Norges miljø- og biovitenskapelige universitet

Norges miljø- og biovitenskapelige universitet
Fakultet for miljøvitenskap og teknologi
Institutt for matematiske realfag og teknologi

Masteroppgave 2016
30 stp

Kontroll av geodata på GML-format

Control of geodata on GML-format

Henrik Gulliksen Schüller
Geomatikk

Forord

Denne mastergradsoppgaven markerer avslutningen på mitt studium innen Geomatikk ved Norges miljø- og biovitenskapelige universitet (NMBU). Oppgaven har omfang på 30 studiepoeng og er skrevet våren 2016 ved Institutt for matematiske realfag og teknologi, IMT.

Aller først vil jeg takke min hovedveileder Håvard Tveite som jeg har hatt mange konstruktive samtaler med i løpet av mastergradsarbeidet mitt. Hans faglige bidrag ga meg en stø kurs i arbeidet, og samtidig har han gitt meg gode programmeringstips.

Jeg vil takke Kartverket som har hatt et ønske om å løse og komme videre i prosjektet med tilnærming til GML-format. I oppgaven har de bidratt med veiledning og tilgang til datamateriale. Jeg takker min biveileder i Kartverket, Andreas Røstad, som har gitt meg tilgang til litteratur og relevant informasjon om temaet.

Ås, 08. Mai 2016

Henrik Gulliksen Schüller

Sammendrag

Denne masteroppgaven har som hovedmål å vise hvordan programmet SOSI-kontroll, et program for kontroll av geodataformater, kan moderniseres for å også kunne kvalitetssikre og kontrollere geodata på GML-formatet. Arbeidet går igjennom hvilken metode som egner seg best for at denne kontrollen skal bli gjennomført på en hensiktsmessig måte.

Utgangspunktet for oppgaven er de tre trinnene for kontroll av XML-dokument som er kjent: syntaks, skjema og schematronkontroll. Det viste seg at disse tre trinnene ikke er tilstrekkelig kontroll av formatet.

I oppgaven ble det undersøkt om det eksisterer funksjonalitet fra SOSI-kontroll, som er ønskelig å implementere i et program som skal kontrollere GML-formatet.

Det vises at det er et behov for at syntakskontrollen blir videreutviklet. Denne kontrollen undersøker om GML-dokumenter og schematronskjema er på velformet XML og om innholdet i disse filene og applikasjonsskjema følger spesifikasjoner.

Kontrollen av geometri på skjemanivå, viste seg å ikke være godt nok. Det er nødvendig med en ekstra kontroll av geometri. Den implementerte kontrollen henter ut all geometri fra GML-dokumentet og så er det opp til brukeren hva han ønsker å undersøke eller kontrollere.

Et resultat av masterarbeidet viser at det er ønskelig med en mer detaljert rapportering. I tillegg til å rapportere informasjon fra syntaks, skjema og schematronkontrollen, så vil rapporten også inkludere en statistikk-, kontroll- og kvalitetsrapport.

En målsetning var å finne ut hva programvare for kontroll av GML må gi brukeren informasjon om. Etter å ha lest denne oppgaven vil en utvikler av et slikt program ha funnet svaret på det.

Abstract

The main goal of the thesis is to show how the computer program SOSI-kontroll, that controls geodata formats, can be modernized so that it can also handle the GML-format. The study evaluates possible methods that are best suited for performing this control in an expedient way.

The thesis is based on the three steps of how to control XML-documents: syntax valdiation, schema validation and schematron validation. It turns out that these steps are not sufficient for an adequate control of the format.

It was investigated whether there exists validation software for other geodata formats (see SOSI-control), and other functionality that will be desirable to implement in software for verification of the GML-format.

It is seen that there is a need for an improved syntax check. This check examines whether GML-documents, applicationschema and schematronschema are written in well-formed XML and that the content of the files follow a set of given specifications.

It also turns out that the verification of geometry, which is done during the schema validation, is not sufficient. It is therefore necessary to implement an additional control of geometry. This control retrieves the geometry from the GML-documents and gives the user options for what they want to examine or verify.

The results from this thesis show that it is desirable to report the results in more detail. In addition to reporting information from the syntax validation, schema validation and schematron validation, the report should also include a statistics-, control- and quality-report.

A goal was to determine what software for control of GML must provide the user information about. A developer of such programs should have the answer to that after reading this thesis.

Innhold

Forkortelser

XML	Extensible Markup Language
GML	Geography Markup Language
SOSI	Samordnet Opplegg for Stedfestet Informasjon
UML	Unified Modeling Language
OGC	Open Geospatial Consortium
ISO	International Organization for Standardization
DTD	Document Type Definition
XSD	XML Schema Definition
Xlink	XML Linking Language
GIS	Geografiske informasjonssystemer

Figurliste

1. Innledning

Utviklingen av digitale kart har vært økende de siste to generasjonene. Nøyaktighet i kartmaterialet og bruk av internasjonale standarder er snart en selvfølge for alle som skal ferdes på land, hav eller i luftrommet. Ved å benytte de samme standarder over landegrensene øker vi både nøyaktighet og sikkerhet, men det krever et omfattende arbeid med å kvalitetssikre og kontrollere nye formater når gamle byttes ut.

Samordnet Opplegg for Stedfestet Informasjon (SOSI) (Kartverket, 2012) er både en standard og det dominerende geodataformatet for å utveksle digitale kart og infrastruktur i Norge. Det har blitt utviklet av Kartverket (tidl. Statens kartverk). Formatet har vært kontinuerlig revidert og strategisk er det ønskelig at formatet tilnærmer seg internasjonale standarder. Det er bestemt at den normative ISO 19136 (ISO, 2007), standarden for GML-formatet, skal ta over for SOSI-formatet (Kartverket, 2015b).

1.1 Problemstilling

GML skal tas i bruk som utvekslingsformat for geodata i SOSI-standarden, og det er sterkt behov for å modernisere programmet SOSI-kontroll (Kartverket, 2010) til å kunne kontrollere GML-formatet. Av den grunn er det ønskelig å undersøke hva som blir håndtert ved syntakskontroll av formatet, og hva som kan kontrolleres med standard XML/GML skjema og schematronkontroll.

Ved modernisering av SOSI-kontroll vil det også bli undersøkt om det er eksisterende funksjonalitet som bør videreføres.

Ved å finne hvilken funksjonalitet et program som skal kontrollere SOSI-data på GML-formatet trenger, så er det mulig å finne svaret på: «Hvordan skal GML-formatet kvalitetssikres og kontrolleres med utgangspunkt i SOSI-standarden?»

1.2 Litteratur

Det var vanskelig å finne litteratur på kontroll/validering av GML-formatet. Relevante funn var hovedsakelig artikler (Ahn, Park, Yoo, & Bae, 2004; Chao, Xiao, & Zhang, 2008; Detlev Wagner, 2014; Gao et al., 2013; Hu, 2010; Hugo, 2014; Klimek, Benda, & Necasky, 2014; Lake, 2005; LAN, LU, ZHANG, & Jiang, 2005; Maly & Necasky, 2015; Min, 2010; Myer, 2005; van den Brink, Stoter, & Zlatanova, 2013), og de er ikke direkte om kvalitetssikring og kontroll av formatet, men informasjon om XML/GML. Den mest relevante litteraturen som ble funnet på temaet var standardene til ISO (19100-serien) (ISO, 2003, 2006, 2007, 2009, 2011, 2013, 2014, 2015) og til OGC (OGC, 2007a, 2007b, 2011, 2012a, 2012b, 2012c, 2016).

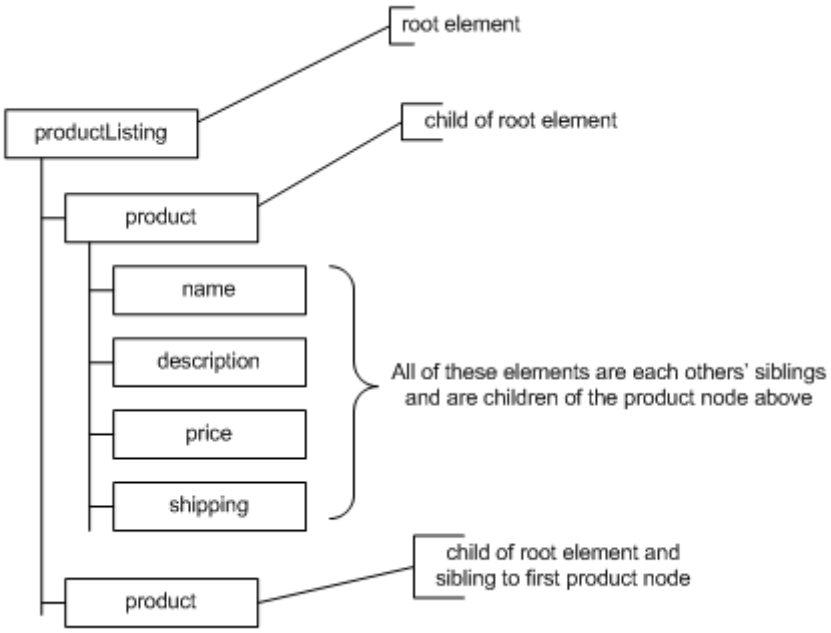
Kartverket (Kartverket, 2007, 2010, 2012, 2014, 2015a) står bak mye god og relevant informasjon på norsk. Dette var blant annet informasjon om SOSI, generell informasjon om kontroll av geodata og om geodatakvalitet.

Norge Digitalt (Norge-digitalt, 2015) har laget en god og grunnleggende «Veileder for GML» på norsk. På engelsk er Inspire-direktivet (INSPIRE, 2010) sitt arbeid med GML en enkel, grunnleggende og relevant informasjonskilde.

2. Bakgrunn og teori

2.1 Extensible Markup Language (XML)

XML er et markeringsspråk som definerer et sett regler for å kode strukturerte data på et format som både er leselig for mennesker og for maskiner, og fungerer som et verktøy for deling mellom informasjonssystemer. Det blir brukt som kommunikasjonsmiddel mellom ulike dataformater, kommunikasjonssystemer og til koding av dokumenter. Det er viktig at strukturen og innholdet i et XML-dokument er korrekt for at den skal bli håndtert effektivt. Av den grunn er det en rekke syntaksregler for at dokumentene skal kunne bli kalt «velformet». Dersom et XML-dokument ikke overholder disse kravene, så blir det ikke sett på som XML. En prosessering av en slik fil vil melde ifra om feil og avslutte prosessen. Figur 1 viser den logiske strukturen i et XML-dokument.



Figur 1: Den logiske strukturen i et XML-dokument (Myer, 2005)

I tillegg til å være «velformet» (*well-formed*) kan et XML-dokument være «validert» (*valid*). Dette betyr at dokumentet inneholder en referanse til *Document Type Definition* (DTD), og elementene og attributtene følger da det syntaks som er spesifisert i DTD.

I senere tid har det kommet en annen metode, XML-skjema (XSD-format[1]), for å definere regler for hva som kan og ikke kan eksistere i et XML-dokument. Denne metoden gir en mye bedre beskrivelse enn DTD. Skjemaene gir dataprogrammer muligheten til å validere data og gir et rammeverk for å kunne strukturere data.

Xlink er en generell XML-mekanisme som blir mye brukt. Denne metoden brukes for å linke mellom elementer i XML-dokumenter eller internt i samme dokument. Xlink fungerer godt

[1] XML Schema Definition

for å referere til maskinlesbare registre hvor det finnes definisjoner av kodeverdier, og det fungerer bra i forbindelse med delt geometri i GML sammenheng.

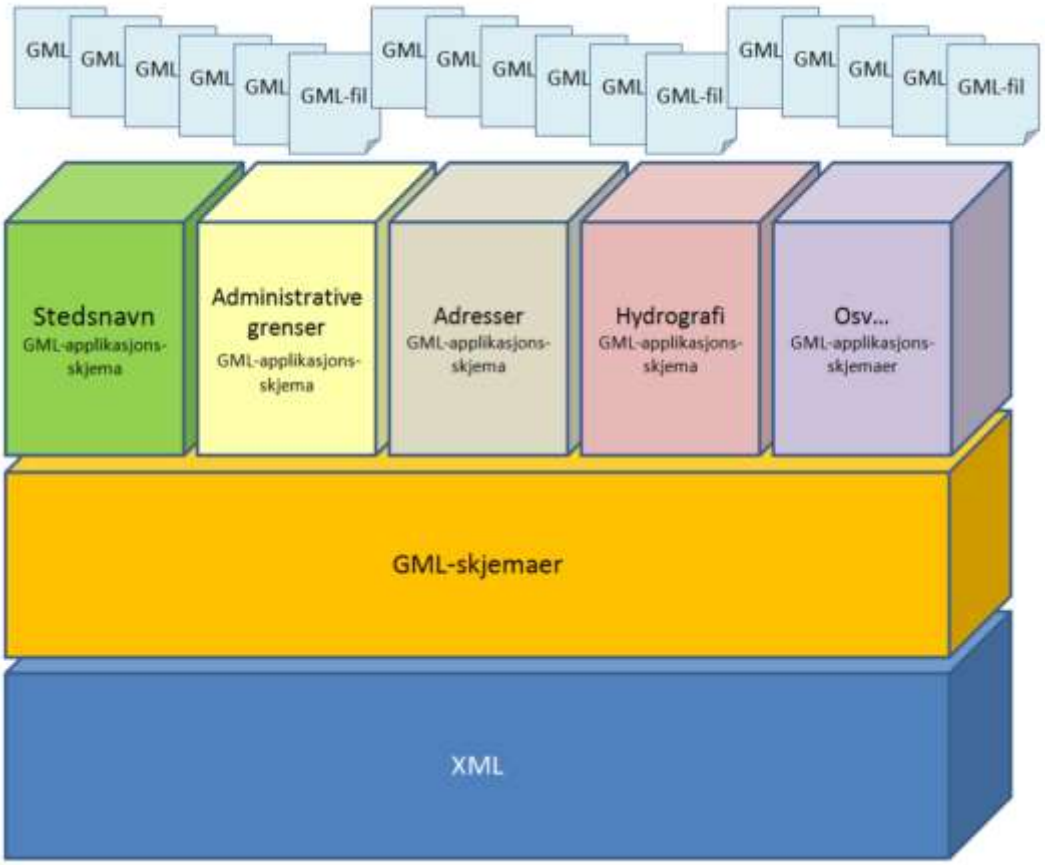
2.2 Geography Markup Language (GML)

GML er et XML-basert format for utveksling av geografisk informasjon. Hovedkonseptene brukt i GML for å modellere verden er tatt fra ISO 19100 serien av internasjonale standarder og OGC sin abstrakte spesifikasjon (OGC, 2007a).

GML består av 3 hoveddeler:

- Generelle regelbeskrivelser (GML-skjema)
- Avanserte regelbeskrivelser (GML-applikasjonsskjema)
- Datadokument (GML-dokument)

Ut ifra de generelle regelbeskrivelsene (GML-skjema) for GML kan man lage enkle GML-dokumenter som inneholder geografiske objekter som punkter, linjer, polygoner eller annen grunnleggende geometri. Dersom man ønsker å lage mere komplekse geografiske objekttyper, så kan dette gjøres ved å lage GML-applikasjonsskjema. I et applikasjonsskjema defineres nye geografiske objekttyper med utgangspunkt i objekttyper fra GML-skjema. Ved bruk av GML-applikasjonsskjema, så kan man vite sikkert at datastrukturen til GML-dokumenter laget i samsvar med skjemaene blir ivaretatt. Grunnsteinene i SOSI-GML vises i Figur 2.



Figur 2: Grunnsteinene i SOSI-GML (Norge-digitalt, 2015)

Noen fordeler med GML:

- Offisiell internasjonal standard
- Direkte basert på UML-modeller
- Tekstbasert språk med god lesbarhet for menneske og datamaskin (hierarkisk struktur)
- Enkel validering av dokument (standard XML)
- God realisering av komplekse modeller
- XML er lett å transformere ved bruk av de fleste programmeringsspråk
- Effektivisering av informasjonsflyt

2.2.1 GML-skjema

Et GML-skjema omfatter XML-komponenter, attributter, enkle typer, komplekse typer og attributtgrupper, og er en samling XSD-beskrivelser av generelle geografiske objekter med tilhørende egenskaper. Skjemaene er beskrevet i OGC sin internasjonale standard (OGC, 2007a) og blir opprettet, endret og forvaltet av OGC. XML-kodingen for GML-skjemaer er i samsvar med ISO 19118 (ISO, 2011). Skjemaene inneholder de mest grunnleggende geografiske objekttypene som brukes i GML, og mange av disse samsvarer med de som finnes i SOSI-formatet. OGC har laget en rekke GML-skjema. Veileder for GML-formatet (Norge-digitalt, 2015) viser følgende oversikt:

- *Objekttyper*
- *Geometriske primitiver*
- *Topologi*
- *Tidsaspektet (temporal)*
- *Definisjoner og kataloger*
- *Enhet/mål og verdier*
- *Koordinatreferansesystem*
- *Retning*
- *Observasjoner*
- *Coverages (gjelder fra GML 3.3)*

2.2.2 GML-applikasjonsskjema

Et GML-applikasjonsskjema er XML-beskrivelser av objekttyper. Skjemaene er som regel basert på en UML-datamodell (Unified Modelling Language) (ISO, 2012), og blir opprettet, endret og forvaltet av ulike organisasjoner, ut ifra egne behov. Et applikasjonsskjema (XSD) er ikke XML, men blir beskrevet ved hjelp av XML-skjema. Et GML-applikasjonsskjema skal referere til et eller flere skjema hvor det finnes informasjon om grunnleggende geografiske objekttyper. Det er mulig at et GML-applikasjonsskjema refererer til et eller flere andre applikasjonsskjema.

GML-applikasjonsskjema kan lages på to alternative måter som er støttet av OGC sin internasjonale standard (OGC, 2007a):

- Ved å følge reglene i ISO 19109 (ISO, 2015) for applikasjonsskjemaer i UML. Dette må gjøres i samsvar med både hvilke begrensinger slike skjemaer har og regler for GML som er angitt i standarden
- Ved å følge reglene for GML-applikasjonsskjemaer som er gitt i OGC sin standard for å lage applikasjonsskjemaene direkte i et XML-skjema

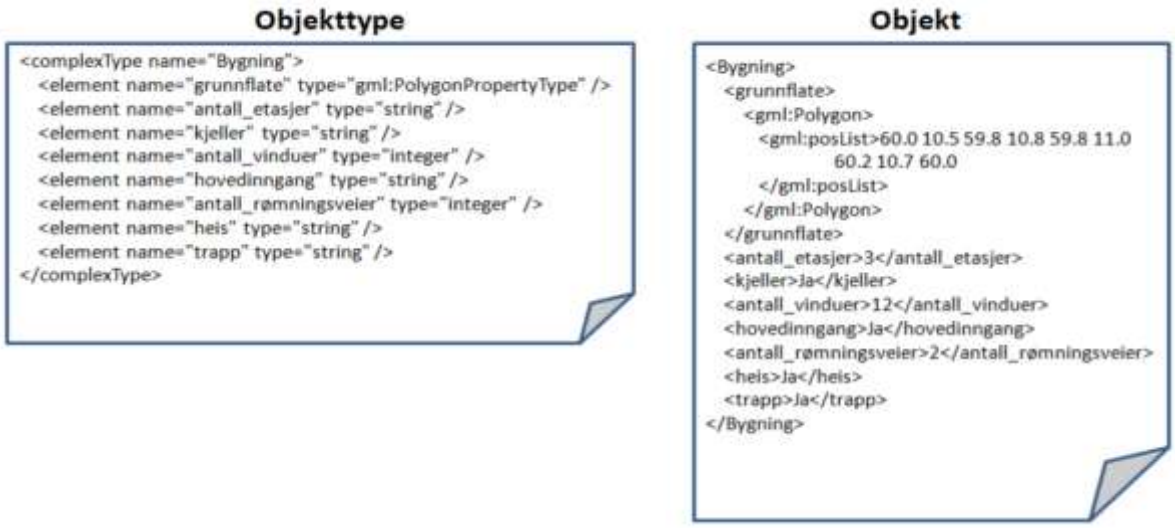
Det forventes at alle applikasjonsskjema modelleres i samsvar med General Feature Model som er spesifisert i ISO 19109 (ISO, 2015).

Et generelt GML-applikasjonsskjema skal inneholde:

- Navnerom (URL) og plassering for GML-skjema, eventuelt GML-skjemaer
- Referanser til andre GML-applikasjonsskjema, dersom flere brukes
- Referanser til objekttyper i GML-skjema/applikasjonsskjema som det har blitt referert til
- Definisjoner av objekttyper
- Angivelse av interne kodelister
- Referanser til eventuelle eksterne kodelister

GML-applikasjonsskjemaer blir oftest håndtert av programvare, så brukeren trenger ikke å forholde seg til applikasjonsskjemaene om man kun skal åpne et GML-dokument eller se på data i et GIS-program.

Objekttypene som ønskes brukt defineres i et GML-applikasjonsskjema. Disse objekttypene bestemmer datastrukturen og setter krav til innholdet i objektene. Objektene inneholder verdier for de definerte objekttypene og kan ses på som data bærere. Se Figur 3.



Figur 3: Definisjon av objekttype med tilhørende objekt (Norge-digitalt, 2015)

2.2.3 GML-dokument

GML-objektene blir angitt i GML-dokumentet, og det er denne informasjonen som kan vises i GIS-programvare som kartbilder. Disse dokumentene inneholder referanser til GML-applikasjonsskjemaene og til andre filer som skal brukes.

Et GML-dokument er bygget opp på denne måten (Norge-digitalt, 2015):

- Tildeling av navnerom og referanser til eventuelle GML-applikasjonsskjema
- Opplisting av objekter med tilhørende egenskaper

I GML-dokumenter er det mulig å referere til XML-elementer ved bruk av Xlink. Dette er tidligere nevnt i delen om XML og er en metode som brukes for å lage interne og eksterne linker i XML-dokumenter. Når Xlink brukes for å lage referanser i GML-dokumenter, er det alltid GML-id som vil bli brukt som identifikasjon av objektet det skal pekes til.

Det er hovedsakelig tre måter å bruke Xlink referanser på i et dokument:

- Internt i det samme dokumentet
- Eksternt til et annet dokument, men lokalt i samme katalog på datamaskinen
- Eksternt til et annet dokument som ikke er på samme datamaskin (f.eks. over internett)

Selv om Xlink kan brukes til å lage eksterne referanser, så anbefales det ikke.

2.2.4 GML-profiler

GML-profiler er ikke en av GML sine 3 hoveddeler, men er begrensninger for GML som utrykkes i et dokument eller i et skjema. Profilene brukes for å forenkle bruken av GML for bestemte formål og kan brukes for å legge til rette for økt bruk av GML-formatet.

Når det gjelder profiler, så er det viktig å vite at det ikke er det samme som GML-applikasjonsskjema. GML-applikasjonsskjemaer er en modell av virkeligheten som blir uttrykt i komplett GML eller innenfor en GML-profil. En GML-profil er en direkte begrensing av GML.

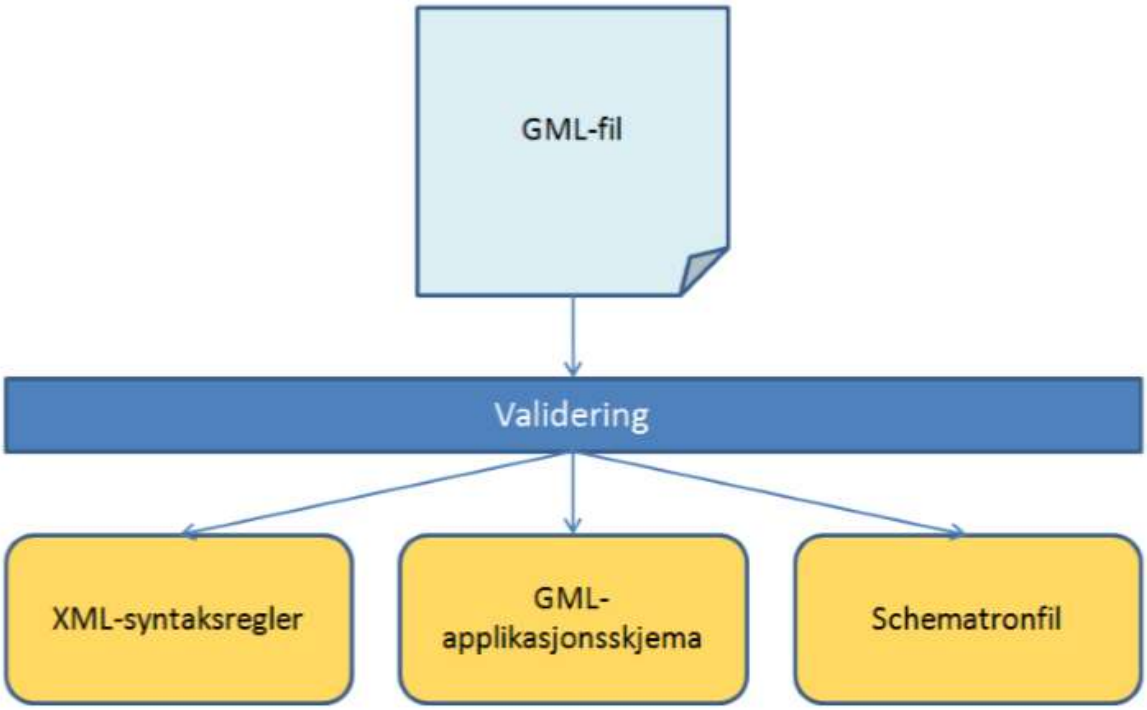
2.3 Kontroll av GML

For å kontrollere GML-data, må man se på hvordan XML-dokumenter blir validert. Validering av XML-dokumenter, deles i 3 kvalitetsnivåer (se Figur 4 for illustrasjon):

- Syntakskontroll – Velformet XML (*well-formed*)
- Skjemakontroll
- Schematronkontroll

Ved kontroll av GML-dokumenter vil det alltid være nødvendig å undersøke om filen er skrevet på velformet XML for å vite om syntaksreglene overholdes og at dokumentet kan bli sett på som XML. Det er nødvendig å gjennomføre en skjemakontroll for å sjekke at

objektene i dokumentet er definert i et skjema. I noen tilfeller vil det være nødvendig med en ekstra kontroll dersom det er spesielle krav eller spesifikasjoner som skal overholdes. Tilleggskrav som ikke blir dekket av syntakskontroll eller skjemakontroll vil i noen tilfeller kunne bli dekket av en schematronkontroll.



Figur 4: Valideringstypene som ivaretar de 3 kvalitetsnivåene for GML-dokumenter (Norge-digitalt, 2015)

2.3.1 Syntakskontroll - Velformet XML

For å bestemme at et XML-dokument er velformet, gjennomføres en enkel syntakskontroll som sjekker om innholdet i dokumentet er i henhold til XML sine generelle syntakskrav. Det vanligste som blir undersøkt i en slik kontroll er at innholdet:

- i dokumentet er definert
- er avgrenset av en start og slutt-tag
- er riktig nøstet (parents within roots, children within parents)
- består av riktig Unicode-tegn

og det har:

- ingen bruk av spesielle syntakstegn som < og &
- en start, slutt og «tomt-element» tagger som er riktig nøstet, ingen mangler eller overlapper

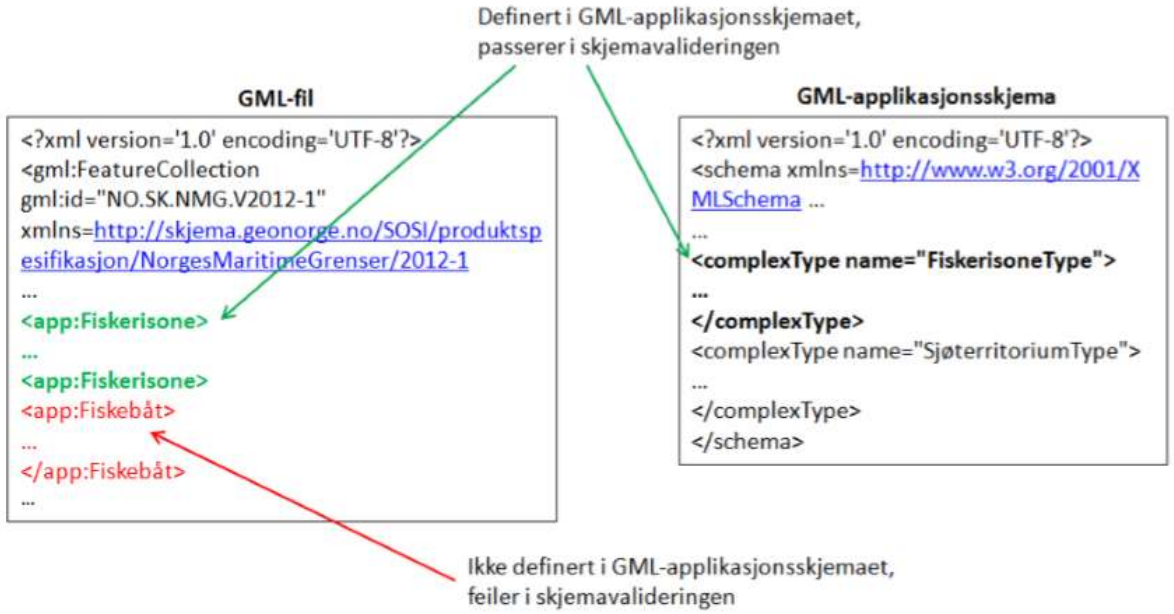
- elementkoder som er case-sensitive, så start/slutt-tagger må skrives likt. Kodene kan heller ikke inneholde tegn som: !"#$%&'()*+,/;<=>?@[\]^`~, og kan heller ikke inneholde mellomrom eller starte med -,, eller et tall
- et «root» element som inneholder alle de andre elementene i dokumentet

Det kreves ikke at dokumentet refererer til et GML-applikasjonsskjema, og det gjennomføres heller ingen syntakskontroll som sjekker skjema som det refereres til.

2.3.2 Skjemakontroll

En skjemakontroll gjennomføres for å kontrollere innholdet i et GML-dokument. Denne kontrollen undersøker om objektene i dokumentet er definert i et GML-applikasjonsskjema og de tilhørende GML-skjemaene. Her vil både verdier og elementer bli sjekket opp mot det som er definert som gyldige kombinasjoner i de gjeldende skjemaene, se Figur 5.

Siden GML også er XML, så kan skjemavalideringsprogrammer for XML brukes til å validere GML.



Figur 5: Hovedprinsippet i skjemavalidering (Norge-digitalt, 2015)

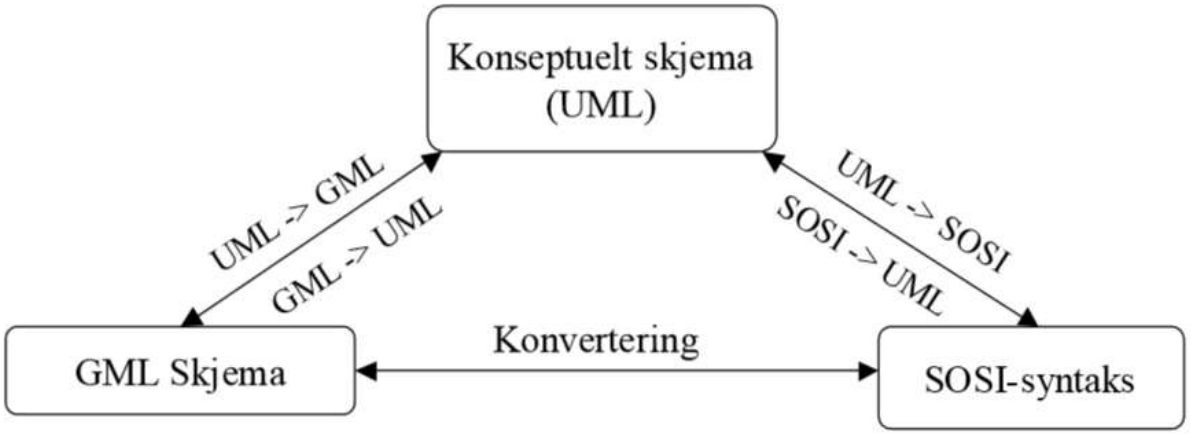
2.3.3 Schematronkontroll

Schematron er et XML-basert språk og en internasjonal standard (ISO/IEC 19757) (ISO, 2006) som brukes for å kontrollere avanserte forhold mellom elementer i et XML-dokument. Språket kan gi ekstra begrensinger ut over det GML-skjema og applikasjonsskjema har muligheten til. Ved å bruke schematron er det for eksempel mulig å be om eller kreve at bestemte elementer i dokumentet skal ha bestemte attributter eller å angi nødvendige relasjoner mellom XML-dokumenter.

For at en schematronkontroll skal gjennomføres automatisk av en datamaskin, må schematronskjema bli referert til i GML-dokumentet på samme måte som en referer til et applikasjonsskjema. Denne referansen kan lages ved å bruke et element av formen «xml-model» som fungerer som en prosesseringsinstruksjon (W3C, 2012).

2.4 Unified Modeling Language (UML)

UML er en industristandard for datarelatert modellering. Modelleringsspråket gir en standard måte å visualisere utformingen av et system og er det modelleringsspråket som brukes ved modellering av konseptuelle skjemaer for både GML og SOSI. Figur 6 viser realiseringen av slike modeller for SOSI og GML.



Figur 6: Realisering av modeller i form av SOSI og GML (Kartverket, 2012)

UML tilbyr en måte å visualisere et systems arkitektur i et diagram og kan inkludere følgende elementer:

- Eventuelle aktiviteter (jobber)
- De enkelte komponentene i systemet
    - Hvordan de kan kommunisere med andre programvarekomponenter
- Hvordan systemet vil kjøres
- Hvordan deler av modellen kommuniserer med andre (komponenter og grensesnitt)
- Eksterne brukergrensesnitt

Modelleringsspråket var i utgangspunktet kun ment for dokumentasjon av objektorientert design, men er senere utvidet for å dekke et større sett av prosjektdokumentasjon og har blitt et nyttig verktøy i mange sammenhenger.

Det er viktig å skille mellom UML-modellen og et sett av diagrammer. Et diagram er en delvis grafisk representasjon av et system sin modell og et sett med diagrammer trenger ikke nødvendigvis å dekke hele modellen.

For eksempler på UML anbefales det å sjekke ut «City GML» (OGC, 2012a).

2.5 Samordnet Opplegg for Stedfestet Informasjon (SOSI)

SOSI (Kartverket, 2012) er både en standard og et filformat for utveksling av geografisk informasjon som benyttes i Norge. Standarden utgitt første gang i 1987, og har kontinuerlig blitt revidert og videreutviklet. SOSI sin utvikling har vært sterkt knyttet opp mot de internasjonale standardene i ISO 19100-serien.

Hovedstrukturen i SOSI-filer består av:

- Innledende opplysninger (.HODE)
- Brukerstyrte definisjoner (.DEF) - valgfri
- Objekttypedefinisjoner (.OBJDEF) - valgfri
- Datagrupper, samt eventuelle beskrivelsesgrupper (selve dataene)
- Avslutning (.SLUTT)

Hodet (.HODE) inneholder generelle opplysninger for hele fila. For eksempel informasjon om koordinatsystemet, dekningsområde og egenskapsinformasjon som gjelder elementene i filen. Dersom en ønsker å definere spesielle egenskapsnavn som ikke er definert i den offisielle SOSI-standarden, så kan dette gjøres i delen for brukerstyrte definisjoner (.DEF). Videre i strukturen til SOSI-formatet finnes definisjoner av objekttyper (.OBJDEF) med sine tilhørende egenskaper og forhold. Disse objekttypene er standardiserte definisjoner tilhørende SOSI's sin generelle objektkatalog.

2.6 SOSI-Kontroll

SOSI-kontroll er et brukerprogram (Kartverket, 2016) som brukes for å kvalitetskontrollere SOSI-filer mot en produktdefinisjon. Programmet SOSI-kontroll startet som et studentprosjekt ved høgskolen i Gjøvik i 1992, kalt KVAKK (Kartverket, 1999). Videre arbeid med prosjektet har vært organisert av Kartverket (tidl. Statens Kartverk). Grunnet formatet sine avanserte muligheter og frie form, så inneholder SOSI-filer ofte feil, noe som medfører et behov for kvalitetssikring. Det er også viktig for leverandører og kjøpere av kartdata å ha muligheten til å kontrollere produktene ved kjøp eller salg (Kartverket, 2010).

SOSI-kontroll deles inn i 7 kontrollgrupperinger:

1. Formell formatsjekk
2. Gruppevis innholdssjekk
3. Knutepunkt kontroll
4. Flate-kontroll
5. Objekt-kontroll
6. Statistikk
7. Rapportering

Mer detaljert informasjon om disse punktene finnes i Vedlegg B – SOSI-kontroll og i SOSI-kontrollen sitt hjelpdokument (Kartverket, 2010).

3. Programfunksjonalitet og programmering

I denne masteroppgaven har det blitt sett funksjonalitet fra programmet SOSI-kontroll som bør videreføres for å kunne kontrollere GML-formatet basert på SOSI-standarden. Det har også blitt utviklet et program som kontrollerer deler av det som er nødvendig. Måten dette er gjort på er beskrevet i dette kapittelet og programmet finnes i Vedlegg G – Python kode. Tester av dette programmet med eksempeldata finnes i resultatkapitellet.

3.1 Videreføring av funksjonalitet fra programmet SOSI-kontroll

Programmet SOSI-kontroll har vært igjennom mange oppdatering og endringer siden det først ble laget, og inneholder nå mye av det som burde være med i programvare for kontroll av geodata.

En del av kontrollene programmet gjennomfører går på karakteristikk som gjelder for SOSI-formatet og vil ikke ha noen hensikt å implementere i en kontroll av GML-formatet. Eksempler på dette er SOSI-nivå (kontrollgruppe 2.3) og knutepunktskontroll (kontrollgruppe 3). SOSI-kontrollen inneholder funksjonalitet som ikke vil være relevant å implementere direkte i en GML-kontroll, men det finnes også funksjonalitet som burde være med.

Funksjonalitet, som allerede eksisterer i programmet SOSI-kontroll og som burde være med i en SOSI-basert kontroll av GML er:

- Syntakssjekk og formatsjekk (dekkes av syntakskontroll - velformet XML og skjemakontroll)
- Innholdssjekk
- Geometrikontroll
- Kontrollrapport
- Statistikkrapport

Syntaks- og formatsjekk dekkes av syntakskontroll – velformet XML og av skjemakontroll, så dette implementeres istedenfor å lage nye kontroller som gjør akkurat det samme.

Innholdssjekk av SOSI-formatet sjekker annen informasjon i filen som ikke blir dekket av syntaks- eller formatsjekken. Dette er funksjonalitet som kontroll av GML-formatet trenger og må da sjekke viktig innhold i GML-dokumenter, applikasjonsskjema og schematronskjema. I oppgaven blir denne sjekken omtalt som en videreutviklet syntakskontroll.

En geometrikontroll henter ut og kontrollerer geometrien i filen, og hva som kontrolleres kan avhenge av brukerens ønsker. I SOSI-kontroll gjennomføres en kontroll av geometri i trinnene flate- og objektkontroll. En tilsvarende kontroll er nødvendig å implementere i en SOSI-basert kontroll av GML-formatet.

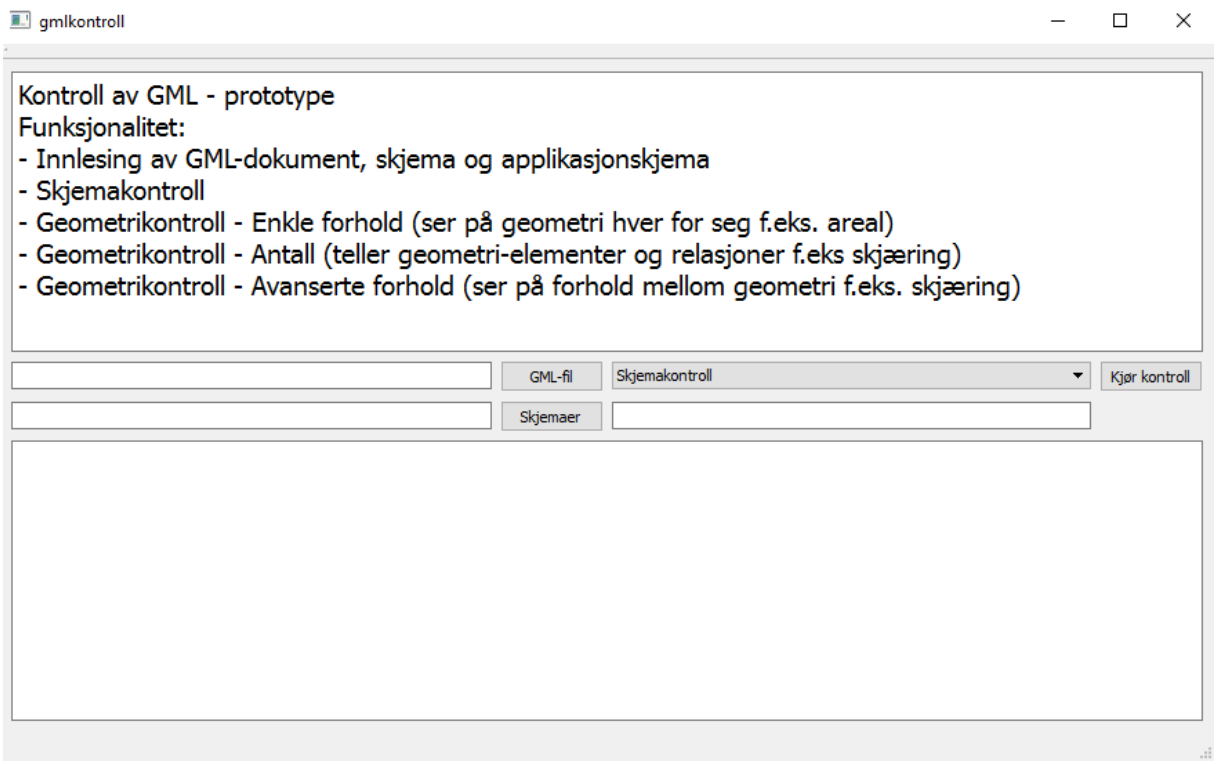
SOSI-kontroll gir brukeren relevant informasjon etter gjennomført kontroll i form av en kontroll- og en statistikkrapport. Her finnes all informasjon fra kontrollene som er interessant, og er noe som burde være med ved kontroll av all geodata.

3.2 Beskrivelse av program for kontroll av GML

Programmet som ble utviklet er skrevet i programmeringsspråket Python versjon 2.7 og det ble valgt å bruke funksjonsbiblioteker for å unngå unødvendig programmering av funksjoner (et funksjonsbibliotek i en programmeringssammenheng er en verktøykasse med funksjoner som er klare for bruk videre i kodingen). Det eksisterer flere parsingsverktøy (parse = syntaktisk analyse), blant annet «Elementree» og «lxml», men for denne oppgaven ble «lxml» valgt. Dette verktøyet kan benyttes til å parse XML, Schematron, DTD og Relax NG, validere XML, og gjennomføre skjema- og schematronkontroll. Her ble verktøyet mest brukt for å lese GML-dokumenter, navigering i dokumentet og til gjennomføring av skjemakontroll. Geometrien ble hentet ut ved hjelp av funksjonsbibliotekene «OGR/GDAL» og «Shapely» etter parsing for så å bli lagret i lister. Andre måter å lagre informasjon på er ved bruk av matriser (array) eller i ordbøker (dictionaries). Det ble videre brukt funksjoner fra overnevnte funksjonsbiblioteker til å gjennomføre testing på geometrien. Til slutt ble Python koden koblet opp mot et grafisk brukergrensesnitt, se Vedlegg H – UI (gmlkontroll.ui).

3.3 Utvikling av grafisk brukergrensesnitt

For at programmet skulle være brukervennlig, ble det laget et grafisk brukergrensesnitt. Dette grafiske brukergrensesnittet bestod av hurtigtaster og menyer som er veldig intuitive for en bruker. Det finnes flere programmer som kan benyttes for å lage grafiske brukergrensesnitt, i denne oppgaven ble programmet «QT Creator» benyttet. Se Figur 7.



Figur 7: Enkelt grafisk brukergrensesnitt laget med QT Creator

3.4 Testing av program

Programmet ble testet med et egenprodusert datasett som bestod av polygoner kalt «polytest.gml», datasettene som Kartverket gjorde tilgjengelige «Veg.gml» og «VegSti.gml». Disse inneholder geometri som eksisterte i tilhørende FKB-applikasjonsskjema (http://skjema.geonorge.no/SOSITEST/produktspesifikasjon/). For testing av programmet ble hovedsakelig datasettene FKB-TraktorvegSti46.xsd, polytest.gml, samt en versjon av VegSti.gml som innehold litt færre objekter og ingen høydeverdier (VegStiForenklet.gml) benyttet.

4. Resultater

Tilgangen til GML-data var begrenset for denne oppgaven, noe som medførte at datasettene omtalt i kapittel 3.4 Testing av program var lite avanserte. Det var mulig å gjennomføre skjemakontroll med datasettene fra Kartverket, men grunnet mangel på schematronskjema ble denne kontrollen fjernet fra det endelige programmet.

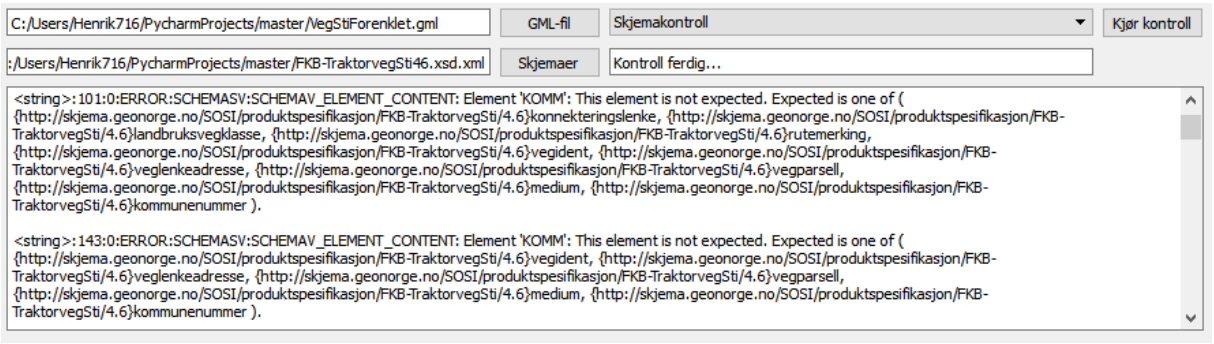
Datasettene inneholdt enkle polygoner og linjer, som det ble utført enkle geometriske tester på ved hjelp av programmet utviklet for oppgaven.

Programmet lager store resultatfiler, avhengig av hvor mye som skal undersøkes. Det vises derfor bare utsnitt av disse nedenfor.

4.1 Skjemakontroll

Figur 8 viser deler av en utskrift etter en gjennomført skjemakontroll. Utskriften viser hvilke linjer i GML-dokumentet VegStiForenklet.gml feil befinner seg og hvilke elementer som er forventet å finne.

Programmet laget for denne oppgaven klarte å finne GML-skjemaene på nett av seg selv, men det slet med å finne applikasjonsskjemaene. Dette medførte at applikasjonsskjemaene måtte lastes inn eller bli definert med en sti (URL).



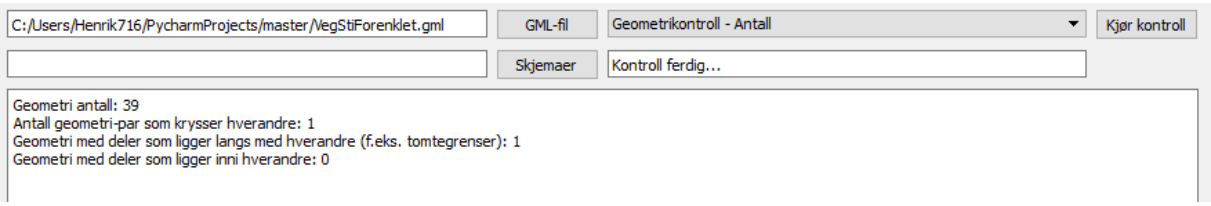
Figur 8: Utsnitt av utskrift ved skjemakontroll

4.2 Geometrikontroll

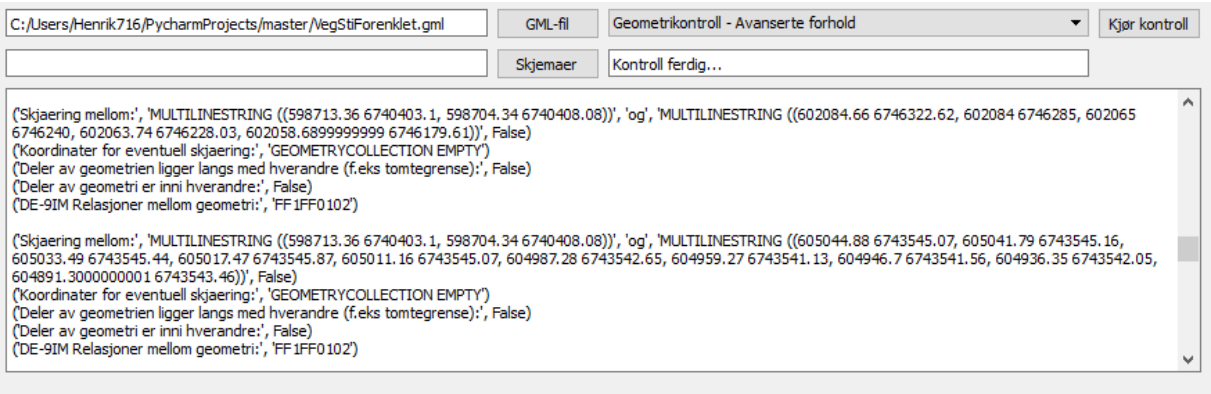
Figur 9 viser hvordan programmet kan gi informasjon til brukeren om hvert enkelte geometri-element i et GML-dokument. Her er dette vist med et par «multilinestrings» i GML-dokumentet VegStiForenklet.gml.

Figur 9: Utsnitt av utskrift ved geometrikontroll - enkle forhold

Figur 10 og Figur 11 viser hvordan programmet kan gi brukeren informasjon om hvordan geometri-elementene i et GML-dokument er i forhold til hverandre. Dette er igjen vist med datasettet VegStiForenklet.gml.



Figur 10: Utsnitt av geometrikontroll - antall



Figur 11: Utsnitt av utskrift ved geometrikontroll – avanserte forhold

4.3 Rapportering

Figur 12 viser et eksempel på hvordan en rapport kan se ut.

SOSI-kontroll (GML)

Resultater av kontroll

Informasjon om prosjektet:

Prosjektets navn: VegStiForenklet Demo

Dato: 27.04.2016

Koordinatsystem: Default (f.eks. WGS 1984)

Programvare brukt: GML-kontroll versjon 1.0

Kontrollerte filer: VegStiForenklet.gml, VegSti.sch

(+ skjemaer på nett og andre objekter det er referert til)

Syntakskontroll

GML-dokument: OK

Skjemaer og applikasjonsskjemaer: OK

Schematron: Feil

- Eks feil: Element nr. 2, 3
  mangler navnerom: *http://purl.oclc.org/dsdl/schematron*
- Eks fatal feil: Schematron ikke konform

Skjemakontroll

Sjekk mot GML-skjemaer: OK

Sjekk mot applikasjonsskjema: Feil

Elementet «KOMM» for «Multilinestring» nr. 4, 8, 16, 32
er definert feil. Forventet en av følgende elementer… f.eks.

- http://skjema.geonorge.no/SOSI/produktspesifikasjon/FKB-TraktorvegSti/4.6 – landbruksvegklasse, vegident, medium

Sjekk mot refererte objekter: OK

Schematronkontroll

Schematronsvalidering: Fatal feil

- Ikke gyldig schematronskjema

Statistikkrapport

Antall geometri-elementer: 39

- Linjer: 39

Antall geometri-par som skjærer: 1

Antall geometri-par som berører hverandre (f.eks tomtegrense): 1

Geometri som befinner seg inne i annen: 0

Gjennomsnittlig lengde: 149.67m

Gjennomsnittlig areal: 0

Kvalitet: …

Punkttetthet: …

Antall punkt pr. linje/kurve: …

Info om objekter: …

Kvalitetsrapport

Rapportere informasjon om relevante kvalitets-elementer.

Hva som er relevant å rapportere vil være avhengig av datasettet:

- Fullstendighet
- Logisk konsistens
- Posisjonsnøyaktighet
- Tematisk nøyaktighet
- Temporal kvalitet

Kontrollrapport

Informasjon om geometri vil variere fra bruker til bruker.

Det burde derfor være mulig å huke av hva som skal være med:

- Hvilke geometri som skjærer annen
- Koordinatliste for skjæring mellom geometri
- Eventuelle høydeverdier
- Kan være ønskelig at programmet skal kunne tegne noe av geometrien og vise det i denne delen av rapporten

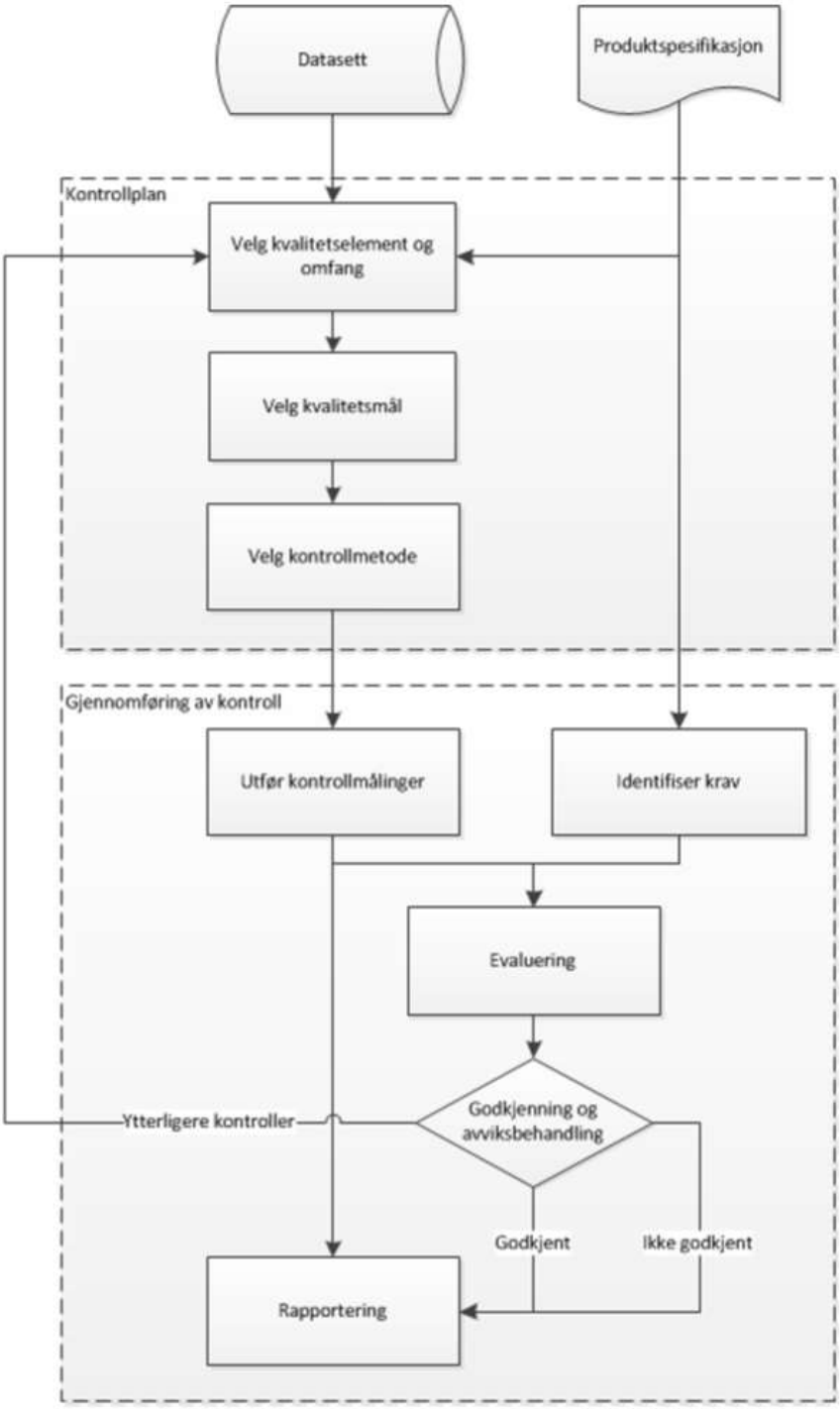
Figur 12: Eksempel på rapport etter kontroll av GML

5. Diskusjon

5.1 Kontroll av datakvalitet – Kontrollprosessen

Når en kontrollerer datakvalitet er det vanlig å dele opp prosessen i to trinn:

- Et trinn for oppsetting av en kontrollplan
- Et trinn for gjennomføring av selve kontrollen



Figur 13: Skjematisk oversikt over kontrollprosessen (Kartverket, 2015a)

Når en skal lage en kontrollplan, er det en rekke steg man må igjennom. Disse stegene er som vist i Figur 13, valg av:

- kvalitetselement og omfang
- kvalitetsmål
- kontrollmetode

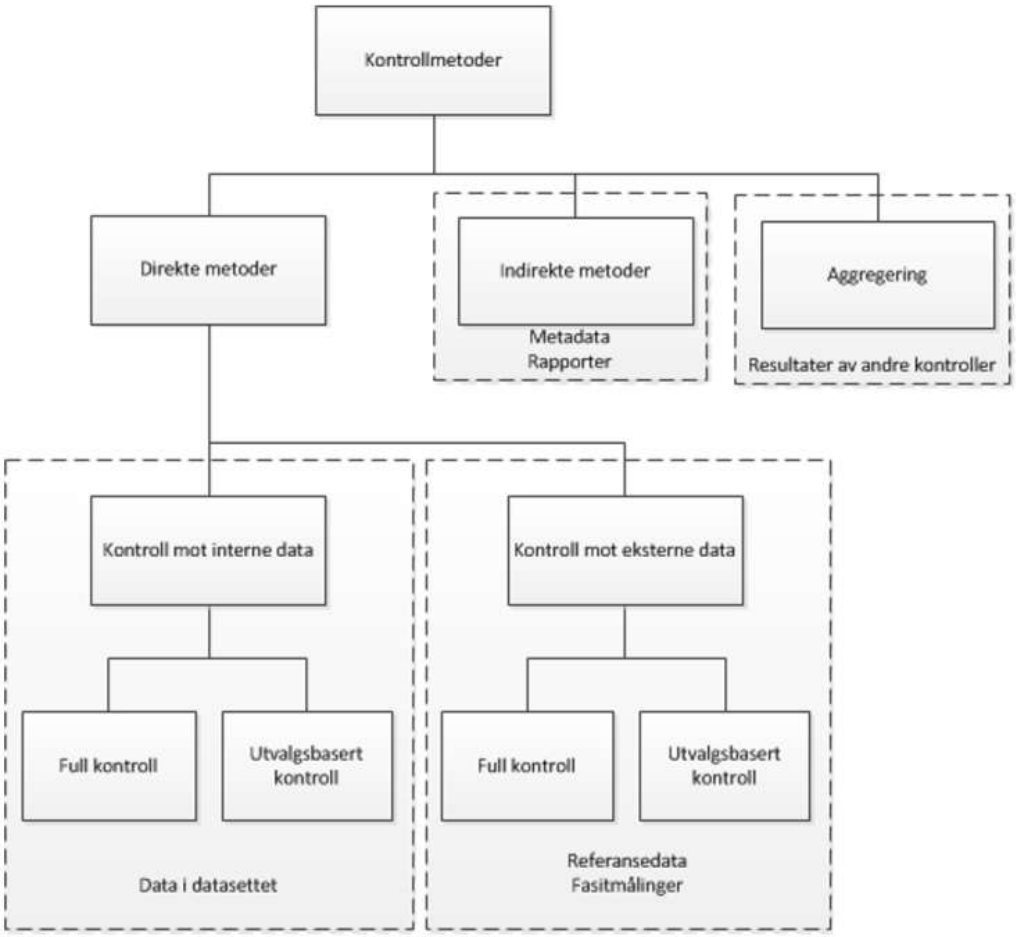
De første to trinnene vil være uavhengig av om vi skal gjennomføre en kontroll hvor data er på GML-formatet eller på SOSI-formatet.

Ved valg av kvalitetselement og omfang må brukeren velge en kvalitetsenhet som kontrollen skal gjennomføres med (en kvalitetsenhet: en kombinasjon av kvalitetselementer og et omfang). Det er viktig at det finnes frem mest mulig homogene data når en velger et omfang, slik at informasjonen som blir valgt ut gir best mulig representasjon av resten av datasettet.

Kvalitetsmålet angir hva resultatet av kontrollen skal vise. Valg av kvalitetsmål vil variere fra bruker til bruker avhengig av hvor strenge krav de har for kvaliteten til datasettet som kontrolleres. En liste over kvalitetsmål finnes i Kartverket sin standard for geodatakvalitet (Kartverket, 2015a).

Når det kommer til valg av kontrollmetode, er det en liten forskjell mellom GML-formatet og SOSI-formatet. Det er i hovedsak mest relevant å bruke direkte metoder for kontroll av begge formatene. Direkte kontrollmetode er en metode som anvendes direkte på datautvalget som skal kontrolleres. Her er det naturlig å gjennomføre en «full kontroll» istedenfor en «utvalgsbasert kontroll». Det er fordi at det vil være viktig å se etter feil i hele datasettet og ikke bare kontrollere et representativt utvalg. Illustrasjon av kontrollmetodene og deres egnethet vises i Figur 14 og Figur 15.

Skillet mellom metodene som blir brukt for kontroll av GML-formatet og SOSI-formatet vil være om det blir brukt kontroll av interne data, eksterne data eller begge deler. Ved kontroll av interne data brukes kun kontrolldata fra selve datasettet for å kontrollere dataene. Dette er metoden som brukes til å kontrollere dagens SOSI-format. Kontroll mot interne data vil være en viktig del av kontroll av GML-formatet ved at f.eks. burde gjennomføres diverse geometrikontroller av datasettet.



Figur 14: Klassifisering av kontrollmetoder etter behov for kontrolldata (Kartverket, 2015a)

Det å kontrollere datasettet med den interne metoden vil ikke være nok for å kontrollere GML fordi at formatet ikke kun bruker GML-dokumentet for å beskrive datasettet, men også GML-skjema, applikasjonsskjema, profiler og schematron. Dette blir sett på som eksterne data, noe som medfører at det også må bli gjennomført en kontroll mot eksterne data. Her vil det ikke kun være viktig å kontrollere GML-dokumentet opp mot de eksterne skjemaene, men også å kontrollere de eksterne filene.

Informasjon for hvordan eksterne data skal kontrolleres vil bli diskutert i senere avsnitt.

Mer informasjon om kontrollprosessen finnes i Vedlegg A – Kontrollprosessen og i Kartverket sin standard for geodatakvalitet (Kartverket, 2015a).

Kategori	Kvalitetselement	Kontrollmetoder								
		Indirekte	Direkte							
			Interne data	Eksterne data						
			Kontorarbeid				Fotogrammetri		Markarbeid	
		Kontroll av dokumentasjon	Automatiserte kontroller	Visualisering	Visuell kontroll mot ortofoto	Kontroll mot andre data	Visuell kontroll	Kontrollmåling	Synfaring	Kontrollmåling
Logisk konsistens	Formatkonsistens	+	++	-	-	-	-	-	-	-
	Domenekonsistens	+	++	-	-	-	-	-	-	-
	Konseptuell konsistens	+	++	-	-	-	-	-	-	-
	Topologisk konsistens	+	++	+	-	+	-	-	-	-
Fullstendighet	Manglende data	+	-	-	+	+	++	-	++	-
	Overskytende data	+	-	-	+	+	++	-	++	-
Egenskaps kvalitet	Ikke-kvantitativ egenskapsriktighet	+	-	-	-	++	-	-	++	-
	Kvantitativ egenskapsnøyaktighet	+	-	-	-	++	-	+	++	++
	Klassifikasjonsriktighet	+	-	-	+	++	+	-	++	-
Stedfestings nøyaktighet	Absolutt stedfestingsnøyaktighet	+	-	-	+	+	+	++	-	++
	Nabonøyaktighet	+	-	+	-	-	-	++	-	++
	Posisjonsnøyaktighet i rasterdata	+	-	-	+	++	-	-	-	-
	Stedfestingspålitelighet	+	-	-	-	-	-	++	-	++
Kvalitet på tidfesting	Tidsgyldighet	+	++	+	-	-	-	-	-	-
	Tidskonsistens	+	++	-	-	-	-	-	-	-
	Tidsnøyaktighet	+	-	-	-	++	-	-	-	-
Egnethet	Egnethetselement	+	-	-	-	-	-	-	-	-
	Aggregering	+	++	-	-	-	-	-	-	-

Figur 15: Oversikt over metodens egnethet for kontroll av ulike kvalitetselementer(Kartverket, 2015a)

- ++ Metoden er godt egnet
- + Metoden kan være egnet
- - Metoden er uegnet

5.2 Fordeler med GML

I Norge har vi lenge hatt SOSI-formatet, noe som andre land kunne ønske de selv hadde tilgjengelig på et internasjonalt språk. Formatet har gitt oss et fortrinn og muligheten til å ha mye kartdata tilgjengelig for egen bruk. Som en internasjonal standard vil det i fremtiden eksistere mye GML-data fra andre land. Dette er data som kan være ønskelig å bruke. GML-formatet støttes av flere systemer uten norsk tilpassing. Dette vil gjøre at flere enn vi nordmenn kan bidra med å holde formatet oppdatert.

GML er veldig fleksibelt, og er bedre for realisering av komplekse modeller som assosiasjoner og lineære referanser sammenlignet med dagens SOSI-format.

Innholdene i filene er skrevet i XML, noe som medfører at mye av innholdet i filene er lett å validere (XML 3-trinns validering). Et XML-basert format har også en lesbarhet som SOSI-formatet ikke har, og kan forstås lett av både menneske og datamaskin. Formatet XML er lett å transformere fra en form til en annen ved hjelp av de fleste programmeringsspråk.

5.3 Programvarens funksjonalitetskrav

Funksjonaliteten til et program som kontrollere GML-formatet basert på SOSI-standarden vil være avhengig av brukerens ønsker. Noen ønsker en rask oversikt og andre ønsker mer detaljert informasjon. Programvaren bør være fleksibel og møte brukernes behov. Et annet viktig spørsmål er om programvare skal være norsk og/eller engelsk. Funksjonalitetskrav beskrives mer detaljert i de følgene avsnitt.

5.3.1 Tolking av dokumenter og skjemaer

Skrive og lese GML

Programmet må kunne lese, samt forstå GML-dokumenter og applikasjonsskjemaer korrekt uten at det oppstår noen form for informasjonstap.

Programmet bør kunne skrive GML-dokumenter, skjemaer og applikasjonsskjemaer på gyldig XML.

Metadata

Programvaren må kunne behandle metadata og sikre at data hvor verdien er oppgitt som metadata om et objekt blir identifisert riktig.

Xlink støtte

Programvaren burde ha implementert mulighet til å prosessere GML-objekter på XML-formatet. Det må da også være støtte for referanser til andre objekter innen eller utenfor det samme GML-dokumentet. Programmet burde også ha implementert funksjon for utvidet Xlink.

Koordinatsystem

Programmet må ha muligheten til å definere et koordinatsystem, dette gjelder også i eksempler hvor et koordinatsystem ikke er oppgitt i filen. Det må i dette tilfellet settes et standard (default) koordinatsystem hvor de geometriske objektene i filen blir plassert.

5.3.2 Syntakskontroll – Velformet XML, skjemakontroll og schematronkontroll

Syntakskontroll – Velformet XML

Programvaren må ha muligheten til å gjennomføre en syntakskontroll for å finne ut om dokumenter er skrevet på velformet XML (denne kontrollen faller bort dersom utvidet syntakskontroll blir implementert).

Skjemakontroll

Programmet må ha muligheten til å kjøre skjemakontroll mellom GML-dokument, skjemaer, profiler og applikasjonsskjemaer.

Schematronkontroll

Det burde være mulighet for å gjennomføre en schematronkontroll.

5.3.3 Utvidet syntakskontroll

Kontroll av GML-dokument

Programmet burde ha muligheten til å kontrollere GML-dokumenter i samsvar med kapittel 5.4.1 Utvidet syntakskontroll - Kontroll av GML-dokument.

Kontroll av GML-applikasjonsskjema

Det burde også være mulighet for å kontrollere GML-skjema og applikasjonsskjema i samsvar med det som er beskrevet i kapittel 5.4.2 Utvidet syntakskontroll - Kontroll av GML-applikasjonsskjema.

Kontroll av schematronskjema

Selv om schematron ikke alltid vil bli benyttet i kontroll av GML-formatet, så vil det fortsatt være ønskelig at programvaren skal kunne kontrollere det. Her vil det bli sjekket om schematron har blitt implementert riktig ved at informasjonen som er i filen er gyldig og at det som er omtalt i kapittel 5.4.3 Utvidet syntakskontroll - Schematron implementering følges.

5.3.4 Rapportering av resultater og statistikk

Det er veldig greit for brukeren om de får en tilbakemelding på hvordan kontrollen har gått, og det finnes flere måter å gi brukeren denne informasjonen. Metoden som er blitt valgt for denne oppgaven er å få programmet til å skrive ut en eller flere rapporter. Dette er valgt fordi SOSI-kontrollen allerede benytter kontroll og statistikkrapporter for å gi brukerne av programmet informasjon.

Statistikkrapport

Rapportering av statistikk gjør at man kan se hvilke feil som eksisterer i datasettet og hvor disse feilene befinner seg. En statistikkrapport som er tilnærmet lik den som blir brukt i SOSI-kontrollen burde holde for å få dekket behov av denne typen.

En slik rapport burde i hvert fall inneholde informasjon om feil i datasettet, og det vil være ønskelig å gi brukeren muligheten til å velge selv hva som skal være med i en slik rapport.

Kontrollrapport

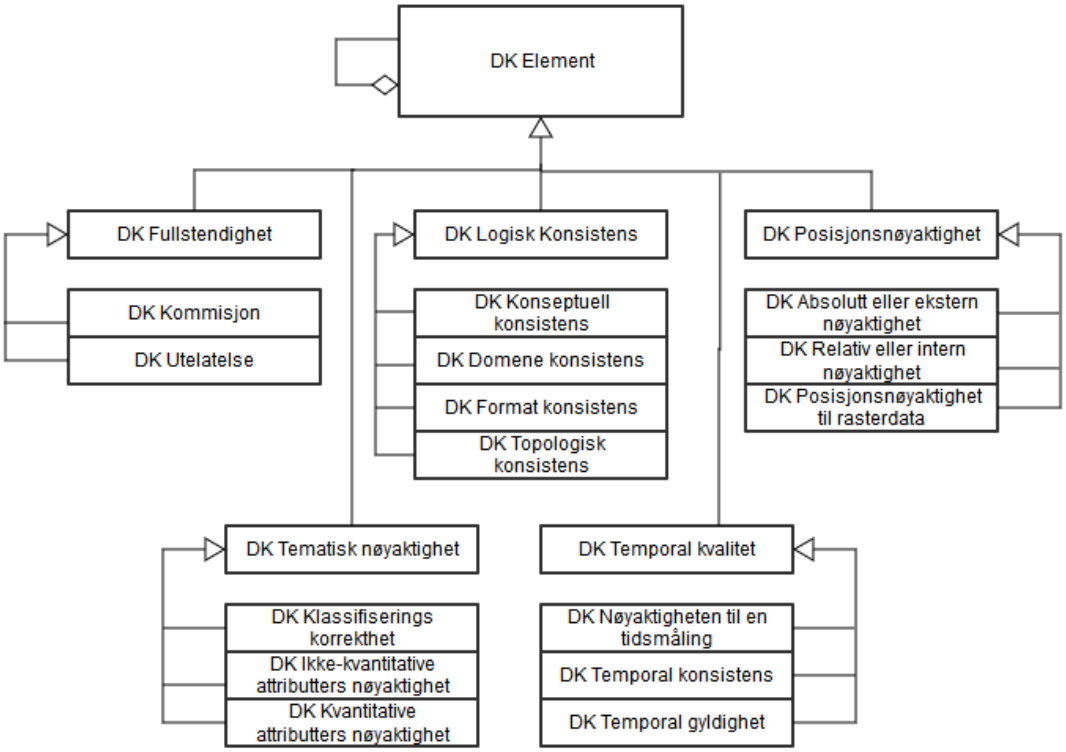
En kontrollrapport burde inneholde en utskrift av informasjon som ikke inngår i statistikk eller kvalitetsrapporten. Dette kan være informasjon om hvilke datasett som er blitt sjekket,

syntaks/formatsjekk og eventuelt annen sjekk av innholdet i dokumentet/skjemaene som skulle være ønskelig.

Her vil det også være naturlig å rapportere mye av informasjon fra den ekstra geometrikontrollen (5.4.5).

Kvalitetsrapport

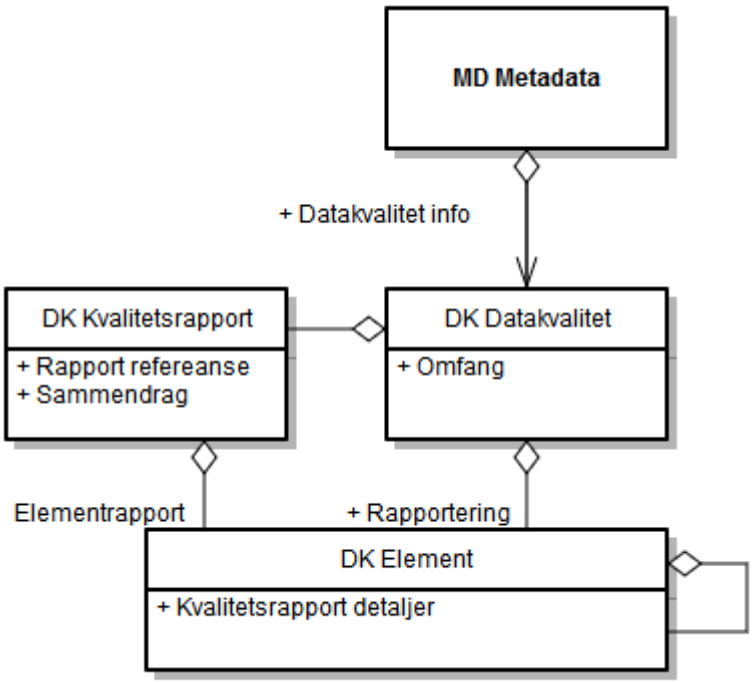
Når det gjelder datakvalitet, så er det en del elementer som kan være ønskelig å få med i en kvalitetsrapport. En beskrivelse av elementene finnes i Vedlegg C – Kvalitetselementer (ISO) og illustreres her med Figur 16.



Figur 16: Oversikt over datakvalitetselementene

Etter at datakvalitetselementene er blitt evaluert (se kapittel 5.1 Kontroll av datakvalitet – Kontrollprosessen, Vedlegg A – Kontrollprosessen eller ISO standarden (ISO, 2013) for beskrivelsen av denne prosessen), så er det på tide å rapportere resultatene.

Datakvalitet skal rapporteres som metadata og skal følge ISO standardene ISO 19115-1:2014 (ISO, 2014) og ISO 19115-2:2009 (ISO, 2009), se Figur 17. For å gi mer informasjon enn det som er rapportert som metadata, så er det mulig å lage en frittstående kvalitetsrapport. Det er kun et krav for denne rapporten i ISO standarden og det er at den ikke skal erstatte metadataen som skal bli rapportert. Dette betyr at det er opp til de som lager et program som kan rapportere datakvalitet å bestemme hvordan strukturen til denne rapporten skal være.



Figur 17: Rapportering av datakvalitet basert på figur fra ISO19157 (ISO, 2013)

5.4 SOSI-kontroll for GML-formatet

En utfordring i oppgaven var å finne ut hva programvare for kontroll av GML skal gi brukeren informasjon om. Svaret på dette vil en utvikler av et slikt program få innblikk i etter å ha lest denne oppgaven.

Når en skal lage programvare som kan kontrollere GML-formatet er en programmeringsbasert metode med bruk av funksjonsbiblioteker den mest aktuelle. I motsetning til en metode basert på ren programmering hvor man må lage alle funksjonene som skal brukes selv. Ved kontroll av geodata vil det i hovedsak være nødvendig å lage funksjoner som tar for seg konseptuelle regler og geometriberegninger.

Ved bruk av funksjonsbiblioteker deles arbeidet opp ut ifra hva som skal være med i hovedprogrammet og hva som skal være i bibliotekene. Det kan for eksempel være biblioteker med funksjoner for geometri som dekker kravene spesifisert i ISO standarder. Disse funksjonsbibliotekene trenger ikke nødvendigvis å være laget kun for bruk på GML-formatet og kan brukes på de fleste nåværende og fremtidige lagringsformater. Slike biblioteker trenger derfor kun å lages en gang i henhold til ISO standarder, og må i senere tid holdes oppdatert. Ved valg av brukergrensesnitt, så vil et grafisk brukergrensesnitt være det mest brukervennlig. Dette er det motsatte av et tekstbasert brukergrensesnitt hvor man må skrive inn kommandoer for at programmet skal utføre oppgaver, noe som kan bli sett på som komplisert for personer med lite datakunnskap.

Ved kontroll av geodata, er det viktig at man har de verktøyene som man trenger tilgjengelig. Det ble undersøkt om det eksisterer programvare som kan kontrollere GML-formatet på samme måte som SOSI-formatet blir kontrollert av programmet SOSI-kontroll. I studiet viste det seg at det ikke var noe program med samme funksjonalitet, men det eksisterer programmer som kan undersøke om dokumentene og skjemaene er skrevet på velformet XML (syntaksvalidering), skjemakontroller og schematronkontroller. Mye av denne programvaren er siktet mot XML og er ikke laget spesifikk for GML.

Fremgangsmåten som i dag gjelder for kontroll av GML er kjent og består av stegene:

- Syntakskontroll (velformet XML)
- Skjemakontroll
- Schematronkontroll

Disse tre stegene er et godt utgangspunkt når det kommer til å kontrollere GML, men det vil ikke holde dersom formatet virkelig skal kvalitetssikres. For at GML-formatet skal kunne bli kontrollert på et nivå som er godt nok, så kreves en:

- Utvidet syntakskontroll

En videreutvikling av syntakskontrollen vil sjekke innholdet i GML-dokumenter, applikasjonsskjema og schematronsskjema på en grundig måte. Hva som blir kontrollert av den nye syntakskontrollen blir beskrevet i kapittel 5.4.1 Utvidet syntakskontroll - Kontroll av GML-dokument, 5.4.2 Utvidet syntakskontroll - Kontroll av GML-applikasjonsskjema og 5.4.3 Utvidet syntakskontroll - Schematron implementering, og det må også utføres et fjerde steg:

- Ekstra kontroll av geometri

Dette steget vil være nødvendig ettersom at GML-skjema/applikasjonsskjema kun kan finne ut om geometri er definert feil og finner da ikke andre geometriske feil. Dette steget blir gått nøyere igjennom i kapittel 5.4.5 Ekstra kontroll av geometri.

Den egenutviklede koden greide å detektere feil i GML testdatasettene som ble brukt i denne oppgaven.

5.4.1 Utvidet syntakskontroll - Kontroll av GML-dokument

Det å kontrollere syntaksen til et GML-dokument er viktig selv om det ikke nødvendigvis er mye komplisert informasjon i dokumentet. De viktigste tingene som bør kontrolleres i et GML-dokument er *riktig syntaks, referansesjekk* og *konformitetsjekk.*

Riktig syntaks

Er en enkel syntakskontroll som sjekker om filen er skrevet på velformet XML og er et av de tre kjente stegene for å kontrollere GML. Det eksisterer flere programmer og verktøy som kan gjennomføre denne prosessen.

Referanser

Et GML-dokument skal referere til tilhørende applikasjonsskjema. Måten dette gjøres på er vist i Figur 18, hvor attributtet «xsi:schemaLocation» referer til applikasjonsskjemaet sin plassering. Det vil også være lurt at programvare som kontrollerer GML-dokumentet automatisk undersøker om applikasjonsskjemaene det refereres til eksisterer. Dette gjøres ved at programmet undersøker om det er tilgang til skjema som det blir direkte eller indirekte referert til.

```
<?xml version="1.0" encoding="utf-8"?>
<gml:FeatureCollection
 xmlns:gml="http://www.opengis.net/gml/3.2"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xlink="http://www.w3.org/1999/xlink"
 xmlns:app="http://skjema.geonorge.no/SOSI/produktspesifikasjon/FKB-TraktorvegSti/4.6"
 xsi:schemaLocation="http://skjema.geonorge.no/SOSI/produktspesifikasjon/FKB-TraktorvegSti/4.6
 http://skjema.geonorge.no/SOSITEST/produktspesifikasjon/FKB-TraktorvegSti/4.6/FKB-TraktorvegSti46.xsd"
 gml:id="LOCAL_ID">
 <gml:boundedBy>
   <gml:Envelope srsName="EPSG:5972">
     <gml:lowerCorner>594529.00 6737462.00 NaN</gml:lowerCorner>
     <gml:upperCorner>610444.00 6754465.00 NaN</gml:upperCorner>
   </gml:Envelope>
 </gml:boundedBy>
```

Figur 18: Utsnitt fra hodet til et GML-dokument (eksempel)

Konformitet

Ved kontroll av GML-dokumenter vil det være nødvendig at programmet kan verifisere at GML-applikasjonsskjema det blir referert til er konforme med den internasjonale standarden til OGC (OGC, 2007a). Det vil også være nødvendig å sjekke om selve GML-dokumentet er i samsvar med alle begrensningene spesifisert i denne standarden.

Mer informasjon om hvordan GML-dokumenter burde kontrolleres finnes i Vedlegg D – Kontroll av GML-dokument.

5.4.2 Utvidet syntakskontroll - Kontroll av GML-applikasjonsskjema

Applikasjonsskjemaer er med på å kontrollere at informasjonen i GML-dokumentet er riktig, men skjemaene har også behov for å kontrolleres. De viktigste tingene programvare som skal kunne kontrollere GML-applikasjonsskjemaer burde kunne undersøke er *bruk av navnerom, generelle regler, referanser, riktig definering av elementer, objekter og geografiske egenskaper (features),* og *korrekt identifisering av egenskaper og samlinger*

Navnerom

Det må kontrolleres at alle komponentene i et applikasjonsskjema er forbundet med et XML-navnerom og dette navnerommet skal ikke være «http://opengis.net/gml/3.2». Et eksempel på riktig brukt av navnerom er i Figur 19. Her vises et utsnitt av applikasjonsskjemaet «traktorvegsti», og bruk av navnerom er av formen «import namespace».

```
  <import namespace="http://www.interactive-instruments.de/ShapeChange/AppInfo"
  schemaLocation="http://shapechange.net/resources/schema/ShapeChangeAppinfo.xsd"/>
  <import namespace="http://www.opengis.net/gml/3.2" schemaLocation="http://schemas.opengis.net/gml/3.2.1/gml.xsd"/>
  <!--XML Schema document created by ShapeChange - http://shapechange.net/-->
  <simpleType name="BelysningType">
    <annotation>
      <documentation>om det er permanent belysning langs sti eller løype

-- Tilleggsopplysninger FKB ---
Data som tidligere var registrert som objekttype Lysløype skal i FKB-TraktorvegSti angis som veglenker med belysning.</documentation>
      <appinfo>
        <taggedValue xmlns="http://www.interactive-instruments.de/ShapeChange/AppInfo" tag="SOSI_navn">BELYSNING</taggedValue>
      </appinfo>
    </annotation>
    <union memberTypes="app:BelysningEnumerationType app:BelysningOtherType"/>
  </simpleType>
  <simpleType name="BelysningEnumerationType">
    <annotation>
      <documentation>om det er permanent belysning langs sti eller løype

-- Tilleggsopplysninger FKB ---
Data som tidligere var registrert som objekttype Lysløype skal i FKB-TraktorvegSti angis som veglenker med belysning.</documentation>
      <appinfo>
        <taggedValue xmlns="http://www.interactive-instruments.de/ShapeChange/AppInfo" tag="SOSI_navn">BELYSNING</taggedValue>
      </appinfo>
    </annotation>
    <restriction base="string">
      <enumeration value="1"/>
      <enumeration value="2"/>
      <enumeration value="3"/>
    </restriction>
  </simpleType>
```

Figur 19: Utsnitt fra GML-applikasjonsskjema «traktorvegsti»

Generelle regler

Applikasjonsskjemaet må tilfredsstille generelle regler beskrevet i ISO 19136:2007, 21.2.1 (ISO, 2007; OGC, 2012a). Dette kan kontrolleres med programvare som inspirerer applikasjonsskjemaet og sjekker om det er i samsvar med reglene definert i standarden.

Referanser

Ved kontroll av applikasjonsskjemaet må det undersøkes at GML-skjemaene og GML-profilene det blir referert til blir importert riktig. Dette gjøres ved å undersøke om import uttalelser for både skjema og profiler i applikasjonsskjemaet er riktige.

Riktig definering av elementer, objekter og geografiske egenskaper (features)

I GML-applikasjonsskjema defineres det mange forskjellige elementer, objekter og geografiske egenskaper. Det vil være nødvendig å kontrollere at disse er definert riktig. En feil under defineringen av de forskjellige typene kan for eksempel resultere i at elementer blir tolket som objekter.

Korrekt identifisering av egenskaper og samlinger

Det finnes mange typer egenskaper som er aktuelle for GML. Dette kan være egenskaper om metadata og datakvalitet, romlig geometri, romlig topologi, temporale egenskaper og egenskaper for beliggenhet. Det vil være nødvendig å kontrollere at disse egenskapene blir identifisert riktig, slik at for eksempel metadata blir identifisert som metadataegenskap.

Denne type kontroll vil også være nødvendig for identifisering av samlinger av objekter eller egenskaper.

Mer informasjon om hvordan applikasjonsskjema burde kontrolleres finnes i Vedlegg E – Kontroll av GML-applikasjonsskjema.

5.4.3 Utvidet syntakskontroll - Schematron implementering

Et schematron-skjema regnes som en dokumenttype og inneholder påstander, begrensninger eller vilkår til dokumenter. Skjemaet er merket opp med ulike elementer og attributter for å teste disse påstandene, og inneholder påstander for simplifisering og gruppering. Figur 20 viser et eksempel på en schematron påstand.

Rent teoretisk reduseres et schematronskjema til en regelsystem hvor vilkårene er boolske funksjoner som fungerer som et eksternt spørrespråk (Query Language) mot de gjeldende XML-dokumentene (f.eks. GML-applikasjonsskjema). Egenskapene til schematron gjør at det kan brukes til å kontrollere deler av GML-dokumenter og applikasjonsskjema.

```
<?xml version="1.0" encoding="utf-8"?>
<schema xmlns="http://purl.oclc.org/dsdl/schematron" xmlns:sch="http://purl.oclc.org/d
sdl/schematron">
    <title>Schematron constraints for schema 'Svalbardplan 20130601 test'</title>
    <ns prefix="sch" uri="http://purl.oclc.org/dsdl/schematron" />
    <ns prefix="svp" uri="http://skjema.geonorge.no/sosi/fagområdespesifikasjon/Svalba
rdplan/4.5" />
    <pattern>
        <rule context="svp:AdministrativEnhetskode">
            <assert test="svp:landkode = 'NO'">Landkode alltid NO: Landkode kan bare
være 'NO' </assert>
        </rule>
    </pattern>
</schema>
```

Figur 20: Eksempel på schematron påstand som sjekker «landkode» (Norge-digitalt, 2015)

Forhold i applikasjonsskjema som schematron kan brukes til å kontrollere er for eksempel:

- Semantiske krav
- Topologiske krav

Semantikk

I et schematronskjema lages det valideringsfunksjoner. Dette er funksjoner som går igjennom to steg. Først transformeres skjemaet til *minimalistisk syntaks*, for så å teste instansen mot den *minimalistiske syntaksen*. Resultatet av testen er enten en feilmelding eller at instansen er

gyldig eller ikke gyldig. Slike funksjoner kan validere hele applikasjonsskjema, instanser som trengs validering og eksterne XML-dokumenter som det refereres til.

En valideringsfunksjon i schematron vil være avhengig av rekkefølgen som elementer, mønstre og påstander blir testet. Det vil derfor være viktig å tenke på hvordan valideringsfunksjonene blir laget.

Syntaks

På samme måte som GML-dokumenter, så har et schematronskjema krav om riktig syntaks:

- Det er viktig at schematronskjemaet er skrevet på gyldig XML. Dette kan verifiseres på samme måte som med GML-dokumentene og applikasjonsskjemaene.
- Alle elementer i schematron skal være kvalifisert med navnerommet URI: *http://purl.oclc.org/dsdl/schematron*
- Krav til riktig bruk av mellomrom (whitespace).
- Riktig bruk av kjerne og hjelpe-elementer.

Konformitet

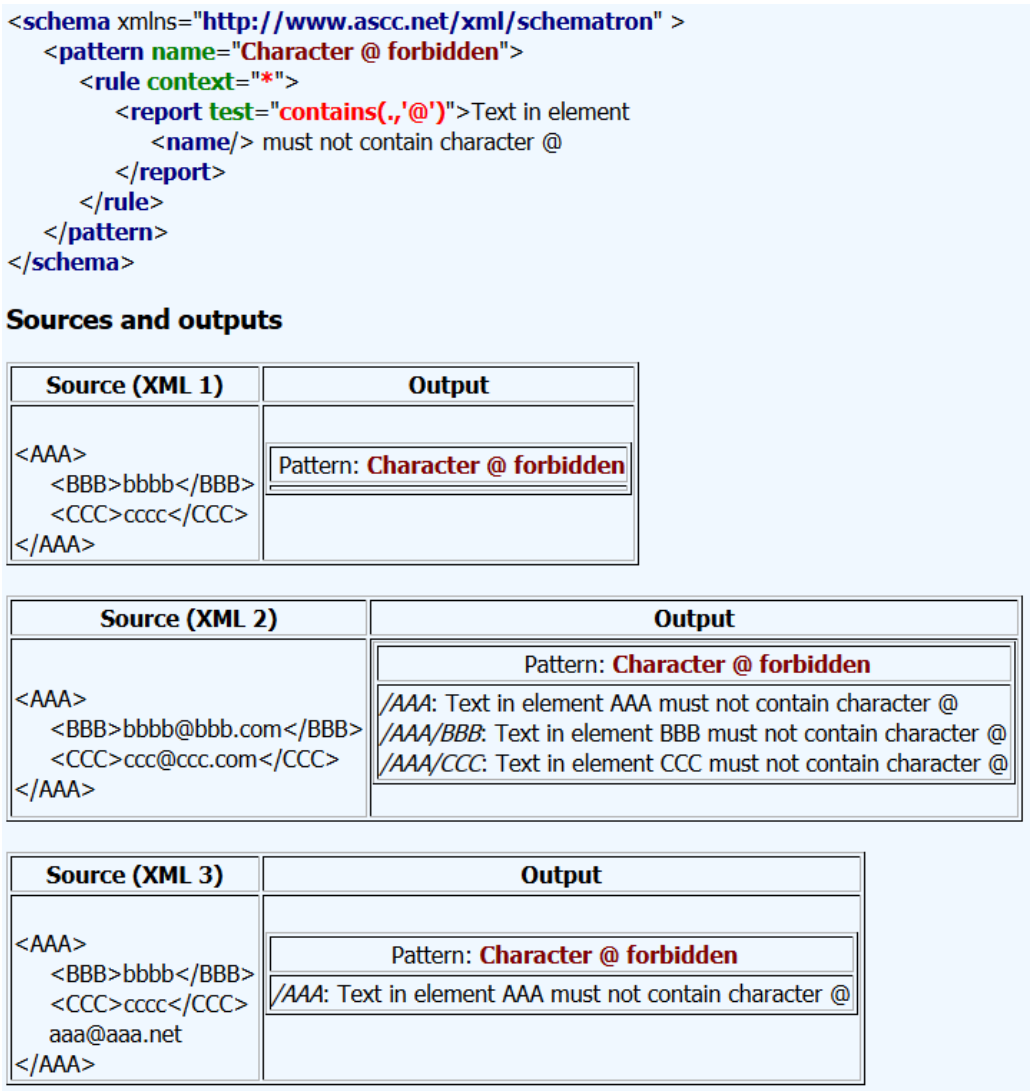
Schematron er en ISO standard (ISO, 2006) noe som medfølger strenge regler for hvordan skjemaene skal skrives. Skjemaene kan enten skrives i henhold til «enkelt konformitet» eller «full konformitet. Dersom «enkel konformitet» brukes som konformitetsnivå, så skal implementasjonen av schematron klare å rapportere for alle XML-dokumenter om strukturen ikke er i samsvar med reglene som angir et gyldig schematronskjema. Ved «full konformitet», så skal en implementasjon av schematron klare å bestemme for alle XML-dokumenter om det er et riktig skjema.

Eksempler på schematron-påstander (Nic), se Figur 21 for et illustrert eksempel, er blant annet å sjekke:

- om rot-elementet har et spesifikt navn
- om et element er tomt
- om verdien til et element er heltall (integer)
- at en paragraf i XML2 kun kan starte med ord som er spesifisert i XML1 (file source1.xml)

eller å bestemme at:

- et element må ha et bestemt attributt dersom det er inne i et annet element, men trenger ikke å ha det ellers
- enkelte symboler ikke skal kunne brukes i teksten
- dersom et element har et attributt, så må den også ha et annen attributt
- summen av verdier til relevante elementer skal være lik 100



Figur 21: Schematron eksempel hvor @ ikke er lov å bruke i teksten – Schematron Tutorial (Nic)

Mer informasjon om schematron implementering finnes i Vedlegg F – Schematron implementering.

5.4.4 Skjemakontroll

Skjemakontroll er et av de tre kjente stegene i kontroll av GML-formatet. Og det vil ofte være nødvendig med flere enn et applikasjonsskjema for å kunne kontrollere et GML-dokument.

Skjemakontrollen har ikke noe behov for videreutvikling, men det burde i hovedsak være 9 forskjellige applikasjonsskjemaer tilgjengelig til bruk ved kontroll av formatet. Hvor mange av disse som benyttes, vil avhenge av GML-dokumentet sitt innhold.

De 9 applikasjonsskjemaene er de som definerer (de første 4 er nok de som vil bli mest brukt) (OGC, 2016):

- *egenskaper og samlinger av egenskaper*
- *romlig geometri*
- *romlig topologi*
- *tid*
- *koordinatreferansesystemer*
- *dekning (coverage)*
- *observasjoner*
- *dictionaries og definisjoner*
- *verdier*

Det vil i flere tilfeller være ønskelig med andre mere spesifikke applikasjonsskjemaer ut ifra hva som datasettene skal inneholde. Slike applikasjonsskjemaer vil det være opp de som har behov for dem å lage.

Programmet laget for denne oppgaven slet med å finne applikasjonsskjemaer på nett, noe som et ferdig program bør klare, grunnen er at det ikke er vanlig å ha slike skjema liggende på sin egen datamaskin. Det kan være ønskelig å kunne laste inn applikasjonsskjema manuelt som gjort i oppgaven i tilfeller hvor skjema ikke eksisterer på nett enda, eller ved mangel på nettilgang.

5.4.5 Ekstra kontroll av geometri

GML-skjema og applikasjonsskjema brukes for å definere all romlig geometri som det skal være mulig å bruke i et GML-dokument. Ved kontroll mot disse skjemaene vil det være mulig å se om det finnes geometri i filen som er definert feil, men det vil ikke være mulig å finne andre feil i geometrien. Det som definerer hva som er feil geometri kan avhenge av brukeren eller en eventuell standard. Eksempler på geometri som kan være feil er f.eks. polygoner som krysser hverandre eller geometri som ligger utenfor området definert i GML-dokumentet (bounded box).

Det vil være ønskelig at et program som skal kontrollere GML-formatet også skal ha muligheten til å kontrollere, samt rapportere om geometrien stemmer eller ikke i henhold til kravene som er satt. Et program vil kunne gjøre dette dersom det programmeres egnede funksjoner som henter informasjon ut fra GML-dokumentet, for så gjennomføre tester på geometrien som skal kontrolleres.

```
# Henter ut koordinatene fra GML-dokumentet
gml = etree.parse('polytest.gml')
linearing = []
for geography in gml.findall('{http://www.opengis.net/gml/3.2}featureMember'):
    for polygon in geography.findall('{http://www.opengis.net/gml/3.2}Polygon'):
        for coordinates in polygon.findall("{http://www.opengis.net/gml/3.2}outerBoundaryIs/"
                                           "{http://www.opengis.net/gml/3.2}LinearRing"):
            linearing.append((coordinates.findtext("{http://www.opengis.net/gml/3.2}coordinates")))
```

Figur 22: Eksempel på egenutviklet kode som kan hente ut informasjon om polygoner.

Informasjon ble hentet ut av dokumentet som vist i eksempelet i Figur 22. Dette ble gjort med «lxml-etree», som gjør det lett å navigere og søke i XML-dokumenter (en fordel med XML-dokumenter er at de er lette å navigere og søke i). Etter at informasjon er hentet ut av dokumentet, så kan geometrien lagres på en slik måte at den er lett tilgjengelig for videre bruk. I denne oppgaven ble blant annet funksjonsbiblioteket «GDAL/OGR» benyttet for videre testing av geometri. Dette biblioteket inneholder funksjoner for bruk/manipulering av geografisk data, blant annet finnes det en funksjon for skjæring (intersection) og union som kan undersøke deler av geometrien. Se Figur 23 for eksempel på bruk av «OGR/GDAL».

```
# Lager polygoner utifra informasjonen som er hentet - OGR/GDAL
polygon = []
for poly in linearing:
    gml = """<gml:Polygon><gml:outerBoundaryIs><gml:LinearRing><gml:coordinates>""" + poly + \
          """</gml:coordinates></gml:LinearRing></gml:outerBoundaryIs></gml:Polygon>"""
    polygon.append(ogr.CreateGeometryFromGML(gml))

# Undersøker om det finnes polygoner som skjærer hverandre - OGR/GDAL
for i in xrange(len(polygon)):
    for j in xrange(i + 1, len(polygon)):
        intersection = polygon[i].Intersection(polygon[j])
        print 'Skjæring mellom: ' + str(polygon[i]), str(polygon[j])
        print 'Koordinatene for skjæring: ' + intersection.ExportToGML()
```

Figur 23: Eksempel på egenutviklet kode som sjekker om polygoner skjærer hverandre – OGR/GDAL

Funksjonsbiblioteket «Shapely» har også blitt benyttet. Her brukes en rekke funksjoner fra «GEOS» biblioteket. Dette er en overføring av «JTS» (Java Topology Suite), som er biblioteket for geometri som blir benyttet i programmet «PostGIS». Etter litt testing viste det seg at «Shapely» er mere brukervennlig i sammenligning med «OGR/GDAL». Funksjonsbiblioteket inneholder også flere muligheter for å gjennomføre romlige analyser, samt muligheter for å hente ut diverse informasjon om geometrien i et datasett. Se Figur 24.

Det vil nok være naturlig å bruke en kombinasjon av disse to funksjonsbibliotekene eller noe som tilsvarer deres funksjonalitet.

```
# Skriver ut diverse informasjon om polygonene - Shapely
for polygon in shapelyPolygon:
    print polygon
    print 'Arealet er:', polygon.area
    print 'Lengden er:', polygon.length
    print 'Geometritype:', polygon.geom_type
    print 'Geometrien har høydeverdi:', polygon.has_z
    print 'Geometrien er gyldig:', polygon.is_valid
    print 'Geometrien krysser ikke seg selv:', polygon.is_simple

for i in xrange(len(shapelyPolygon)):
    for j in xrange(i + 1, len(shapelyPolygon)):
        print 'Skjæring mellom:', shapelyPolygon[i], shapelyPolygon[j], \
            shapelyPolygon[i].intersects(shapelyPolygon[j])
        print 'Koordinater for eventuell skjæring:', shapelyPolygon[i].intersection(shapelyPolygon[j])
        print 'Deler av polygonene ligger langs med hverandre (f.eks tomtegrense):', \
            shapelyPolygon[i].touches(shapelyPolygon[j])
        print 'Deler av polygonene er inni hverandre:', shapelyPolygon[i].within(shapelyPolygon[j])
        print 'DE-9IM Relasjoner mellom polygonene:', shapelyPolygon[i].relate(shapelyPolygon[j])
```

Figur 24: Eksempel på egenutviklet kode som printer ut diverse informasjon om polygoner – Shapely

Som vist i Figur 25 brukes «OGR/GDAL» for å lese GML-dokumentet og for å hente ut geometrien i form av «WKT» (Well Known Text). Geometrien blir så lagret som «Shapely-objekter» for at funksjonene i dette funksjonsbiblioteket skal kunne brukes videre for å kontrollere geometrien.

Det å gjennomføre alle mot alle sammenligninger, som gjort i programmet her krever mye arbeidskraft, og vil ikke være en god løsning dersom datasettene er store. Om det skulle være ønskelig å gjennomføre en slik test vil det være lurt å dele opp geometrien i indekser. Dette vil gjøre at geometri-par som ikke kan ha noen relevante relasjoner ikke blir sjekket.

```
from osgeo import ogr
from shapely.wkt import loads

def geometriFraGML():
    gmlDok = ogr.Open('VegStiForenklet.gml')
    layer = gmlDok.GetLayer()
    geometri = []
    for _ in xrange(layer.GetFeatureCount()):
        feature = layer.GetNextFeature()
        geometri.append(loads(str(feature.GetGeometryRef())))
```

Figur 25: Egenutviklet kode som henter ut all geometri fra GML-dokument

6. Konklusjoner og videre arbeid

6.1 Konklusjoner

De viktigste konklusjonene fra dette arbeidet er at GML-formatet ikke kan kvalitetssikres og kontrolleres med bare de kjente trinnene:

- Syntakskontroll
- Skjemakontroll
- Schematronkontroll

Det vil i tillegg til dette være nødvendig med:

- Utvidet syntakskontroll

Denne utvidete syntakskontrollen skal i tillegg til å sjekke om GML-dokumentet er på velformet XML også sjekke om og schematronskjema er det. Kontrollen vil sjekke om navnerom, referanser, generelle regler, definering av elementer, objekter og geografiske egenskaper (features), og identifisering av egenskaper og samlinger er riktig i filene. Kontrollen skal kunne undersøke om filene er konforme med det som er definert i deres tilhørende standard.

- Ekstra kontroll av geometri

En ekstra kontroll av geometri skal kunne brukes til å undersøke om geometri i GML-dokumentet er definert feil, og om det er geometri i filen som ikke skal være der eller annen interessant informasjon om geometrien i filen.

Dette er nødvendig ettersom at GML-skjema/applikasjonsskjema kun sjekker om geometrien i GML-dokumentet er definert i skjemaene, men sjekker ikke om geometrien i seg selv er riktig.

- Rapportering

Det å rapportere resultatene fra en kontroll av geodata på GML-formatet er viktig. En rapport vil gjøre det enklere for brukeren å finne ut om informasjonen GML-filene er riktige, og det gir brukeren muligheten til å hente ut ønsket informasjon fra GML-dokumentet på en oversiktlig måte.

6.2 Videre arbeid

I denne oppgaven er det beskrevet hvordan GML-formatet skal kvalitetssikres og kontrolleres med utgangspunkt i SOSI-standarden, og det vil være naturlig å jobbe videre med å lage et program som gjør dette. Det vil i hovedsak være to aktuelle måter å gjøre dette på:

- Videreutvikling av programmet SOSI-kontroll
- Lage nytt selvstendig program

Den første muligheten er for Kartverket å videreutvikle SOSI-kontroll, slik at programmet har mulighet for å kontrollere GML-formatet. Dette vil være en grei løsning for oss i Norge, men det vil trolig være ønskelig å lage programvare som er siktet mot internasjonal bruk med engelsk brukergrensesnitt.

Den andre løsningen er å lage et nytt selvstendig program som har den funksjonaliteten som er beskrevet i kapittel 5.3 Programvarens funksjonalitetskrav. Det å lage et slikt program er noe en bedrift med nok ressurser kan gjøre. Verdien av et slikt program vil være størst dersom det gjøres internasjonalt lett tilgjengelig.

7. Referanser

Ahn, Y. S., Park, S. Y., Yoo, S. B., & Bae, H. Y. (2004). Extension of Geography Markup Language (GML) for mobile and location-based applications. In M. L. Gavrilova, V. Kumar, Y. Mun, C. J. K. Tan, & O. Gervasi (Eds.), *Computational Science and Its Applications - Iccsa 2004, Pt 2* (Vol. 3044, pp. 1079-1088).

Chao, L., Xiao, Z., & Zhang, X. (2008). An Integrated Method for GML Application Schema Match. In D. S. Huang, D. C. Wunsch, D. S. Levine, & K. H. Jo (Eds.), *Advanced Intelligent Computing Theories and Applications, Proceedings: With Aspects of Contemporary Intelligent Computing Techniques* (Vol. 15, pp. 39-46).

Detlev Wagner, V. C., Joachim Benner. (2014). *SEMANTIC VALIDATION OF GML-BASED GEOSPATIAL DATA* Stuttgard-University of Applied Sciences, Stuttgard, Germany. Retrieved from https://www.researchgate.net/publication/281585715_Semantic_Validation_of_GML-based_Geospatial_Data

Gao, X., Cui, Z., Yu, J., Cui, W., Zhang, S., Lv, G., & Ieee. (2013). Research on GML Data Validity Verification Based on Stack Mechanism *2013 Sixth International Symposium on Computational Intelligence and Design* (pp. 355-358).

Hu, C. (2010). *Adopted native XML Database to Store and Index GML Spatial Data*. Deyang, China: Sichuan college of architectural technology.

Hugo, L. (2014). *On the validation of solids represented with the international standards for geographic information.* Delf University of Technology, Delf, Netherlands. Retrieved from https://3d.bk.tudelft.nl/hledoux/pdfs/13_cacaie.pdf

INSPIRE. (2010). Guidelines for the encoding of spatial data (pp. 41): INSPIRE.

ISO. (2003). ISO 19107: Geographic information — Spatial schema (pp. 1-178). Genéve, Sveits: International Organization for Standardization.

ISO. (2006). ISO/IEC 19757-3: Information technology — Document Schema Definition Languages (DSDL) — Part 3: Rule-based validation — Schematron (pp. 1-38). Genéve, Sveits: International Organization for Standardization.

ISO. (2007). ISO 19136: Geographic information - Geography Markup Language (GML). Genéve, Sveits: International Organization for Standardization.

ISO. (2009). ISO 19115-2: Geographic information - Metadata Part 2: Extensions for imagery and gridded data. Genéve, Sveits: International Organization for Standardization.

ISO. (2011). ISO 19118: Geographic information - Encoding. Genéve, Sveits: International Organization for Standardization.

ISO. (2012). ISO/IEC 19505: Information technology -- Object Management Group Unified Modeling Language (OMG UML) -- Part 1: Infrastructure. Genéve, Sveits: International Organization for Standardization.

ISO. (2013). ISO 19157: Geographic information - Data Quality (pp. 1-154). Genéve, Sveits: International Organization for Standardization.

ISO. (2014). ISO 19115-1: Geographic information - Metadata Part 1: Fundamentals. Genéve, Sveits: International Organization for Standardization.

ISO. (2015). ISO 19109: Geographic Information - Rules for application schema. Genéve, Sveits: International Organization for Standardization.

Kartverket. (1999). KVAKK - Brukerveiledning. Hønefoss: Statens Kartverk i sammarbeid med Høgskolen i Gjøvik.

Kartverket. (2007). Kontroll av geodata (pp. 1-90). Hønefoss: Kartverket.

Kartverket. (2010). SOSI-kontroll, hjelp dokument. Hønefoss: Kartverket.

Kartverket. (2012). SOSI Del 1 Realisering i SOSI - format og GML (pp. 111). Hønefoss: Kartverket.
Kartverket. (2014). SOSI Generell del: SOSI produktspesifikasjoner - Krav og godkjenning. Hønefoss: Kartverket.
Kartverket. (2015a). Geodatakvalitet (pp. 1-106). Hønefoss: Kartverket.
Kartverket. (2015b). Retningslinjer for dataformater ved formidling av stedsdata. Hønefoss: Kartverket.
Kartverket. (2016). SOSI programmer og verktøy. Retrieved from http://kartverket.no/geodataarbeid/standarder/sosi/programmer-og-verktoy/
Klimek, J., Benda, S., & Necasky, M. (2014). Translation of Structural Constraints from Conceptual Model for XML to Schematron. *Journal of Universal Computer Science, 20*(3), 277-301. Retrieved from <Go to ISI>://WOS:000339390000003
Lake, R. (2005). The application of geography markup language (GML) to the geological sciences. *Computers & Geosciences, 31*(9), 1081-1094.
LAN, X., LU, G., ZHANG, S., & Jiang, Y. (2005). Implementation of Universal GML 3.0 Parsing Engine [J]. *Geo-information Science, 1*, 013.
Maly, J., & Necasky, M. (2015). Model-driven approach to modeling and validating integrity constraints for XML with OCL and Schematron. *Information Systems Frontiers, 17*(4), 917-946. doi:10.1007/s10796-013-9471-4
Min, H. (2010). *GML Validation Based on Norwegian standard.* Høgskolen i Gjøvik, Gjøvik. Retrieved from http://brage.bibsys.no/xmlui/handle/11250/144130
Myer, T. (2005). A good introduction to XML. Retrieved from http://www.sitepoint.com/really-good-introduction-xml/
Nic, M. Schematron Tutorial. Retrieved from http://zvon.org/xxl/SchematronTutorial/General/toc.html
Norge-digitalt. (2015). Veileder for Geography Markup Language (GML) (pp. 1-119). Hønefoss: Kartverket.
OGC. (2007a). OpenGIS Geography Markup Language (GML) Encoding Standard (pp. 1-437): Open Geospatial Consortium.
OGC. (2007b). Revision Notes for OpenGIS® Implementation Specification: Geographic information - Geography Markup Language Version 3.2.1 (3.2.1) (pp. 1-22): Open Geospatial Consortium.
OGC. (2011). OGC® Geography Markup Language (GML) simple features profile Technical Note (2.0) (pp. 1-15): Open Geospatial Consortium.
OGC. (2012a). CityGML UML diagrams (pp. 21): Open Geospatial Consortium.
OGC. (2012b). OGC® City Geography Markup Language (CityGML) Encoding Standard (pp. 344): Open Geospatial Consortium.
OGC. (2012c). OGC® Geography Markup Language (GML) — Extended schemas and encoding rule (pp. 1-91): Open Geospatial Consortium.
OGC. (2016). GML 3.2 (ISO 19136:2007) Conformance Test Suite. Retrieved from http://cite.opengeospatial.org/teamengine/about/gml32/3.2.1/site/
van den Brink, L., Stoter, J., & Zlatanova, S. (2013). UML-Based Approach to Developing a CityGML Application Domain Extension. *Transactions in Gis, 17*(6), 920-942. doi:10.1111/tgis.12026
W3C. (2012). Associating Schemas with XML documents 1.0 (Third Edition): World Wide Web Consortium.

Vedlegg A – Kontrollprosessen

Kontrollplan

En kontrollplan består av en rekke trinn som tidligere sett i figur 7. Under er en oppsummering av disse trinnene fra Kartverket sin standard for geodatakvalitet (Kartverket, 2015a)[2].

Valg av kvalitetselement og omfang

I dette trinnet må brukeren velge hvilken kvalitetsenhet som kontrollen skal gjennomføres for. Denne kvalitetsenheten er en kombinasjon av kvalitetselementer og et omfang.

Et omfang er et utvalgt av data som representerer kontrollen, denne dataen skal være mest mulig homogen. Omfanget kan består av ett eller flere komplette datasett, et utvalg av geografiske områder innen disse datasettene, bestemte objekttyper eller andre utvalg.

Valg av kvalitetsmål

Her bestemmes kvalitetsmålet som angir hva resultatet av kontrollen skal vise. En oversikt over kvalitetsmål finnes i Kartverket sin standard for geodatakvalitet (Kartverket, 2015a).

Valg av kontrollmetode

I dette trinnet bestemmes hvilken kontrollmetode som skal brukes. En kontroll kan bruker forskjellige typer kontrollmetoder avhengig av hva som skal kontrolleres.

Gjennomføring av kontroll

På lik linje med kontrollplanen så er det en rekke trinn i gjennomføringen av en kontroll. Under er en oppsummering av disse trinnene fra Kartverket sin standard for geodatakvalitet(Kartverket, 2015a).

Utføring av kontrollmålinger

I dette trinnet utføres kontrollmålinger for å fremskaffe fasitdata. Tilsvarende verdier hentes så fra datasettet, og det utføres beregninger mellom fasitverdien fra kontrollmålingene og datasettets verdi. I dette trinnet beregnes også verdien for det aktuelle kvalitetsmålet for hele omfanget.

Identifisering av krav

Her settes krav for de ulike kvalitetsmålene. Dette blir hentet fra produktspesifikasjonen for datasettet som benyttes. I tilfellene hvor det ikke eksisterer noen produktspesifikasjon blir resultatene fra kontrollen direkte rapportert.

Sammenlikning av beregnede verdier for kvalitetsmål mot krav

I dette trinnet sammenliknes kravet for kvalitetsmålet med verdien fra kontrollen. Dette skal gi et resultat som ikke skal forkastes med mindre det er signifikant dårligere enn kravet.

Godkjenning og avviksbehandling

Dette trinnet gjennomføres etter at kontrollen av datasettet er fullført. Her bestemmes det om datasettet kan godkjennes, ikke godkjennes eller om kontrollen må bli utvidet.

[2] Referanser i vedleggene henviser til rapportens referanseliste.

Rapportering
Her blir kontrollen rapportert.

Kontrollmetoder for geodata
Klassifiseringen av kontrollmetoder avhenger av hvilken data som er nødvendig for kontrollen. Metodene det skilles mellom er direkte metoder, indirekte metoder og aggregering. De direkte metodene deles videre opp avhengig av hvilke type data (interne eller eksterne) som brukes. En direkte kontroll vil deretter deles opp igjen i enten en full kontroll hvor man ser på hele omfanget eller en utvalgsbasert kontroll (Kartverket, 2015a).

Direkte metoder
En direkte kontrollmetode er en metode som anvendes direkte på datautvalget som skal kontrolleres. Uavhengig om kontrolldataen som blir brukt er intern eller ekstern, så vil en direkte metode enten gjennomføre en full kontroll eller en basert på et utvalg.

Full kontroll: En full kontroll brukes når en ønsker å kontrollere alle forekomster innen omfanget. Dette egner seg veldig godt dersom kontrollen skal gjennomføres av program vare.

Utvalgsbasert kontroll: En utvalgsbasert kontroll brukes når en ønsker å kontrollere et representativt utvalg av dataene. Dette brukes ofte når det er objekttyper som representerer hele omfanget av kontroller og/eller ved stikkprøveområder.

Kontroll mot interne data: Brukes når man skal bruke kontrolldata fra selve datasettet til å kontrollere dataene.

Eksempler på metoder for kontroll mot interne data(Kartverket, 2015a):

- Kontroll med programvare (f.eks. SOSI-kontroll)
- 3D-visualisering

Kontroll mot eksterne data: Bukes når man skal bruke kontrolldata utenfor datasettet for til å kontrollere dataene.

Eksempler på metoder for kontroll mot eksterne data (Kartverket, 2015a):

- Visuell kontroll mot ortofoto
- Kontroll mot andre data, f.eks. laserdata
- Synfaring
- Landmåling
- Fotogrammetrisk kontroll i DFA

Indirekte metoder
Det er også mulig å vurdere datakvaliteten ut ifra andre kilder enn selve dataene som blir brukt, dette kalles en indirekte metode. Informasjon som kan fungere for å vurdere

datakvaliteten istedenfor kan være kilder som metadata, produksjonsrapporter, kjennskap til opprinnelse og produksjonsmetoder.

Det kan i enkelte tilfeller være vrient å rapportere datakvaliteten til indirekte evaluert data som et kvantitativt resultat. Det må i disse tilfellene bli beskrevet som et beskrivende resultat ved hjelp av tekst.

Dersom man skal kunne bruke indirekte metoder, så må man har god informasjon om kvaliteten til de kildene som brukes. Dersom denne informasjonen er god, så gir de indirekte metodene mulighet for en skjønnsmessig vurdering og gir derfor ikke muligheten til å måle og tallfeste kvaliteten til et datasett. En indirekte metode vil derfor ikke være egnet dersom en ønsker å kvantifisere kvaliteten til geodata opp mot kravene i en produktspesifikasjon.

Eksempler på indirekte kontroller:

- Vurdering av fotogrammetrirapporter
- Vurdering av landmålingsrapporter

Aggregering

Dersom man har resultater fra andre kontroller (direkte eller indirekte), så kan en benytte metoden aggregering for å kombinere resultatene. Resultatet av aggregeringen kan deretter brukes til å gi en samlet evaluering av datasettet.

Vedlegg B – SOSI-kontroll

Under er en litt mer detaljert oppsummering av trinnene i SOSI-kontrollen. Sjekk ut SOSI-kontrollen sitt hjelp dokument (OGC, 2007a) for en mer detaljert beskrivelse av trinnene.

1. Formell formatsjekk
   1.1 Hodekontroll
      - Skriver ut hodegruppen på rapportfilen med en del kommentarer til enkelte opplysninger.
      - Sjekker at hodegruppen inneholder all obligatorisk informasjon.
      - Sjekker at syntaksen på hodeinformasjonen er korrekt.
      - Sjekker at det ikke er benyttet udefinerte basiselement.
      - Sjekker om enkelte kontroller må kobles ut p.g.a ufullstendig hodeinformasjon.
      - Sjekker at oppgitt SOSI-nivå er lovlig (1-4).

   1.2 Multiple hoder
      - Sjekker at det er kun en hodegruppe på fila.

   1.3 Tegnsettkontroll

   1.4 Sluttkontroll
      - Finner posisjonen for .SLUTT dersom merket finnes.
      - Sjekker om det finnes informasjon mellom logisk sluttmerke (.SLUTT) og fysisk sluttmerke (End Of File).
2. Gruppevis innholdssjekk
   2.1 Serienummer
      - Kontrollerer at alle gruppene på SOSI-filen har unike serienummer.

   2.2 Område
      - Finner max/min nord/øst koordinater på fila.
      - Sjekker om det finnes koordinater utenfor området definert i .HODE.
      - Beregner areal av område oppgitt i .HODE og areal som filen dekker.
      - Beregner utstrekning av område oppgitt i .HODE og utstrekning som filen dekker.
      - Sjekker at område opplysninger ikke er oppgitt speilvend. (max < min).
      - Kobler ut videre kontroll dersom område opplysninger skulle vise seg å være speilvendte.

   2.3 SOSI-nivå (SOSI-NIVÅ er ikke lenger obligatorisk i hode på SOSI-filen)

   2.4 Nøyaktighet/enhet
      - Kontrollerer at nøyaktighet og enhet oppfyller kravet spesifisert.

   2.5 Høydeinformasjon
      - Lager statistikk over høydeinformasjon på grunnlag av gruppene på fila.
      - Påviser grupper der det finnes kombinasjoner av ..NØ og/eller ..NØH og/eller ..HØYDE.
      - Påviser data uten høydeverdier, men som er registrert med måleinstrument som kan gi høydeverdier.
      - Påviser data med høydeverdier, men som mangler nord og øst koordinater.

   2.6 SOSI-syntaks

- Kontrollerer riktig syntaks og lager statistikk over basiselementer.

3. Knutepunktskontroll

   3.1 Knutepunkt/noder

   - Lager en statistikk over punkter markert med ...KP på den aktuelle SOSI-filen.
   - Grupperer nodene i knutepunktslag.
   - Grupperer knutepunkter i noder.
   - Grupperer punkter i forskjellige kategorier.
   - Sjekker om blindnoder befinner seg på områdekant. (blindnode : 1'er node).

   3.2 Grupper som kan slåes sammen

   - Undersøker alle 2'er noder og SOSI-grupper der disse inngår om de har like egenskaper.
   - Presenterer en liste over grupper som kan slås sammen til en gruppe.

   3.3 Knyttbare punkter

   - Lager en statistikk over punkter som kan gjøres om til knutepunkter markert med ..KP. Benytter forskjellige toleranser.

4. Flatekontroll

   4.1 Referanser

   - Kontrollerer at refererte grupper finnes på fila.

   4.2 Flater

   - Kontrollerer at flategrenser henger sammen med identiske endepunkter.
   - Påpeker grenser som ikke henger sammen i punkt markert med ...KP.
   - Teller antall flater som har flere enn 2000 punkter i grensene, og skriver ut serienummer til flaten med flest punkter. Toleransegrensen på 2000 kan endres.

   4.3 Representasjonspunkt

   - Kontrollerer at flatens representasjonspunkt befinner seg innenfor flatens grenser. Hvis flaten mangler representasjonspunkt, er dette en feil som blir avslørt i syntakskontrollen (Test 2.6).

5. Objektkontroll

   5.1 Objektkontroll

   - Kontrollerer egenskaper og kodeverdier for objekter på 2 (gruppenivå) og 3 prikks-nivå (punktinfo.).

   5.2 Geometrityper

   - Kontrollen sjekker om gruppeelementet er av riktig geometritype.

   5.3 Min./maks. multiplisitet/kardinalitet

   - Sjekker min. og maks. multiplisitet/kardinalitet på SOSI-gruppeobjektene.

   5.4 Dobbel geometri

   - Sjekker og feilmelder dobbel geometri.

   5.5 Flateavgrensing

   - Kontrollerer om flaten avgrenses av korrekte linjer.

6. Statistikk

   6.1 Objekter

   - Klassifiserer alle gruppeelementene på SOSI-fila i objektklasser.

6.2 Antall punkt pr.linje/kurve
6.3 Gjennomsnitt linjelengder
6.4 Punkttetthet
6.5 Punkter spesifisert på temakoder
6.6 Kvalitetsnoder

7. Rapporter
   7.1 Kontrollrapport
      - Diverse informasjon (testdato, versjon av dll'en o.s.v).
      - Formell formatsjekk (rapporter fra kontrollene 1.1 til 1.4).
      - Gruppevis innholdssjekk (rapporter fra kontrollene 2.1 til 2.6).
      - Knutepunkt kontroll (rapporter fra kontrollene 3.1 til 3.3).
      - Flate kontroll (rapporter fra kontrollene 4.1 til 4.3).
      - Objektkontroll (rapporter fra kontrollene 5.1 til 5.3).
      - Statistikk (kontroll 6.1.1 til 6.3.1).
   7.2 Statistikkrapportene L1, L2 og L3
      - nn.L1: Volumtabell, En oversikt over antall grupper og koordinater av hvert grafisk element for hver SOSI-fil eller filliste.
      - nn.L2: Kodetabell, Alle SOSI-navn og koder i kjøringen med antall ganger de forekommer og forklaringer av kodene.

nn.L3: Hjelpetabell, Alle SOSI-navn og koder sammen med verdi og antall forekomster og første filen hvor de forekommer.

Vedlegg C – Kvalitetselementer (ISO)

Fullstendighet

Datakvalitetselementet *fullstendighet* er definert som tilstedeværelsen og fraværet av egenskaper, attributter og relasjoner i et datasett. Elementet deles inn i datakvalitetselementet «kommisjon», som er overflødig informasjon i datasettet og «utelatelse», som er manglende informasjon i datasettet.

Logisk konsistens

Datakvalitetselementet *logisk konsistens* er definert som den graden logiske regler for data struktur, navngiving og relasjoner blir fulgt. Datastruktur kan være konseptuell, logisk eller fysisk, så denne graden vil variere avhengig av datasettet. De logiske reglene er ikke alltid dokumentert direkte i datasettet et blir jobbet med og det vil i disse tilfellene være nødvendig å referere til reglenes plassering. Elementet deles videre inn i fire datakvalitetselementer:

- «Konseptuell konsistens»: ser på om reglene til det konseptuelle skjemaet blir fulgt.
- «Domene konsistens»: tar for seg verdien til verdi domenene.
- «Format konsistens»: ser på hvilken grad data er lagret i overenstemmelse med den fysiske strukturen til datasettet.
- «Topologisk konsistens»: ser på riktigheten til den kodede topologiske karakteristikken for datasettet.

Posisjonsnøyaktighet

Datakvalitetselementet *posisjonsnøyaktighet* er definert som nøyaktigheten til posisjonen til egenskaper innen et romlig referansesystem. Dette elementer deles videre inn i tre datakvalitetselementer:

- «Absolutt eller ekstern nøyaktighet»: ser på nærheten til rapporterte koordinatverdier og verdier som aksepteres eller er gyldige.
- «Relativ eller intern nøyaktighet»: ser på nærheten til relative posisjoner av egenskaper i datasettet til deres respektive relative posisjoner som er akseptert eller er gyldige.
- «Posisjonsnøyaktigheten til rasterdata»: ser på nærheten av rasterdata sine romlige posisjonsverdier til verdier som er akseptert eller er gyldige.

Tematisk nøyaktighet

Datakvalitetselementet *tematisk nøyaktighet* er definert som nøyaktigheten til kvantitative attributter, korrektheten til ikke-kvantitative attributter og klassifiseringen av egenskaper og deres relasjoner. Dette elementet deles inn i tre datakvalitetselementer:

- «Klassifiserings korrekthet»: sammenligner klassene som er tildelt egenskaper og deres attributter med referansedata.
- «Ikke-kvantitative attributters nøyaktighet»: finner et mål på om de ikke-kvantitative attributtene er korrekte eller ikke.

- «Kvantitative attributters nøyaktighet»: finner hvor nært verdiene til kvantitative attributter er i forhold til verdier som er akseptert eller sett på som gyldige.

Temporal kvalitet

Datakvalitetselementet *temporal kvalitet* er definert som kvaliteten til temporale attributter og temporale relasjoner. Elementet deles inn i tre datakvalitetselementer:

- «Nøyaktigheten til en tidsmåling»: ser på hvor nærme den rapporterte tidsmålingen er en verdi som er akseptert eller gyldig.
- «Temporal konsistens»: ser på korrektheten til rekkefølgen med hendelser.
- «Temporal gyldighet»: ser på gyldigheten til data med hensyn på tid.

Vedlegg D – Kontroll av GML-dokument

Under er en liste over hva som burde inneholde, samt hvordan dette skal testes ved kontroll av GML-dokumenter ifølge OGC sin standard fr 2007 (OGC, 2007a):

1. Referanse til GML-applikasjonsskjema
    - Testens formål: Verifisere at det eksisterer en referanse til et GML-applikasjonsskjema som kan anvendes på GML-dokumentet.
    - Testmetode: Sjekke at et XML-skjema som representerer et GML-applikasjonsskjema er blitt referert til ved bruk av en xsi:schemaLocation attributt i rotelementet til GML-dokumentet.
2. Eksistens av GML-applikasjonsskjema
    - Testens formål: Verifisere at GML-applikasjonsskjema som det er blitt referert til eksisterer.
    - Testmetode: Sjekke at XML-skjemaet som representerer GML-applikasjonsskjemaet det er blitt referert til i GML-dokumenter er mulig å få tilgang til. Sjekk også at det er tilgang til alle dokumenter som det er blitt referert til direkte eller indirekte i referansefilen.
3. Konformitet med GML-applikasjonsskjemaene det har blitt referert til
    - Testens formål: Verifisere at GML-applikasjonsskjemaet som det har blitt referert til i GML-dokumenter er konform med den internasjonale standarden til OGC (OGC, 2007a).
    - Testmetode: Verifisere at applikasjonsskjemaet kommer gjennom alle testene som er spesifisert i ISO 19136:2007 A.1 (ISO, 2007).
4. Gyldig XML
    - Testens formål: Verifisere gyldigheten til GML-dokumentet mot XML-skjema komponenter i overenstemmelse med GML-applikasjonsskjemaet.
    - Testmetode: Valider GML-dokumentet mot GML-applikasjonsskjemaet det er referert til. Denne prosessen må enten bli gjennomført av egnet verktøy for validering eller manuelt ved å sjekke alle de relevante definisjonene fra XML-skjema spesifikasjonene.
5. Konformitet av et GML-dokument
    - Testens formål: Kontrollere at GML-dokumentet er i samsvar med alle begrensningene satt i OGC sin internasjonale standard (OGC, 2007a).
    - Testmetode: Sjekk at kravene «referanse til GML-applikasjonsskjema» og «konformitet med GML-applikasjonsskjemaene det har blitt referert til» er godkjente. Sjekk at GML-dokumentet følger «gyldig XML» og at det er i samsvar med andre begrensninger spesifisert i den internasjonale standarden.

Vedlegg E – Kontroll av GML-applikasjonsskjema

Under er en liste over hva som burde inneholde, samt hvordan dette skal testes ved kontroll av GML-applikasjonsskjemaer ifølge OGC sin standard fra 2007 (OGC, 2007a):

1. Bruk av XML navnerom
   - Testens formål: Verifisere riktig bruk av XML navnerom i et GML applikasjonsskjema
   - Testmetode: Kontrollere at alle komponentene i applikasjonsskjemaet er forbundet med et XML-navnerom og at navnerommet ikke er «http://opengis.net/gml/3.2»
2. Generelle regler
   - Testens formål: Verifisere at GML-applikasjonsskjemaet følger de generelle reglene for konstruksjon av GML-applikasjonsskjemaer.
   - Testmetode: Inspisere applikasjonsskjemaet og sjekk at det tilfredsstiller de generelle reglene som er beskrevet i ISO 19136:2007, 21.2.1 (ISO, 2007).
3. Import av komponentene til GML-skjema
   - Testens formål: Verifisere at GML-applikasjonsskjemaet importerer hele GML-skjemaet og refererer GML-profiler riktig.
   - Testmetode: Inspiser import uttalelser i applikasjonsskjemaet (hele GML-skjema må være direkte eller indirekte importert). Hvis det blir referert til en eller flere GML-profiler, så må det i tillegg sjekkes om XML-skjemakomponenter som er spesifisert i gml:gmlProfileSchema tilfredsstiller alle obligatoriske konformitets krav.
4. Gyldig XML-skjema
   - Testens formål: Verifisere gyldigheten til GML-applikasjonsskjemaet (XML-dokumentet) mot spesifikasjonene til XML-skjemaer.
   - Testmetode: Validere XML-dokumentet (GML-applikasjonsskjemaet) opp mot XML-skjema spesifikasjonene. Denne prosessen må enten bli gjennomført av egnet verktøy for validering eller manuelt ved å sjekke alle de relevante definisjonene fra XML-skjema spesifikasjonene.
5. Støtte for GML modellen og syntaks
   - Testens formål: Kontrollere at GML-applikasjonsskjemaet følger reglene for koding av objekter og egenskaper.
   - Testmetode: Sjekke applikasjonsskjemaet.
6. Substitution group of object elements, type derivation
   - Testens formål: Verifisere at alle objektene i GML-applikasjonsskjemaet er i korrekt substitusjonsgruppe.
   - Testmetode: Sjekke at alle objektelementer med identitet (direkte eller indirekte) i applikasjonsskjemaet er i riktig substitusjonsgruppe av gml:AbstractGML. Sjekke at reglene for avledning fra grunntyper blir fulgt.
7. Eiendomselementer er ikke objektelementer
   - Testens formål: Verifisere at alle eiendomselementer i GML-applikasjonsskjemaet ikke er objekter.

- Testmetode: Sjekke at alle barnlementer for hvert objektelement i applikasjonsskjemaet ikke er direkte eller indirekte i substitusjonsgruppen gml:AbstractObject.

8. Modell for innholdet til eiendomselementer
   - Testens formål: Verifisere at eiendomselementer i GML-applikasjonsskjemaet har gyldig modell for innholdet.
   - Testmetode: Sjekke hvert barnelementer i hvert objektelement i applikasjonsskjemaet.
9. Egenskaper for metadata og data kvalitet
   - Testens formål: Verifisere at alle egenskapene hvor verdien er metadata for et objekt kan bli indentifisert som metadataegenskap.
   - Testmetode: Sjekke at innholdsmodellen for alle eiendomselementer i GML-applikasjonsskjemaet hvor metadata har verdi er avledet fra utvidelsen gml:AbstractMetadataPropertyType.
10. Egenskaper for romlig geometri
    - Testens formål: Verifisere at alle egenskapene hvor verdien er et romlig geometrisk objekt kan bli identifisert som det.
    - Testmetode: Sjekke at alle egenskapene som er et geometrisk objekt eller en sammensetting av slike objekter er deklarert i GML-applikasjonsskjemaet i henhold til ISO 19136:2007, 9.5 (ISO, 2007).
11. Egenskaper for romlig topologi
    - Testens formål: Verifisere at alle egenskaper hvor verdien er et romlig topologisk objekt kan bli identifisert som det.
    - Testmetode: Sjekke at alle egenskapene som er et topologiske objekt eller en sammensetting av slike objekter er deklarert i GML-applikasjonsskjemaet i henhold til ISO 19136:2007, 9.6 (ISO, 2007).
12. Temporale egenskaper
    - Testens formål: Verifisere at alle egenskaper hvor verdien er et temporalt objekt kan bli identifisert som det.
    - Sjekke at alle egenskapene som er et temporalt objekt eller en sammensetting av slike objekter er deklarert i GML-applikasjonsskjemaet i henhold til ISO 19136:2007, 9.7 (ISO, 2007).
13. Egenskaper for beliggenhet
    - Testens formål: Verifisere at alle egenskaper hvor verdien er en referanse/beskrive av beliggenhet kan bli identifisert som det.
    - Testmetode: Sjekke at alle egenskapene med romlige referanser gitt av en geografisk identifikasjon i GML-applikasjonsskjemaet bruker eiendomselementet gml:locationName eller gml:locationReference.
14. GML objektsamlinger
    - Testens formål: Verifisere at alle objektene som er en sammensetting av GML objekter kan bli identifisert som det.
    - Testmetode: Sjekke at slike objekter i GML-applikasjonsskjemaet har en eller flere eiendomselementer med et en innholdsmodell fra gml:AbstractMemberType. Sjekk

også at i tilfeller hvor det er passende at gml:aggregationType attributtene er barnenoder til objektelementer.

15. Substitution group of feature elements
    - Testens formål: Verifisere at alle features i et GML-applikasjonsskjema er i sin riktige substitusjonsgruppe.
    - Testmetode: Sjekke at alle objektelementer som representerer features i GML-applikasjonsskjemaet er enten direkte eller indirekte i substitusjonsgruppen av typen gml:AbstractFeature.
16. GML «feature» samlinger
    - Testens formål: Verifisere at alle features som er sammensettinger av GML-features kan bli identifisert som det.
    - Testmetode: Sjekke at slike features i GML-applikasjonsskjemaet har en eller flere eiendomselementer med en innholdsmodell fra gml:AbstractFeatureMemberType. Sjekk også at i tilfeller hvor det er passende at gml:aggregationType attributtene er barnenoder til objektelementer.

Vedlegg F – Schematron implementering

For mer informasjon om schematronskjema, les ISO/IEC 19757-3:2006 (ISO, 2006). Her forklares det hvordan skjemaene skal lages og brukes. Under er et sammendrag av de viktigste delene av standarden.

Syntaks:

1. Velformet: Et schematronskjema skal være et *velformet* XML-dokument.
2. Navnerom: Alle elementer i schematron er kvalifisert med navnerommet URI: *http://purl.oclc.org/dsdl/schematron*.
3. Mellomrom (*whitespace*): Ethvert element kan ha barne-strenger som kun inneholder mellomrom, hvor mellomrom er representert med bokstavene U+0020, U+0009, U+000D eller U+000A. Det vil her ikke være noe begrensninger til deres relative posisjon i henhold til barne-elementene.
4. Kjerneelementer som brukes i schematron (se ISO/IEC 19757-3:2006 (ISO, 2006) for mere informasjon): **active** element, **assert** element, **extends** element, **include** element, **let** element, **name** element, **ns** element, **param** element, **pattern** element, **phase** element, **report** element, **rule** element, **schema** element, **value-of** element.
5. Hjelpe elementer og attributter som brukes i schematron (se ISO/IEC 19757-3:2006 (ISO, 2006) for mere informasjon): **diagnostic** element, **dir** element, **emph** element, **flag** attributt, **fpi** attributt, **icon** attributt, **p** element, **role** attributt, **see** attributt, **span** element, **subject** attributt, **title** element.

Semantikk:

1. Validerings funksjon: En generell schematron validerer er en funksjon som gir tilbake «valid», «invalid» eller «error». Funksjonen gjennomfører to steg ved å først transformere skjemaet til *minimalistisk syntaks* for så å teste instansen mot den *minimalistiske syntaksen*.
   En schematron validering består av en funksjon som validerer:
   - Et spørrespråk binding
   - Et skjema dokument
   - En instans som skal bli validert
   - Eksterne XML-dokumenter som har blitt adressert ved bruk av informasjon i instansen eller skjemaet
   - Et fasenavn, #ALL om alle mønstre skal bli aktive mønstre eller #DEFAULT om *defaultPhase* attributtet til skjema-elementet skal brukes.
   - En liste med navneverdipar dersom skjemaet bruker eksterne variabler
2. Minimalistisk syntaks: For å forenkle beskrivelsen av semantikk senere, så blir følgende transformasjonstrinn først tilført til det aktuelle skjemaet. Trinnene resulterer i et skjema med *minimalistisk syntaks*:
   - Løse alle slutninger ved å erstatte **include** elementet med resursen som det er linket til.
   - Løse alle abstrakte mønstre ved å bytte ut parameter referanser med aktuelle parameterverdier i alle lukkede attributter som inneholder spørsmål.

- Løse alle abstrakte regler i skjemaet ved å bytte ut **extends** elementene med innholdet som de abstrakte reglene identifiserer.
- Endre alle elementer av formen **report** til **assert** elementer.
- Fjerne elementer brukt for kjennetegn og dokumentasjon.

Det resulterende skjemaet på *minimalistisk syntaks* vil også være en gyldig schematron instans i den fulle syntaksen.

3. Skjema semantikk: Avsnittet 6.3 i ISO 19757-3:2006 viser semantikken til godt skjema som har blitt endret til minimalistisk form, se standarden for dette(ISO, 2006).
4. Spørrespråk binding (Query Language Binding): En spørrespråk binding skal inneholde følgende:
    - Det generelle spørrespråket som er brukt. En navn indikator som indentifiserer spørrespråket. Data modellen.
    - En påstandstest. Dette er en funksjon som returnerer dataverdier til boolske uttrykk.

    En spørrespråk binding kan også inneholde en rekke andre tilleggsfunksjoner, dette står det mer om i ISO 19757-3:2006 (ISO, 2006).

5. Rekkefølge og bivirkninger:
    - Rekkefølgen som elementer blir validert er implementeringsavhengig, uten å endre gyldigheten til instansen.
    - Rekkefølgen som mønstre brukes er implementeringsavhengig, uten å endre gyldigheten til instansen.
    - Rekkefølgen som påstander testes er implementeringsavhengig, uten å endre gyldigheten til instansen.
    - Det eneste elementet som er avhengig av rekkefølge er **rule** og **let** elementene.

Konformitet:

1. Enkel konformitet: implementasjonen av schematron klare å rapportere for alle XML-dokumenter at strukturen ikke er i samsvar med kravene som angir et gyldig schematron-skjema.
    - Et gyldig skjema er i overenstemmelse med begrensningene satt i Annex A i ISO/IEC 19757-2 skjemaet for denne delen av den internasjonale standarden(ISO, 2006).

    En implementasjon med enkel konformitet skal klare å bestemme for ethvert XML-dokument og for ethvert godt skjema om dokumentet er gyldig i forhold til skjemaet.
2. Full konformitet: En implementasjon av schematron med full konformitet skal klare å bestemme for alle XML-dokumenter om det er et riktig skjema.
    - Et gyldig skjema er i overenstemmelse med begrensningene satt i Annex A i ISO/IEC 19757-2 skjemaet for denne delen av den internasjonale standarden(ISO, 2006).
    - Et gyldig skjema er i samsvar med begrensingen satt i Annex B i ISO/IEC 19757-3 (ISO, 2006).

- En riktig skjema attributt er i samsvar med grammatikken som er spesifisert i spørrespråk bindingen som er brukt.
- Et riktig skjema har kun en definisjon i omfanget for ethvert variabelnavn i enhver sammenheng.
- Verdiene til attributtene **flag**, **id**, **name** og **prefix** er velformede navn i versjonen av XML som blir brukt.

En implementasjon med full konformitet skal klare å bestemme for ethvert XML-dokument og for ethvert godt skjema om dokumentet er gyldig i forhold til skjemaet.

Vedlegg G – Python kode

```
# -*- coding: utf-8 -*-
__author__ = 'Henrik G. Schüller'
__email__ = 'henriksc@nmbu.no'

from lxml import etree, html
from osgeo import ogr
from shapely.wkt import loads
import sys
from PyQt4 import QtCore, QtGui, uic
import threading

form_class = uic.loadUiType("gmlkontroll.ui")[0]


class MyWindowClass(QtGui.QMainWindow, form_class):
    def __init__(self, parent=None):
        QtGui.QMainWindow.__init__(self, parent)
        self.setupUi(self)

        # Kontroll
        self.linearing = []
        self.polygon = []
        self.shapely = []

        # GUI
        self.droppMeny.activated.connect(self.droppMeny_activated)
        self.opplastKnapp.clicked.connect(self.opplastKnapp_clicked)
        self.opplastKnapp2.clicked.connect(self.opplastKnapp2_thread)
        self.valgKnapp.clicked.connect(self.valgKnapp_clicked)
        self.nr = 0
```

```
    def opplastKnapp_clicked(self):
        """
        Bestemmer hva som skal skje når knappen "GML-dokument" blir trykket på
        :return:
        """
        gmlDok = QtGui.QFileDialog.getOpenFileName()
        self.fileBrowser.setText(gmlDok)
        self.fil = open(gmlDok, 'r')
        self.ogrMetode = ogr.Open(str(gmlDok))
        self.lag = self.ogrMetode.GetLayer()
        self.feature = self.lag.GetNextFeature()
        self.gml = etree.parse(self.fil)
        self.lagGeometri()

    def opplastKnapp2_clicked(self):
        """
        Bestemmer hva som skal skje når knappen "Skjemaer" blir trykket på
        """
        skjemanavn = QtGui.QFileDialog.getOpenFileName()
        self.fileBrowser2.setText(skjemanavn)
        self.skjemafil = open(skjemanavn, 'r')
        self.skjema = etree.parse(self.skjemafil)
        self.progress.setText('Laster inn skjema...')
        self.xmlSkjema = etree.XMLSchema(self.skjema)
        self.progress.setText('Klar for kontroll...')

    def opplastKnapp2_thread(self):
        """
        Tråd for at ikke programmet skal henge seg mens XML-skjemaet blir definert av lxml (QThread til PyQT anbefales,
        men grunnet mangel av tid ble det gjort slik)
        """
```

```
        t = threading.Thread(target=self.opplastKnapp2_clicked)
        t.start()


    def droppMeny_activated(self, nr):
        """
        Finner ut hvilke element som er valgt i droppdown menyen
        """
        self.nr = nr


    def valgKnapp_clicked(self):
        """
        Bestemmer hva som skal skje når man trykker på "kjør kontroll"-knappen etter valg i droppdown menyen
        """
        if self.nr == 0:
            self.utskriftBoks.setText(str(self.skjemakontroll()))
        elif self.nr == 1:
            self.utskriftBoks.setText(str(self.geometriBasics()))
        elif self.nr == 2:
            self.utskriftBoks.setText(str(self.geometriAntall()))
        elif self.nr == 3:
            self.utskriftBoks.setText(str(self.geometriAvansert()))


    def lagGeometri(self):
        """
        Lager geometri utifra informasjonen i GML-dokumentet som blir lastet inn - Får feilmelding om geometri er feil
        """
        self.shapely = []
        self.linearing = []
        self.polygon = []


        # Henter ut geometri fra gml-dokument
```

```
        if self.feature.GetGeometryRef():
            for _ in xrange(self.lag.GetFeatureCount() - 1):
                feature = self.lag.GetNextFeature()
                self.shapely.append(loads(str(feature.GetGeometryRef())))
        else:
            # Henter ut polygoner fra en egendefinert fil hvor stien er som følger
            for geography in self.gml.findall('{http://www.opengis.net/gml/3.2}featureMember'):
                for polygon in geography.findall('{http://www.opengis.net/gml/3.2}Polygon'):
                    for coordinates in
polygon.findall("{http://www.opengis.net/gml/3.2}outerBoundaryIs/"
                                            "{http://www.opengis.net/gml/3.2}LinearRing"):

self.linearing.append((coordinates.findtext("{http://www.opengis.net/gml/3.2}coordinates")))


            # Lager polygoner utifra informasjonen som er hentet - OGR/GDAL
            for poly in self.linearing:
                gml =
"""<gml:Polygon><gml:outerBoundaryIs><gml:LinearRing><gml:coordinates>""" + poly + \

"""</gml:coordinates></gml:LinearRing></gml:outerBoundaryIs></gml:Polygon>"""
                self.polygon.append(ogr.CreateGeometryFromGML(gml))


            # Lager polygoner utifra wkt laget i OGR/GDAL eksempelet - Shapely
            for poly in self.polygon:
                self.shapely.append(loads(str(poly)))


    def geometriBasics(self):
        """
        Finner informasjon om geometrien hver for seg
        :return: Informasjon om geometrielementene
        """
        self.progress.setText('Kontrollerer geometri - Enkle forhold...')
        geometriInfo = []
```

```
        # Skriver ut diverse informasjon om polygonene - Shapely
        for geometri in self.shapely:
            geom = geometri
            areal = 'Arealet er:', geometri.area
            lengde = 'Lengden er:', geometri.length
            geometritype = 'Geometritype:', geometri.geom_type
            zVerdi = 'Geometrien har Z-verdi:', geometri.has_z
            gyldigGeometri = 'Geometrien er gyldig:', geometri.is_valid
            krysser = 'Geometrien krysser ikke seg selv:', geometri.is_simple

            geometriInfo.append(str(geom) + '\n' + str(areal) + '\n' + str(lengde) + '\n' +
str(geometritype)
                                + '\n' + str(zVerdi) + '\n' + str(gyldigGeometri) + '\n' + str(krysser) + '\n')

        self.progress.setText('Kontroll ferdig...')

        return '\n'.join(geometriInfo)

    def geometriBasics2(self):
        """
        Finner informasjon om geometrien hver for seg
        :return: Informasjon om geometrielementene
        """
        self.progress.setText('Kontrollerer geometri - Enkle forhold...')
        areal = 0
        lengde = 0

        geometriInfo = []
        # Skriver ut diverse informasjon om polygonene - Shapely
        for geometri in self.shapely:
            areal += geometri.area
            lengde += geometri.length
```

```
        self.progress.setText('Kontroll ferdig...')

        return str(areal/len(self.shapely)) + '\n' + str(lengde/len(self.shapely)) + '\n'


    def geometriAntall(self):
        """
        Ser på geometrielementene mot hverandre - For store datasett vil det være behov for "threading" her også
        :return: Informasjon om geometrielementene
        """
        self.progress.setText('Kontrollerer geometri - Antall...')

        geometriAntall = len(self.shapely)
        intersect = 0
        tomteGrenser = 0
        inni = 0

        for i in xrange(len(self.shapely)):
            for j in xrange(i + 1, len(self.shapely)):
                if self.shapely[i].intersects(self.shapely[j]):
                    intersect += 1
                if self.shapely[i].touches(self.shapely[j]):
                    tomteGrenser += 1
                if self.shapely[i].within(self.shapely[j]):
                    inni += 1

        self.progress.setText('Kontroll ferdig...')

        return 'Geometri antall: ' + str(geometriAntall) + '\n' + 'Antall geometri-par som krysser hverandre: ' + \
```

```
            str(intersect) + '\n' + 'Geometri med deler som ligger langs med hverandre (f.eks. tomtegrenser): ' + \
            str(tomteGrenser) + '\n' + 'Geometri med deler som ligger inni hverandre: ' + str(inni)


    def geometriAvansert(self):
        """
        Ser på geometrielementene mot hverandre - For store datasett vil det være behov for "threading" her også
        :return: Informasjon om geometrielementene
        """
        self.progress.setText('Kontrollerer geometri - Avanserte forhold...')
        geometriInfo = []
        for i in xrange(len(self.shapely)):
            for j in xrange(i + 1, len(self.shapely)):
                intersect = 'Skjaering mellom:', str(self.shapely[i]), 'og', str(self.shapely[j]), \
                            self.shapely[i].intersects(self.shapely[j])
                koordIntersect = 'Koordinater for eventuell skjaering:', str(
                    self.shapely[i].intersection(self.shapely[j]))
                tomteGrenser = 'Deler av geometrien ligger langs med hverandre (f.eks tomtegrense):', \
                               self.shapely[i].touches(self.shapely[j])
                inni = 'Deler av geometri er inni hverandre:', self.shapely[i].within(self.shapely[j])
                relasjoner = 'DE-9IM Relasjoner mellom geometri:', self.shapely[i].relate(self.shapely[j])

                geometriInfo.append(str(intersect) + '\n' + str(koordIntersect) + '\n' + str(tomteGrenser) + '\n' + str(
                    inni) + '\n' + str(relasjoner) + '\n')

        self.progress.setText('Kontroll ferdig...')

        return '\n'.join(geometriInfo)
```

```
    def skjemakontroll(self):
        """
        Vanlig XML-skjemakontroll
        :return: At skjemaet er gyldig om det er det. Hvis ikke så returnerers feilene som er oppdaget
        """
        self.progress.setText('Kontrollerer mot skjema...')
        # Validering
        feil = []
        if self.xmlSkjema.validate(self.gml):
            return 'Valideringsresultat:', self.xmlSkjema.validate(self.gml)
        else:
            # Rapporterer feil hvis valideringen = False
            for i in xrange(len(self.xmlSkjema.error_log)):
                error = self.xmlSkjema.error_log[i]
                error_str = unicode(error).encode("utf-8")
                feil.append(error_str + '\n')

        self.progress.setText('Kontroll ferdig...')

        return '\n'.join(feil)



app = QtGui.QApplication(sys.argv)
myWindow = MyWindowClass(None)
myWindow.show()
app.exec_()
```

Vedlegg H – UI (gmlkontroll.ui)

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>gmlkontroll</class>
 <widget class="QMainWindow" name="gmlkontroll">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>881</width>
    <height>520</height>
   </rect>
  </property>
  <property name="windowTitle">
   <string>gmlkontroll</string>
  </property>
  <widget class="QWidget" name="centralWidget">
   <layout class="QGridLayout" name="gridLayout">
    <item row="0" column="0" colspan="4">
     <widget class="QTextEdit" name="textEdit">
      <property name="readOnly">
       <bool>true</bool>
      </property>
      <property name="html">
       <string>&lt;!DOCTYPE HTML PUBLIC &quot;-//W3C//DTD HTML 4.0//EN&quot;
&quot;http://www.w3.org/TR/REC-html40/strict.dtd&quot;&gt;

&lt;html&gt;&lt;head&gt;&lt;meta name=&quot;qrichtext&quot; content=&quot;1&quot;
/&gt;&lt;style type=&quot;text/css&quot;&gt;

p, li { white-space: pre-wrap; }

&lt;/style&gt;&lt;/head&gt;&lt;body style=&quot; font-family:'MS Shell Dlg 2'; font-
size:8.25pt; font-weight:400; font-style:normal;&quot;&gt;

&lt;p style=&quot; margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -
qt-block-indent:0; text-indent:0px;&quot;&gt;&lt;span style=&quot; font-
size:14pt;&quot;&gt;Kontroll av GML - prototype&lt;/span&gt;&lt;/p&gt;
```

<p style=" margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;">Funksjonalitet:</p>

<p style=" margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;">- Innlesing av GML-dokument, skjema og applikasjonskjema</p>

<p style=" margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;">- Skjemakontroll</p>

<p style=" margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;">- Geometrikontroll - Enkle forhold (ser på geometri hver for seg f.eks. areal)</p>

<p style=" margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;">- Geometrikontroll - Antall (teller geometri-elementer og relasjoner f.eks skjæring)</p>

<p style=" margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;">- Geometrikontroll - Avanserte forhold (ser på forhold mellom geometri f.eks. skjæring)</p></body></html></string>

      </property>

     </widget>

    </item>

    <item row="1" column="0">

     <widget class="QLineEdit" name="fileBrowser"/>

    </item>

    <item row="1" column="2">

     <widget class="QComboBox" name="droppMeny">

      <item>

       <property name="text">

        <string>Skjemakontroll</string>

       </property>

      </item>

      <item>

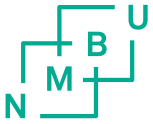
       <property name="text">

        <string>Geometrikontroll - Enkle forhold</string>

```
   </property>
  </item>
  <item>
   <property name="text">
    <string>Geometrikontroll - Antall</string>
   </property>
  </item>
  <item>
   <property name="text">
    <string>Geometrikontroll - Avanserte forhold</string>
   </property>
  </item>
 </widget>
</item>
<item row="1" column="3">
 <widget class="QPushButton" name="valgKnapp">
  <property name="text">
   <string>Kjør kontroll</string>
  </property>
 </widget>
</item>
<item row="1" column="1">
 <widget class="QPushButton" name="opplastKnapp">
  <property name="text">
   <string>GML-fil</string>
  </property>
 </widget>
</item>
<item row="2" column="0">
 <widget class="QLineEdit" name="fileBrowser2"/>
</item>
<item row="2" column="1">
```

```
   <widget class="QPushButton" name="opplastKnapp2">
    <property name="text">
     <string>Skjemaer</string>
    </property>
   </widget>
  </item>
  <item row="4" column="0" colspan="4">
   <widget class="QTextEdit" name="utskriftBoks">
    <property name="readOnly">
     <bool>true</bool>
    </property>
   </widget>
  </item>
  <item row="2" column="2">
   <widget class="QLineEdit" name="progress">
    <property name="readOnly">
     <bool>true</bool>
    </property>
   </widget>
  </item>
 </layout>
</widget>
<widget class="QMenuBar" name="menuBar">
 <property name="geometry">
  <rect>
   <x>0</x>
   <y>0</y>
   <width>881</width>
   <height>21</height>
  </rect>
 </property>
</widget>
```

```
 <widget class="QToolBar" name="mainToolBar">
  <attribute name="toolBarArea">
   <enum>TopToolBarArea</enum>
  </attribute>
  <attribute name="toolBarBreak">
   <bool>false</bool>
  </attribute>
 </widget>
 <widget class="QStatusBar" name="statusBar"/>
</widget>
<layoutdefault spacing="6" margin="11"/>
<resources/>
<connections/>
</ui>
```



Norges miljø- og biovitenskapelig universitet
Noregs miljø- og biovitskapelege universitet
Norwegian University of Life Sciences

Postboks 5003
NO-1432 Ås
Norway