the graph drawer, such as the range it is defined for, the color to use, etc. The class is used both by the server side part and by the graph drawer applet.

Version:
$Revision: 1.4 $ $Date: 2003/05/18 20:21:48 $

Field Summary

private boolean	**active** Holds the state (active/inactive) of the part of the graph that these settings apply to, if true it will be drawn.
private MathInterval	**appliesToInterval** The interval (first dimension values) these settings applies to, i.e. the definition interval for the graph of the related mathematical expression.
private java.awt.Color	**graphColor** The color used for drawing the graph.
private GraphDrawerSettings	**parentSettings** Reference to the settings-object for the graphdrawer this settings object is a part of.
private java.lang.String	**plainTextExpression** Holds the plaintext-version of an mathematical expression for the graph drawer.
private double	**stepSizeForComputation** The difference between each value which will be computed for drawing the graph.

Constructor Summary

GraphSettings(GraphDrawerSettings parentSettings) Creates empty settings object with default settings.

Method Summary	
double	**getComputeStepSize**() Gets the difference between each value which will be computed for drawing the graph.
java.awt.Color	**getGraphColor**() Gets the color used for drawing the graph.
GraphDrawerSettings	**getParentSettings**() Gets a reference to the settings-object for the graphdrawer this settings object is a part of.
java.lang.String	**getPlainTextExpression**() Gets the plaintext-version of an methematical expression for the graph drawer.
MathInterval	**getXInterval**() Gets the interval (first dimension values) these settings applies to.
boolean	**hasPlainTextExpression**() Will return true if this settings object contains a plain-text version of a mathematical expression.
boolean	**isActive**() Returns the state (active/inactive) of the part of the graph that these settings apply to, if true it will be drawn.
void	**setActive**(boolean active) Sets the state of the part of the graph that these settings apply to, if true it will be drawn.
void	**setComputeStepSize**(double stepSize) Sets the difference between each value which will be computed for drawing the graph.
void	**setGraphColor**(java.awt.Color graphColor) Sets the color used for drawing the graph.
void	**setPlainTextExpression**(java.lang.String plainTextExpression) Sets the plaintext-version of an methematical expression for the graph drawer.
void	**setXInterval**(MathInterval interval) Sets the interval (first dimension values) these settings applies to.
void	**toggleActive**() Toggles the status for the active-setting.
java.lang.String	**toString**() Returns a string-representation of the object.

Field Detail

appliesToInterval

```
private MathInterval appliesToInterval
```

The interval (first dimension values) these settings applies to, i.e. the definition interval for the graph of the related mathematical expression.

---

stepSizeForComputation

```
private double stepSizeForComputation
```

The difference between each value which will be computed for drawing the graph. Only applies to graphs where this setting has a meaning, done this to avoid specializations of the class just to include another variable.

---

graphColor

```
private java.awt.Color graphColor
```

The color used for drawing the graph.

---

active

```
private boolean active
```

Holds the state (active/inactive) of the part of the graph that these settings apply to, if true it will be drawn.

---

plainTextExpression

```
private java.lang.String plainTextExpression
```

Holds the plaintext-version of an mathematical expression for the graph drawer. If this is != null the hasPlainTextExpression()-method will return true. The form serialized here is produced by the getPlainTextMath()-method of the ContentMath-class.

See Also:
`ContentMath.getPlainTextMath()`

---

parentSettings

```
private GraphDrawerSettings parentSettings
```

Reference to the settings-object for the graphdrawer this settings object is a part of.

Constructor Detail

GraphSettings

```
public GraphSettings(GraphDrawerSettings parentSettings)
```

Creates empty settings object with default settings.

Method Detail

hasPlainTextExpression

```
public boolean hasPlainTextExpression()
```

Will return true if this settings object contains a plain-text version of a mathematical expression.

---

getPlainTextExpression

```
public java.lang.String getPlainTextExpression()
```

Gets the plaintext-version of an methematical expression for the graph drawer. The form serialized here is the same that is produced by the getPlainTextMath()-method of the ContentMath-class.

See Also:
`ContentMath.getPlainTextMath()`

---

setPlainTextExpression

```
public void setPlainTextExpression(java.lang.String plainTextExpression)
```

Sets the plaintext-version of an methematical expression for the graph drawer. Should follow the syntax supported by the plain text parser, i.e. will typically be produced by the getPlainTextMath()-method of the ContentMath-class.

See Also:
`ContentMath.getPlainTextMath()`

---

getXInterval

```
public MathInterval getXInterval()
```

Gets the interval (first dimension values) these settings applies to. Does this by doing an intersection with the parent interval.

---

setXInterval

```
public void setXInterval(MathInterval interval)
```

Sets the interval (first dimension values) these settings applies to.

---

getComputeStepSize

```
public double getComputeStepSize()
```

Gets the difference between each value which will be computed for drawing the graph.

---

setComputeStepSize

```
public void setComputeStepSize(double stepSize)
```

Sets the difference between each value which will be computed for drawing the graph. Only applies to graphs where this setting has a meaning, done this to avoid specializations of the class just to include another variable.

---

getGraphColor

```
public java.awt.Color getGraphColor()
```

Gets the color used for drawing the graph.

---

setGraphColor

```
public void setGraphColor(java.awt.Color graphColor)
```

Sets the color used for drawing the graph.

---

isActive

```
public boolean isActive()
```

Returns the state (active/inactive) of the part of the graph that these settings apply to, if true it will be drawn.

---

setActive

```
public void setActive(boolean active)
```

Sets the state of the part of the graph that these settings apply to, if true it will be drawn.

---

toggleActive

```
public void toggleActive()
```

Toggles the status for the active-setting. (I.e. if it was on before the call it is off after and vice versa.) Will result in switching the graph on/off in the graph drawer.

---

getParentSettings

```
public GraphDrawerSettings getParentSettings()
```

Gets a reference to the settings-object for the graphdrawer this settings object is a part of.

---

toString

```
public java.lang.String toString()
```

Returns a string-representation of the object. This method is mostly useful for debugging.

Overrides:

`toString` in class `java.lang.Object`

no.inspera.applications.publish.math

Class MathToolsServerSide

```
java.lang.Object
  |
  +--no.inspera.applications.publish.math.MathToolsServerSide
```

public class **MathToolsServerSide**
extends java.lang.Object

This class contains tool-methods not related to one single class in the math application.

It also contains some methods refactored here to avoid using server-side specific classes in the classes which are shared between the applet and the server-side app. (This proved to be a problem when using Sun's plugin as it tried to load classes even if the method with the "problem" signature was never called from the applet part.)

As all methods are static there is no need to instantiate the class to use them.

The class is used only on the server side.

Version:
$Revision: 1.5 $ $Date: 2003/05/18 20:21:48 $

Constructor Summary

private	**MathToolsServerSide**() "Hides" the constructor (the default implementation if not provided is public), as it should never be used -- all the relevant methods here are static, so callers should not instantiate this class.

Method Summary

static GraphDrawerSettings	**getGraphDrawerSettingsFromRequest**(javax.servlet.http.HttpServletRequest request) Creates a GraphDrawerSettings object with settings given in the HttpServletRequest-object.
static void	**writeJPEGImage**(java.awt.image.BufferedImage outputImage, java.io.OutputStream outputStream, int quality) Writes a image to a file/HTTP-request, represented by the given outputstream, in the JPEG format.

Constructor Detail

MathToolsServerSide

```
private MathToolsServerSide()
```

"Hides" the constructor (the default implementation if not provided is public), as it should never be used -- all the relevant methods here are static, so callers should not instantiate this class.

Method Detail

getGraphDrawerSettingsFromRequest

```
public static GraphDrawerSettings
getGraphDrawerSettingsFromRequest(javax.servlet.http.HttpServletRequest req
uest)
```

Creates a `GraphDrawerSettings` object with settings given in the HttpServletRequest-object. The object will also be filled with `GraphSettings`-objects descibing the individual graphs. The primarily use for this method is when creating an image of a graph, for instance for use in a PDF-document.

See Also:

`GraphDrawerSettings`, `GraphSettings`

---

writeJPEGImage

```
public static void writeJPEGImage(java.awt.image.BufferedImage outputImage,
                                  java.io.OutputStream outputStream,
                                  int quality)
                           throws java.io.IOException
```

Writes a image to a file/HTTP-request, represented by the given outputstream, in the JPEG format.

Parameters:

`quality` - A number in the interval [0,100] which indicates the quality (and file size...) of the produced file.

no.inspera.applications.publish.math

Class ContentMathViewer

```
java.lang.Object
  |
  +--no.inspera.applications.publish.math.ContentMathViewer
```

Direct Known Subclasses:

ContentMathImageViewer, ContentMathPlainViewer

---

public abstract class **ContentMathViewer**
extends java.lang.Object

This class provides the interface that has to be implemented by the classes provide the translation from MathML to the end-format shown to the user.

It will do this by sending its results to the given HttpServletRequest-object. (So inclusion into PDF will be done through a HTTP-request that delivers an image file to the FOP-processor.)

This class provides a factory-method to get an instance of such a subclass and common functionality among the different formats (or libraries giving those formats) might also be implemented here.

Version:

$Revision: 1.4 $ $Date: 2003/05/18 20:21:48 $

Field Summary

protected ContentMath[]	**mathObjects** The math objects to display in the requested format.
protected javax.servlet.http.HttpServletRequest	**request** The request we're handling.
protected javax.servlet.http.HttpServletResponse	**response** The response for the request we're handling and will write the result to.

Method Summary	
static ContentMathViewer	**getInstance**(ContentMath[] mathObjectArray, javax.servlet.http.HttpServletRequest request, javax.servlet.http.HttpServletResponse response) Factory-method for getting an instance of a subclass of ContentMathViewer to display the ContentMath-objects in the mathObjectArray array.
static ContentMathViewer	**getInstance**(ContentMath m, javax.servlet.http.HttpServletRequest request, javax.servlet.http.HttpServletResponse response) Factory-method for getting an instance of a subclass of ContentMathViewer to display the ContentMath-object m.
abstract void	**writeResult**() Draw the graph to the given graphics-object, the actual drawing is required to be implemented by subclasses.

Method Detail

writeResult

```
public abstract void writeResult()
```

Draw the graph to the given graphics-object, the actual drawing is required to be implemented by subclasses.

---

getInstance

```
public static ContentMathViewer getInstance(ContentMath m,

javax.servlet.http.HttpServletRequest request,

javax.servlet.http.HttpServletResponse response)
```

Factory-method for getting an instance of a subclass of ContentMathViewer to display the ContentMath-object m. Will create image of a symbolic view or graph view of the expression, according to parameters in the request-object.

---

getInstance

```
public static ContentMathViewer getInstance(ContentMath[] mathObjectArray,

javax.servlet.http.HttpServletRequest request,

javax.servlet.http.HttpServletResponse response)
```

Factory-method for getting an instance of a subclass of ContentMathViewer to display the ContentMath-objects in the mathObjectArray array. For symbolic display (when a request-parameter format set to "symbolic") only one expression should be sent, see also the convenience-method for doing this.

no.inspera.applications.publish.math

Class ContentMathPlainViewer

```
java.lang.Object
  |
  +--no.inspera.applications.publish.math.ContentMathViewer
        |
        +--no.inspera.applications.publish.math.ContentMathPlainViewer
```

public class **ContentMathPlainViewer**
extends ContentMathViewer

This class implements the ContentMathViewer-method to provide a simple dumper of the MathML content directly to the caller.

Version:
$Revision: 1.4 $ $Date: 2003/05/18 20:21:48 $

Field Summary

private static org.apache.log4j.Category	**logCat** The logger for writing messages to the log, provides central control on the level of logging as well as display of the originator of the log entry.

Constructor Summary

ContentMathPlainViewer() Standard ctor that creates a ContentMathPlainViewer.

Method Summary

void	**writeResult**() Write the result to the HttpServletRequest set as the member variable.

Method Detail

writeResult

public void **writeResult**()

Write the result to the HttpServletRequest set as the member variable. In this case a simple dump of the MathML.

Overrides:

writeResult in class ContentMathViewer

no.inspera.applications.publish.math

Class ContentMathImageViewer

```
java.lang.Object
  |
  +--no.inspera.applications.publish.math.ContentMathViewer
        |
        +--no.inspera.applications.publish.math.ContentMathImageViewer
```

public class **ContentMathImageViewer**
extends ContentMathViewer

This class implements the ContentMathViewer-method to provide a image produced from the MathML to the caller. The class will only run on the server side and requires JDK1.2+ to run (uses a JPEG-encoder from the com.sun-package).

Version:
$Revision: 1.8 $ $Date: 2003/05/18 20:21:48 $

Field Summary

private int	**DEFAULT HEIGHT** The height to use for the produced image if no width is indicated in request parameters.
private int	**DEFAULT WIDTH** The width to use for the produced image if no width is indicated in request parameters.
private static org.apache.log4j.Category	**logCat** The logger for writing messages to the log, provides central control on the level of logging as well as display of the originator of the log entry.

Fields inherited from class no.inspera.applications.publish.math.ContentMathViewer
mathObjects, request, response

Constructor Summary

ContentMathImageViewer() Standard ctor that creates a ContentMathImageViewer.

Method Summary

void	**writeResult**() Write the result to the HttpServletRequest set as the member variable.

Field Detail

logCat

```
private static org.apache.log4j.Category logCat
```

The logger for writing messages to the log, provides central control on the level of logging as well as display of the originator of the log entry.

DEFAULT_WIDTH

```
private final int DEFAULT_WIDTH
```

The width to use for the produced image if no width is indicated in request parameters.

DEFAULT_HEIGHT

```
private final int DEFAULT_HEIGHT
```

The height to use for the produced image if no width is indicated in request parameters.

Constructor Detail

ContentMathImageViewer

```
public ContentMathImageViewer()
```

Standard ctor that creates a ContentMathImageViewer.

Method Detail

writeResult

```
public void writeResult()
```

Write the result to the HttpServletRequest set as the member variable. In this case a image produced from the MathML (of either a graph or a symbolic view of an equation) is written.

Overrides:

`writeResult` in class `ContentMathViewer`

no.inspera.applications.publish.math.applets

Class GraphDrawerApplet

```
java.lang.Object
  |
  +--java.awt.Component
        |
        +--java.awt.Container
              |
              +--java.awt.Panel
                    |
                    +--java.applet.Applet
                          |
                          +--
no.inspera.applications.publish.math.applets.GraphDrawerApplet
```

All Implemented Interfaces:

javax.accessibility.Accessible, GraphDrawerAppletSettings, java.awt.image.ImageObserver, java.awt.MenuContainer, java.lang.Runnable, java.io.Serializable

---

public class **GraphDrawerApplet**
extends java.applet.Applet
implements GraphDrawerAppletSettings, java.lang.Runnable

This class is an implementation of a graph drawer in the form of an Applet. It shares classes with the server-side part of the solution to enable output in form of graphs to be included in print documents. The applet itself will be pacakged in a jar-file before being deployed to the application-/webserver, see the create-GraphApplet-jar.bat file for automating that process.

Version:

$Revision: 1.9 $ $Date: 2003/05/18 20:21:48 $

Field Summary

private java.awt.Image	**animationImage** The animation image to use, if any.
private int	**animationStepSize** The stepSize used for animation, counted in number of values.
private java.lang.Thread	**animationThread** Thread used for the animation.
private int	**animationTotalTimeSec** The total time it should take to "play through" the animation, the logic will adjust stepsizes and speed to use the correct amount of time.
private java.awt.Image	**backgroundImage** The background image to use, if any.
private java.awt.Color	**bgcolor** The background color to use in the applet.

private int	**currentPointValueNumber** The current value number (index) that is used for animating or showing the user the value in a point.
private java.awt.Dimension	**d** A Dimension-object that describes the size of the applet.
private int	**drawState** State variabel used to indicating which state the applet is in, will change when the applet receives calls from javascript as well as when a task is finished.
private java.awt.Color	**fgcolor** The foreground color to use in the applet.
private java.awt.FontMetrics	**fm** The font informaion (character width etc.) of the small font used in the applet.
private java.awt.Font	**font** The font used in the applet.
private GraphDrawer	**graphDrawer** This is a graph drawer object which the applet shares with the server side parts of the solution which does the actual drawing (by being instantiated for a particular drawer type).
private GraphDrawerSettings	**graphDrawerSettings** Each of the parts of the graph (if multiple) will have an PVP and a settings-object, this object holds all the settings objects as well as the global settings for the graph drawer.
private boolean	**includeAnimationTrace** If true and in animation mode, a trace will be written (the graph) up to the point where the animationImage is drawed.
private java.awt.Graphics	**offScreenGfx** Graphics-object used for implementing double-buffering in the applet (i.e. everything is drawn to memory before switch the whole image to avoid flickering).
private java.awt.Image	**offScreenImage** Image-object used for implementing double-buffering in the applet (i.e. everything is drawn to memory before switch the whole image to avoid flickering).
private PointValueProducer[]	**pointValueProducerArray** Each of the parts of the graph (if multiple) will have an PVP and a settings-object, this array holds the PVP-objects.
private double	**selectedXVal** The x-value the user has selected (typically to get a value shown in the GUI).
private java.lang.String[]	**statusMessageArray** Used for status messages that will be shown the user

private boolean	**tileBackground** If a background image is used, this parameter tells wether or not it should be tiled (e.g. repeated).
private java.awt.MediaTracker	**tracker** Used for being sure that images are loaded before starting to draw.
private java.awt.Point	**zoomBottomRightPoint** The bottom right point of the selected box for zooming.
private java.awt.Point	**zoomTopLeftPoint** The top left point of the selected box for zooming.

Method Summary

private void	**d**(java.lang.String s) Used for writing a string to the applet console.
void	**destroy**() Lifecycle-method of the standard framework, clean up the resources used (typically called by the JVM when the page is unloaded).
private void	**displayMessage**(java.awt.Graphics g, java.lang.String[] msg) Displays a centered message using default fontmetrics
private void	**displayMessage**(java.awt.Graphics g, java.lang.String[] msg, java.awt.FontMetrics fm) Displays a centered message
private void	**displayMessage**(java.awt.Graphics g, java.lang.String msg, int y) Displays a centered message at y-coord y
private void	**drawAnimationImage**(java.awt.Graphics g) Helper method that draws the animation image in the current state/position.
void	**drawBackground**(java.awt.Graphics g) Draws the background (if background image is set this will be drawn, tiled or not).
private void	**drawRectangle**(java.awt.Graphics g, java.awt.Point topLeftPoint, java.awt.Point bottomRightPoint) Helper method that draws a rectangle using lines, so that the rectangle can go in both directions (and not only down to the left as in drawRect).
private void	**err**(java.lang.String s) Used for writing a string with an error message to the applet console.

private void	**err**(java.lang.String s, java.lang.Exception e) Used for writing a string with an error message and a stack trace to the applet console.
java.lang.String	**getAppletInfo**() Returns a short string describing the applet, used by some browsers in the "about" page.
private boolean	**getBooleanParameter**(java.lang.String name) Gets the parameter named name as a boolean.
private boolean	**getBooleanParameter**(java.lang.String name, boolean defaultVal) Gets the parameter named name as a boolean.
private java.awt.Color	**getColorParameter**(java.lang.String name) Gets the parameter named name as a Color-object, the parameter should be of the format rrggbb.
private double	**getDoubleParameter**(java.lang.String name) Gets the parameter named name as a double.
private int	**getIntParameter**(java.lang.String name) Gets the parameter named name as a int
private PointValue	**getPointValueNumber**(int index) Helper method that is used when animating, gets value at the index given by the parameter.
private GraphSettings	**getSettingsFromParams**(int settingsNumber, GraphDrawerSettings parent) Gets the graph drawer setting from applet parameters (which may be generated by server side parts of the solution).
private int	**getTotalNumberOfPointValues**() Helper method that gets the total number of values available from all PVPs, i.e. the sum across all intervals.
void	**init**() Initialize applet.
boolean	**keyDown**(java.awt.Event e, int key) Called when the user presses a keyboard key, we use old event model with intention to be compatible with old browsers with ancient JVMs.
boolean	**keyUp**(java.awt.Event e, int key) Called when the user releases a keyboard key, we use old event model with intention to be compatible with old browsers with ancient JVMs.
boolean	**mouseDown**(java.awt.Event e, int x, int y) Called when the user clicks the mouse button, we use old event model with intention to be compatible with old browsers with ancient JVMs.

boolean	**mouseMove**(java.awt.Event e, int x, int y) Called when the user moves the mouse pointer, we use old event model with intention to be compatible with old browsers with ancient JVMs.
private void	**p**(java.lang.String s) Used for writing a string to the applet console.
void	**paint**(java.awt.Graphics g) Does the actual painting by drawing to an offscreen buffer to improve performance (while using slightly more memory, of course).
private void	**resetAnimation**() Resets the graph and animation.
void	**run**() Implements the run-method of Runnable to be able to be set as the target of a Thread (payload if you will), to do the animation.
void	**sendMsg**(java.lang.String msg, java.lang.String param) Single interaction-point used for sending simple messages from javascript/HTML GUI to the applet, the applet then changes behaviour accordingly to the message.
void	**start**() Lifecycle-method of the standard framework.
void	**startAnimation**() Starts a new animation by creating a thread that can to the actual animation, the implementation of the logic is in the run()-method of this class.
void	**stop**() Lifecycle-method of the standard framework, suspend (typically called by the JVM when the page is out of focus).
void	**update**(java.awt.Graphics g) Updates the display, clearing is done manually by overwriting it all every time and using double buffering
private void	**updateAnimationState**() Helper method that takes care of updating the state of the animation correctly.

Field Detail

animationThread

```
private java.lang.Thread animationThread
```

Thread used for the animation.

---

fm

```
private java.awt.FontMetrics fm
```

The font informaion (character width etc.) of the small font used in the applet.

---

font

```
private java.awt.Font font
```

The font used in the applet.

---

bgcolor

```
private java.awt.Color bgcolor
```

The background color to use in the applet.

---

fgcolor

```
private java.awt.Color fgcolor
```

The foreground color to use in the applet.

---

d

```
private java.awt.Dimension d
```

A Dimension-object that describes the size of the applet.

---

offScreenImage

```
private java.awt.Image offScreenImage
```

Image-object used for implementing double-buffering in the applet (i.e. everything is drawn to memory before switch the whole image to avoid flickering).

---

offScreenGfx

```
private java.awt.Graphics offScreenGfx
```

Graphics-object used for implementing double-buffering in the applet (i.e. everything is drawn to memory before switch the whole image to avoid flickering).

---

pointValueProducerArray

```
private PointValueProducer[] pointValueProducerArray
```

Each of the parts of the graph (if multiple) will have an PVP and a settings-object, this array holds the PVP-objects.

---

graphDrawerSettings

`private GraphDrawerSettings graphDrawerSettings`

Each of the parts of the graph (if multiple) will have an PVP and a settings-object, this object holds all the settings objects as well as the global settings for the graph drawer.

---

graphDrawer

`private GraphDrawer graphDrawer`

This is a graph drawer object which the applet shares with the server side parts of the solution which does the actual drawing (by being instantiated for a particular drawer type).

---

drawState

`private int drawState`

State variabel used to indicating which state the applet is in, will change when the applet receives calls from javascript as well as when a task is finished.

---

statusMessageArray

`private java.lang.String[] statusMessageArray`

Used for status messages that will be shown the user

---

backgroundImage

`private java.awt.Image backgroundImage`

The background image to use, if any. Will be null if no background image should be used. The background image is specific for the applet and will never be painted on the server side.

---

animationImage

`private java.awt.Image animationImage`

The animation image to use, if any. Will be null if no animation image should be used. The animation parts is specific for the applet and will never be used on the server side.

---

currentPointValueNumber

`private int currentPointValueNumber`

The current value number (index) that is used for animating or showing the user the value in a point.

---

animationStepSize

`private int animationStepSize`

The stepSize used for animation, counted in number of values.

---

animationTotalTimeSec

```
private int animationTotalTimeSec
```

The total time it should take to "play through" the animation, the logic will adjust stepsizes and speed to use the correct amount of time.

---

includeAnimationTrace

```
private boolean includeAnimationTrace
```

If true and in animation mode, a trace will be written (the graph) up to the point where the animationImage is drawed.

---

tileBackground

```
private boolean tileBackground
```

If a background image is used, this parameter tells wether or not it should be tiled (e.g. repeated). If it is not to be repeated, it will be drawn centered.

---

tracker

```
private java.awt.MediaTracker tracker
```

Used for being sure that images are loaded before starting to draw.

---

zoomTopLeftPoint

```
private java.awt.Point zoomTopLeftPoint
```

The top left point of the selected box for zooming.

---

zoomBottomRightPoint

```
private java.awt.Point zoomBottomRightPoint
```

The bottom right point of the selected box for zooming.

---

selectedXVal

```
private double selectedXVal
```

The x-value the user has selected (typically to get a value shown in the GUI).

Method Detail

getAppletInfo

```
public java.lang.String getAppletInfo()
```

Returns a short string describing the applet, used by some browsers in the "about" page.

Overrides:

`getAppletInfo` in class `java.applet.Applet`

---

getIntParameter

```
private int getIntParameter(java.lang.String name)
```

Gets the parameter named `name` as a int

---

getDoubleParameter

```
private double getDoubleParameter(java.lang.String name)
```

Gets the parameter named `name` as a double.

---

getBooleanParameter

```
private boolean getBooleanParameter(java.lang.String name)
```

Gets the parameter named `name` as a boolean. Defaults to `false` if the value is not set.

---

getBooleanParameter

```
private boolean getBooleanParameter(java.lang.String name,
                                    boolean defaultVal)
```

Gets the parameter named `name` as a boolean. Defaults to `defaultVal` if the value is not set as a parameter.

---

getColorParameter

```
private java.awt.Color getColorParameter(java.lang.String name)
```

Gets the parameter named `name` as a Color-object, the parameter should be of the format `rrggbb`. If this method fails, no exception is thrown and a black Color-object is returned.

---

getSettingsFromParams

```
private GraphSettings getSettingsFromParams(int settingsNumber,
                                            GraphDrawerSettings parent)
```

Gets the graph drawer setting from applet parameters (which may be generated by server side parts of the solution).

---

init

```
public void init()
```

Initialize applet.

Overrides:

`init` in class `java.applet.Applet`

---

start

```
public void start()
```

Lifecycle-method of the standard framework.

Overrides:

`start` in class `java.applet.Applet`

---

stop

```
public void stop()
```

Lifecycle-method of the standard framework, suspend (typically called by the JVM when the page is out of focus).

Overrides:

`stop` in class `java.applet.Applet`

---

destroy

```
public void destroy()
```

Lifecycle-method of the standard framework, clean up the resources used (typically called by the JVM when the page is unloaded).

Overrides:

`destroy` in class `java.applet.Applet`

---

run

```
public void run()
```

Implements the run-method of Runnable to be able to be set as the target of a Thread (payload if you will), to do the animation.

Specified by:

`run` in interface `java.lang.Runnable`

---

resetAnimation

```
private void resetAnimation()
```

Resets the graph and animation. Stops any animation. Called from stop(), i.e. typically when the applet has lost focus (but the page is not closed) -- enter a standby-state.

---

startAnimation

```
public void startAnimation()
```

Starts a new animation by creating a thread that can to the actual animation, the implementation of the logic is in the run()-method of this class.

---

updateAnimationState

```
private void updateAnimationState()
```

Helper method that takes care of updating the state of the animation correctly. Called from the main loop of the animation thread.

---

update

```
public void update(java.awt.Graphics g)
```

Updates the display, clearing is done manually by overwriting it all every time and using double buffering

Overrides:

`update` in class `java.awt.Container`

---

paint

```
public void paint(java.awt.Graphics g)
```

Does the actual painting by drawing to an offscreen buffer to improve performance (while using slightly more memory, of course). The actual drawing logic is performed by separate classes that is shared by the server side and client side apps.

Overrides:

`paint` in class `java.awt.Container`

Parameters:

`g` - is the Graphics-object to paint to, will use its own object for the drawing and then draw the entire result to `g`

---

sendMsg

```
public void sendMsg(java.lang.String msg,
                    java.lang.String param)
```

Single interaction-point used for sending simple messages from javascript/HTML GUI to the applet, the applet then changes behaviour accordingly to the message.

The naming is chosen as seen from the "HTML side", to give natural naming of calls in the GUI.

For some complex calls from the GUI we might use specific functions to avoid complex serialization of parameters from javascript, but for simple messages it seems like a good idea to avoid having a separate method for each call.

Parameters:

`msg` - The message received, as defined by protocol rules.

`param` - Any extra information required by the `msg` can be sent using this parameter, values will vary according to protocol rules.

---

displayMessage

```
private void displayMessage(java.awt.Graphics g,
                            java.lang.String[] msg)
```

Displays a centered message using default fontmetrics

---

displayMessage

```
private void displayMessage(java.awt.Graphics g,
                            java.lang.String[] msg,
                            java.awt.FontMetrics fm)
```

Displays a centered message

---

drawBackground

```
public void drawBackground(java.awt.Graphics g)
```

Draws the background (if background image is set this will be drawn, tiled or not). Will also clear the drawing area to the background color.

---

displayMessage

```
private void displayMessage(java.awt.Graphics g,
                            java.lang.String msg,
                            int y)
```

Displays a centered message at y-coord y

---

drawRectangle

```
private void drawRectangle(java.awt.Graphics g,
                           java.awt.Point topLeftPoint,
                           java.awt.Point bottomRightPoint)
```

Helper method that draws a rectangle using lines, so that the rectangle can go in both directions (and not only down to the left as in drawRect). Saves us for some if-tests in the main code as well as giving more control over looks of the line.

---

drawAnimationImage

```
private void drawAnimationImage(java.awt.Graphics g)
```

Helper method that draws the animation image in the current state/position.

---

getPointValueNumber

```
private PointValue getPointValueNumber(int index)
```

Helper method that is used when animating, gets value at the index given by the parameter.

---

getTotalNumberOfPointValues

```
private int getTotalNumberOfPointValues()
```

Helper method that gets the total number of values available from all PVPs, i.e. the sum across all intervals. This will be used for computing the progress of the animation (and for animations the computation assume there is no overlap, which also seems quite natural (if not, the animation image would jump back or forward when changing defintion intervals).

---

keyDown

```
public boolean keyDown(java.awt.Event e,
                       int key)
```

Called when the user presses a keyboard key, we use old event model with intention to be compatible with old browsers with ancient JVMs.

Overrides:

`keyDown` in class `java.awt.Component`

---

keyUp

```
public boolean keyUp(java.awt.Event e,
                     int key)
```

Called when the user releases a keyboard key, we use old event model with intention to be compatible with old browsers with ancient JVMs.

Overrides:

`keyUp` in class `java.awt.Component`

---

mouseMove

```
public boolean mouseMove(java.awt.Event e,
                         int x,
                         int y)
```

Called when the user moves the mouse pointer, we use old event model with intention to be compatible with old browsers with ancient JVMs.

Overrides:

`mouseMove` in class `java.awt.Component`

---

mouseDown

```
public boolean mouseDown(java.awt.Event e,
                         int x,
                         int y)
```

Called when the user clicks the mouse button, we use old event model with intention to be compatible with old browsers with ancient JVMs.

Overrides:

`mouseDown` in class `java.awt.Component`

---

p

```
private void p(java.lang.String s)
```

Used for writing a string to the applet console.

---

d

```
private void d(java.lang.String s)
```

Used for writing a string to the applet console. Will only output the value if GraphDrawerAppletSettings.DEBUG=true.

See Also:

`GraphDrawerAppletSettings`

---

no.inspera.applications.publish.math.applets

Interface GraphDrawerAppletSettings

All Known Implementing Classes:

GraphDrawerApplet

public interface **GraphDrawerAppletSettings**

This interface defines constants used by the graphdrawer applet. Settings such as colors etc. can also be set using parameters to the applet, this is intended solely to provide default values and settings not relevant to the server side part of the solution at all (such as DEBUG-level of the output to the applet console). The constants that indicates the state of the applet (i.e. drawing a graph, showing message etc.) is also defined here and this interface should thus be implemented by other classes that needs to query the applet state.

Version:

$Revision: 1.6 $ $Date: 2003/05/18 20:21:48 $

Field Summary

`static java.awt.Color`	**BGCOLOR** The default background color.
`static boolean`	**DEBUG** The debuglevel, is true also debuginfo will be output in applet console window.
`static java.awt.Color`	**FGCOLOR** The default foreground color.
`static int`	**PADDING_GRAPHAREA** The size of the padding around the area used for the graph.
`static int`	**STATE_ANIMATING** Constant indicating one of the possible states the applet can be in.
`static int`	**STATE_ANIMATION_START** Constant indicating one of the possible states the applet can be in.
`static int`	**STATE_DRAW_GRAPH** Constant indicating one of the possible states the applet can be in.
`static int`	**STATE_SELECT_ZOOMBOX** Constant indicating one of the possible states the applet can be in.
`static int`	**STATE_SHOW_STATUSMESSAGE** Constant indicating one of the possible states the applet can be in.
`static int`	**STATE_SHOW_USERSELECTED_VALUE** Constant indicating one of the possible states the applet can be in.

static int	**STATE_SHOWMINMAX** Constant indicating one of the possible states the applet can be in.

Field Detail

DEBUG

```
public static final boolean DEBUG
```

The debuglevel, is true also debuginfo will be output in applet console window.

FGCOLOR

```
public static final java.awt.Color FGCOLOR
```

The default foreground color.

BGCOLOR

```
public static final java.awt.Color BGCOLOR
```

The default background color.

STATE_SHOW_STATUSMESSAGE

```
public static final int STATE_SHOW_STATUSMESSAGE
```

Constant indicating one of the possible states the applet can be in. This value indicates that a statusmessage is shown to the user.

STATE_DRAW_GRAPH

```
public static final int STATE_DRAW_GRAPH
```

Constant indicating one of the possible states the applet can be in. This value indicates that the graph should be drawn.

STATE_SELECT_ZOOMBOX

```
public static final int STATE_SELECT_ZOOMBOX
```

Constant indicating one of the possible states the applet can be in. This value indicates that the user is selecting a rectangle to zoom in on.

STATE_ANIMATION_START

```
public static final int STATE_ANIMATION_START
```

Constant indicating one of the possible states the applet can be in. This value indicates that we're waiting to start an animation.

STATE_ANIMATING

```
public static final int STATE_ANIMATING
```

Constant indicating one of the possible states the applet can be in. This value indicates that an animation is taking place.

STATE_SHOWMINMAX

```
public static final int STATE_SHOWMINMAX
```

Constant indicating one of the possible states the applet can be in. This value indicates that the graph is being drawn with indicators for min and max.

STATE_SHOW_USERSELECTED_VALUE

```
public static final int STATE_SHOW_USERSELECTED_VALUE
```

Constant indicating one of the possible states the applet can be in. This value indicates that the graph is being drawn along with indicators for pointvalues, selected by the user.

PADDING_GRAPHAREA

```
public static final int PADDING_GRAPHAREA
```

The size of the padding around the area used for the graph.

no.inspera.applications.publish.math

Class MathInterval

```
java.lang.Object
  |
  +--no.inspera.applications.publish.math.MathInterval
```

public class **MathInterval**
extends java.lang.Object

This class represents a mathematical interval. It might be open ended. The class is used both by the server side part and by the graph drawer applet.

Version:
$Revision: 1.4 $ $Date: 2003/05/18 20:21:48 $

Field Summary

private java.lang.Double	**maxVal** The maximum value of the interval, might be `null` which indicates there is no maximum value.
private java.lang.Double	**minVal** The minimum value of the interval, might be `null` which indicates there is no minimum value.
(package private) boolean	**openEndedMax** If true, the right end of the interval is open, i.e. the same as a >-style parenthesis in mathematical notation.
(package private) boolean	**openEndedMin** If true, the left end of the interval is open, i.e. the same as a <-style parenthesis in mathematical notation.

Constructor Summary

MathInterval() Creates a new math interval with no limits in either end.

Method Summary

java.lang.Double	**getMaxVal**() Gets the maximum value of the interval, might be `null` which indicates there is no maximum value.
java.lang.Double	**getMinVal**() Gets the minimum value of the interval, might be `null` which indicates there is no minimum value.

MathInterval	**intersect**(MathInterval otherInterval) Returns the intersection of this interval and the given interval.
boolean	**isOpenEndedMax**() Returns true if the interval is open in the right end, false if it is closed.
boolean	**isOpenEndedMin**() Returns true if the interval is open in the left end, false if it is closed.
void	**parseFromString**(java.lang.String s) Parses the values in this object from a String.
void	**setMaxVal**(java.lang.Double maxVal) Sets the maximum value of the interval, might be `null` which indicates there is no maximum value.
void	**setMinVal**(java.lang.Double minVal) Sets the minimum value of the interval, might be `null` which indicates there is no minimum value.
void	**setOpenEndedMax**(boolean openEndedMax) Sets the type of interval in the right end -- if true, the right end of the interval is open, i.e. the same as a >-style parenthesis in mathematical notation.
void	**setOpenEndedMin**(boolean openEndedMin) Sets the type of interval in the left end -- if true, the left end of the interval is open, i.e. the same as a <-style parenthesis in mathematical notation.
java.lang.String	**toString**() Returns a string-representation of the interval.
void	**zoom**(double factor) Changes the interval to another value via a zooming with the given factor (2 to make it twice the size, 0.5 to make it half the size for instance).

Field Detail

minVal

```
private java.lang.Double minVal
```

The minimum value of the interval, might be `null` which indicates there is no minimum value.

---

maxVal

```
private java.lang.Double maxVal
```

The maximum value of the interval, might be `null` which indicates there is no maximum value.

---

openEndedMin

```
boolean openEndedMin
```

If true, the left end of the interval is open, i.e. the same as a <-style parenthesis in mathematical notation. If false [is assumed, i.e. the interval includes the limiting value.

---

openEndedMax

```
boolean openEndedMax
```

If true, the right end of the interval is open, i.e. the same as a >-style parenthesis in mathematical notation. If false] is assumed, i.e. the interval does not include the limiting value.

Method Detail

getMinVal

```
public java.lang.Double getMinVal()
```

Gets the minimum value of the interval, might be `null` which indicates there is no minimum value.

---

setMinVal

```
public void setMinVal(java.lang.Double minVal)
```

Sets the minimum value of the interval, might be `null` which indicates there is no minimum value.

---

getMaxVal

```
public java.lang.Double getMaxVal()
```

Gets the maximum value of the interval, might be `null` which indicates there is no maximum value.

---

setMaxVal

```
public void setMaxVal(java.lang.Double maxVal)
```

Sets the maximum value of the interval, might be `null` which indicates there is no maximum value.

---

isOpenEndedMin

```
public boolean isOpenEndedMin()
```

Returns true if the interval is open in the left end, false if it is closed.

---

setOpenEndedMin

```
public void setOpenEndedMin(boolean openEndedMin)
```

Sets the type of interval in the left end -- if true, the left end of the interval is open, i.e. the same as a <-style parenthesis in mathematical notation. If false [is assumed, i.e. the interval includes the limiting value.

---

isOpenEndedMax

```
public boolean isOpenEndedMax()
```

Returns true if the interval is open in the right end, false if it is closed.

---

setOpenEndedMax

```
public void setOpenEndedMax(boolean openEndedMax)
```

Sets the type of interval in the right end -- if true, the right end of the interval is open, i.e. the same as a >-style parenthesis in mathematical notation. If false] is assumed, i.e. the interval does not include the limiting value.

---

zoom

```
public void zoom(double factor)
```

Changes the interval to another value via a zooming with the given factor (2 to make it twice the size, 0.5 to make it half the size for instance).

---

intersect

```
public MathInterval intersect(MathInterval otherInterval)
```

Returns the intersection of this interval and the given interval. I.e. it returns a interval with values present in both intervals, for example if this interval is [-3,3] and the given interval is [1,10] the interval [1,3] is returned.

---

toString

```
public java.lang.String toString()
```

Returns a string-representation of the interval. An example of a string returned could be [1.0,2.5> for an interval with values 1.0 and 2.5, openEndedMin=false and openEndedMax=true. The string "inf" will be used used if one of the values (min/max) are missing. This method is mostly useful for debugging. This class also supports parsing the values directly from a string of the same format, for this see the parseFromString-method.

Overrides:

`toString` in class `java.lang.Object`

See Also:

`parseFromString(java.lang.String)`

---

parseFromString

```
public void parseFromString(java.lang.String s)
```

Parses the values in this object from a String. For details of the string format, see the toString-method. This method will throw exceptions if format is not according to rules, so the caller must perform the format checking.

See Also:

`toString()`

no.inspera.db

Class MathDataAccess

```
java.lang.Object
  |
  +--no.inspera.db.DataAccess
        |
        +--no.inspera.db.MathDataAccess
```

All Implemented Interfaces:

no.inspera.services.metadataservice.DataTypeConstants, no.inspera.utils.ElearningConstants, no.inspera.utils.MagicObjectConstants, no.inspera.applications.publish.util.PublishConstants, java.io.Serializable

---

public class **MathDataAccess**
extends no.inspera.db.DataAccess
implements no.inspera.applications.publish.util.PublishConstants, no.inspera.services.metadataservice.DataTypeConstants, java.io.Serializable

Provides DB-access for the math-related classes.

This class uses a number of SQL-statements. These statements are located in a PropertyResourceBundle to enable editing without editing this class. The propertyResourceBundle is named MathDataAccess. This bundle is loaded through a static initializer so you need to restart the JVM if you need to update the SQL-statements.

Version:

$Revision: 1.4 $ $Date: 2003/05/18 20:21:48 $

Field Summary

private static java.util.PropertyResourceBundle	**bundle** The bundle to get the SQL-statements from.
private static java.lang.String	**bundleName** The name of the bundle to get the SQL-statements from.
private static MathDataAccess	**dataAccess** A reference used as a singleton to return a reference to clients.
private static org.apache.log4j.Category	**logCat** The logger for writing messages to the log, provides central control on the level of logging as well as display of the originator of the log entry.

Constructor Summary

protected	**MathDataAccess**() Std ctor for creating DA object, protected -- clients should always call factory methods to get DA instance

Method Summary

(package private) static void	D:\diplom-CD\javadoc\no\inspera\db\MathDataAccess.html - <clinit>()() Static initializers are being run when the class is loaded.
ContentMath[]	**getContentMath**(java.util.ArrayList contentItemIds, long languageId) Gets a set of live ContentMath-objects given a list of contentItemIds and a languageId.
ContentMath	**getContentMath**(long contentItemId, long languageId) Gets the live ContentMath-object given a contentItemId and a languageId.
static java.lang.String	**getCVSId**() Gets the CVSId of the current revision of the class (at compile time).
static MathDataAccess	**getDataAccess**() Factory method that gets a refererence to a MathDataAccess-instance used by clients when interfering with the database, synonym of getInstance().
static MathDataAccess	**getInstance**() Factory method that gets a refererence to a MathDataAccess-instance used by clients when interfering with the database, synonym of getDataAccess().
private static void	**instantiateSingleton**() Helper method used to instanatiate the singleton object.
long	**newContentMath**(ContentMath mathObject, long languageId, long marketplaceId, java.util.Locale locale) Stores a new ContentMath-object to the database.

Field Detail

bundleName

```
private static java.lang.String bundleName
```

The name of the bundle to get the SQL-statements from.

---

bundle

```
private static java.util.PropertyResourceBundle bundle
```

The bundle to get the SQL-statements from.

---

dataAccess

```
private static MathDataAccess dataAccess
```

A reference used as a singleton to return a reference to clients.

---

logCat

```
private static org.apache.log4j.Category logCat
```

The logger for writing messages to the log, provides central control on the level of logging as well as display of the originator of the log entry.

Constructor Detail

MathDataAccess

```
protected MathDataAccess()
```

Std ctor for creating DA object, protected -- clients should always call factory methods to get DA instance

Method Detail

```
static void ()
```

Static initializers are being run when the class is loaded. This one initializes the resourcebundle that holds the relevant SQL-statements.

---

getCVSId

```
public static java.lang.String getCVSId()
```

Gets the CVSId of the current revision of the class (at compile time). Useful for status reports and querying versions.

---

instantiateSingleton

```
private static void instantiateSingleton()
```

Helper method used to instanatiate the singleton object. This makes it unecessary to have the calling method synchronized, which is a great benefit for performance reasons.

---

newContentMath

```
public long newContentMath(ContentMath mathObject,
                           long languageId,
                           long marketplaceId,
                           java.util.Locale locale)
                     throws java.sql.SQLException
```

Stores a new ContentMath-object to the database. If the contentItem is set, a new revision of an existing item is created.

Returns:

the revisonId of the created content

---

getContentMath

```
public ContentMath getContentMath(long contentItemId,
                                  long languageId)
                           throws java.sql.SQLException
```

Gets the live ContentMath-object given a contentItemId and a languageId.

---

getContentMath

```
public ContentMath[] getContentMath(java.util.ArrayList contentItemIds,
                                    long languageId)
                             throws java.sql.SQLException
```

Gets a set of live ContentMath-objects given a list of contentItemIds and a languageId.

---

getDataAccess

```
public static MathDataAccess getDataAccess()
```

Factory method that gets a refererence to a MathDataAccess-instance used by clients when interfering with the database, synonym of getInstance().

---

getInstance

```
public static MathDataAccess getInstance()
```

Factory method that gets a refererence to a MathDataAccess-instance used by clients when interfering with the database, synonym of getDataAccess().

no.inspera.applications.publish.servlets

Class MathDisplayServlet

```
java.lang.Object
  |
  +--javax.servlet.GenericServlet
        |
        +--javax.servlet.http.HttpServlet
              |
              +--no.inspera.kernel.dispatcher.RequestProcessor
                    |
                    +--
no.inspera.applications.publish.servlets.MathDisplayServlet
```

All Implemented Interfaces:

no.inspera.utils.MagicObjectConstants, java.lang.Runnable, java.io.Serializable, javax.servlet.Servlet, javax.servlet.ServletConfig

---

public class **MathDisplayServlet**
extends no.inspera.kernel.dispatcher.RequestProcessor

This servlet creates mathematical expressions as graphs or in symbolic form to show to the user. This might be done as plain MathML or as a generated image.

This is performed via helper classes (mainly ContentMathViewer-family of classes and possibly GraphDrawer), almost no logic is present in this servlet itself. A lot of the classes used are common to the applet and the serverside solution as well. This is documented in each of the classes.

Version:

$Revision: 1.7 $ $Date: 2003/05/18 20:21:48 $

Field Summary

private static org.apache.log4j.Category	**logCat** The logger for writing messages to the log, provides central control on the level of logging as well as display of the originator of the log entry.

Method Summary

void	**doGet**(javax.servlet.http.HttpServletRequest request, javax.servlet.http.HttpServletResponse response) Handles HTTP Get requests with a request.
void	**doPost**(javax.servlet.http.HttpServletRequest request, javax.servlet.http.HttpServletResponse response) Handles HTTP Post requests, treats them equal to GET requests.
static java.lang.String	**getCVSId**() Returns the CVS Id of the file.

Field Detail

logCat

```
private static org.apache.log4j.Category logCat
```

The logger for writing messages to the log, provides central control on the level of logging as well as display of the originator of the log entry.

Method Detail

getCVSId

```
public static java.lang.String getCVSId()
```

Returns the CVS Id of the file. To make sure this is correct, the servlet should be commited to CVS before it is compiled and deployed.

---

doPost

```
public void doPost(javax.servlet.http.HttpServletRequest request,
                   javax.servlet.http.HttpServletResponse response)
            throws javax.servlet.ServletException,
                   java.io.IOException
```

Handles HTTP Post requests, treats them equal to GET requests.

Overrides:

`doPost` in class `javax.servlet.http.HttpServlet`

---

doGet

```
public void doGet(javax.servlet.http.HttpServletRequest request,
                  javax.servlet.http.HttpServletResponse response)
           throws javax.servlet.ServletException,
                  java.io.IOException
```

Handles HTTP Get requests with a request.

Overrides:

`doGet` in class `javax.servlet.http.HttpServlet`

no.inspera.applications.publish.math

Class MathException

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--no.inspera.services.exception.BaseException
                    |
                    +--no.inspera.applications.publish.math.MathException
```

All Implemented Interfaces:

java.io.Serializable

---

public class **MathException**
extends BaseException

This exception indicates a general type of error state in the math part of the publish-application, might be subclassed for more specific uses.

Via the superclass it provides localized error message and centralized logging.

Version:

$Revision: 1.2 $ $Date: 2003/04/14 14:02:22 $

Constructor Summary

MathException(java.lang.String messageId, java.util.Locale locale, java.lang.Class originator) Creates a new MathException.
MathException(java.lang.String messageId, java.util.Locale locale, java.lang.Class originator, java.lang.Exception e) Creates a new MathException with the originator e as the previous exception in the chain.

Methods inherited from class no.inspera.services.exception.BaseException
D:\diplom-CD\javadoc\no\inspera\services\exception\BaseException.html - <clinit>(), getLocalizedMessage, getMembersXML, getMessageId, getUserErrorMessage, getUserErrorMessage, getXML, init, printStackTrace, printStackTrace, printStackTrace, sendMail, setName

Constructor Detail

MathException

```
public MathException(java.lang.String messageId,
                     java.util.Locale locale,
                     java.lang.Class originator)
```

Creates a new MathException.

Parameters:

`messageId` - A key to a property-file which will store localized versions of the message to show the user, affects the output from getUserErrorMessage().

`locale` - The Locale to get the error message to show end user

`originator` - A Class-object representing the originator of the error/exception, used for logging and tracing

---

MathException

```
public MathException(java.lang.String messageId,
                     java.util.Locale locale,
                     java.lang.Class originator,
                     java.lang.Exception e)
```

Creates a new MathException with the originator e as the previous exception in the chain.

Parameters:

`messageId` - A key to a property-file which will store localized versions of the message to show the user, affects the output from getUserErrorMessage().

`locale` - The Locale to get the error message to show end user

`originator` - A Class-object representing the originator of the error/exception, used for logging and tracing

Appendiks C – Elektroniske eksempler fra graftegneren

Enkelte eksempler på bruk av graftegneren, slik den er integrert i Inspera sitt Content Management-system, er eksportert til rene HTML-filer og tatt med på vedlagte CD-ROM i katalogen med navn ”graphdrawer_samples”. Dette er gjort slik at det er mulig å se graftegneren i praktisk bruk bare man har en nettleser med javamotor, uten å teste hele løsningen.

Det som ligger på CD-ROMen er altså sluttresultatet etter en XSL-transformasjon og en traversering av content-delen av MathML-uttrykket, samt java-appleten som gir visse dynamiske muligheter.

Følgende eksempler finnes i katalogen:

- advanced_gui.html – et eksempel på et uttrykk satt inn i et XML-dokument med guiMode satt til ”advanced”, noe som resulterer i at sluttbrukeren får mulighet til å zoome, markere min/max, skru av/på akser, sette visningsfarge for grafer, m.v.
- animation.html – et eksempel på en animasjon laget via graftegneren (guiMode=”animation”), med bakgrunnsbilde og bilde som beveger seg langs grafen. Grafen som animeres er satt sammen av to funksjonsuttrykk med ulike definisjonsmengder, slik at denne funksjonaliteten også blir demonstrert indirekte.
- plain.html – et enkelt eksempel der brukeren bare får presentert grafen som en del av et dokument (guiMode=”none”), men ikke får noe grensesnitt for å manipulere grafen (men grafen tegnes i klienten i eksemplet, det er ikke en ferdiggenerert bildefil)
- parameters_animation.html – et eksempel med animasjon av grafen til en rett linje der brukeren kan styre parametrene for stigningstall og skjæring med y-aksen

For alle eksemplene har jeg også lagt med en fil med endelse ”.xml” som er utdraget fra XML-innholdet siden ble generert fra, der der finnes referanser til matematiske objekter iht. DTDen som tidligere er presentert.

Dersom man har mulighet til å kjøre en J2EE-plattform, men ikke Insperas, kan man også eksperimentere i graftegneren direkte med filen GraphDrawerController.jsp som tidligere er omtalt. For brukervennlig inntasting trenger man Design Science sin WebEq-redigeringskomponent, men den er ikke med på CD-ROMen av lisensieringsgrunner. Men man kan ta content-MathML fra andre programmer (eller skrive dette for hånd, dersom man har masochistiske tendenser) for å tegne opp grafen til disse dersom man ikke ønsker å laste ned en evalueringsversjon fra Design Science.

Appendiks D – Testresultater fra ytelsestest

Testoppsett og fulle resultater finnes som nevnt på vedlagte CD-ROM i katalogen ”stresstest”, her gjengis kun tallene som ligger til grunn for grafene som er brukt i teksten.

For fullt utbytte av resultatene (med hierarkiske menyer og fulle detaljer om testgjennomføringene, velg ”reports”) behøves det tidligere nevnte testverktøyet jeg har benyttet, men testoppsettet er også forklart kort i en tekstfil med navn ”setup_explanation_plaintext.txt” i samme katalog på CDen. På CDen finnes også installasjonsprogrammet til stresstest-verktøyet.

Distribuert modell

Gjennomstrømning/throughput

Antall brukere (*N*)	1	10	50	75	150	300	500	1000	2000
Forespørsler per sekund	0,0300	0,290	1,41	2,06	4,06	8,35	14,1	28,4	58,7

Antall brukere (*N*)	4000	6000	10000	20000	25000
Forespørsler per sekund	124	181	323	528	475

Maksimal responstid

Antall brukere (*N*)	1	10	50	75	150	300	500	1000	2000
Millisekunder	15,4	33,5	26,3	32,0	28,8	279	39,0	33,0	23,4

Antall brukere (*N*)	4000	6000	10000	20000	25000
Millisekunder	45,9	546	1186	(*)	9079

(*) Rapportert maksimal responstid fra testverktøyet var lav, men siden antallet tilkoblingsfeil (socket errors, av typen connect, var 12000) var spesielt høyt er i praksis ikke den maksimale verdien for de som faktisk fikk svar så relevant. Tilsvarende er det også for 25000 brukere så mange socket errors at det konkrete tallet på 9 sekunder kan være villedende, siden mange brukere ikke får svar i det hele tatt.

Ren serverside-basert modell

Gjennomstrømning/throughput

Antall brukere (*N*)	1	10	25	50	60	75	100	130	200
Forespørsler per sekund	0,0300	0,330	0,62	1,22	1,37	1,37	2,02	1,81	1,95

Maksimal responstid

Antall brukere (*N*)	1	10	25	50	60	75	100	130	200
Millisekunder	437	888	807	4076	5500	11883	16042	11483 (*)	18823

(*) Også her er tallene i den øvre enden noe villedende fordi ikke alle forespørsler blir besvart.

For tolking av tallene og kommentarer rundt de to modellene vises det til selve teksten i rapporten.