- Standarder geografisk informasjon

SOSI Generell del
Regler for UML-modellering

Versjon 5.1 - februar 2020

Kartverket

INNHOLDSFORTEGNELSE

FIGURLISTE

1 Orientering og introduksjon

SOSI står for "Samordnet Opplegg for Stedfestet Informasjon" og utgjør felles regelsett i form av standarder samt en objektkatalog over fagområder.

Denne standarden, Regler for UML-modellering, beskriver modellering av data og tjenester fra "use case" og forretningsmodell til tjenester med tilhørende informasjonsmodeller, og er en av standardene som inngår i SOSI.

Siden forrige hovedversjon av SOSI (versjon 4) ble utgitt i 2006/2007 har det skjedd en betydelig endring i bruken av modeller. Fra et utgangspunkt i diagrammer i UML-modellene som ble satt inn som figurer i standardene, er modellene nå et utgangspunkt for generering av en rekke ulike representasjoner (GML, SOSI-syntaks, XML), dokumentasjon (grafiske og tekstlig forklaring til modellene, ulike definisjonsspråk (WSDL for tjenester, BPEL for forretningsmodeller), etc. Og dette er bare starten; semantisk web (OWL og RDF), nye formater (JSON, GeoJSON, TopoJSON, JSON-LD), samt funksjoner som språkuavhengighet i modellene, er i ferd med å realiseres. Og dette er ikke noe som vi i Norge er alene om, denne utfordringen deler vi med andre land, og det gjør vi også med tanke på løsningene (f.eks. internasjonal standardisering).

Denne utviklingen krever større kvalitet i modellene og et mer omfattende regelsett for hvordan disse modellene skal utvikles for at det skal være mulig å automatisk generere dette mangfoldet.

Denne versjonen følger opp revisjoner av internasjonale standarder (som også er norske standarder), med fokus på ISO 19103 Conceptual schema Language som gir generelle regler for modellering med UML, ISO 19109 Rules for Application Schema som gir klare regler for modellering av et UML-applikasjonsskjema samt ISO 19119 Services som gir regler for modellering av tjenester. Samtidig er det også forsøkt å ta hensyn til tidligere versjoner av SOSI for å være mest mulig bakoverkompatibel. Men denne versjonen tar i langt større grad utgangspunkt i arbeidet med internasjonale standarder og hvordan disse er implementert i f.eks. de dokumenter som Geodataloven refererer til.

Standarden inneholder en rekke krav og anbefalinger. Navnene på disse kravene er i mange tilfeller identer som er hentet fra internasjonale standarder, og med engelske navn.

Kapittel 8	Bare norske krav og anbefalinger
Kapittel 9	Krav og anbefalinger fra ISO 19119 Services, angitt som "rec" for recommendation og "req" for requirement.
Kapittel 10	Krav og anbefalinger fra ISO 19103, Conceptual schema language, angitt med nummere.
Kapittel 11	Krav og anbefalinger fra ISO 19109 Rules for application schema, angitt som "rec" for recommendation og "req" for requirement.

I tillegg til de krav og anbefalinger som er videreført fra de internasjonale standardene finnes det også en rekke krav og anbefalinger som ikke har sitt utspring i internasjonale standarder, og som er en ytterligere spesifisering basert på erfaring i Norge. Disse har norske navn.

Denne standarden gir ingen full beskrivelse av det som er spesifisert i de internasjonale standardene som ligger til grunn, og vil ikke være noen erstatning for disse. Flere av de kravene som er beskrevet her refererer til krav med tilhørende konformitetstester i de internasjonale standardene. Disse har engelske navn for identifikasjon, og en må ha tilgang til de internasjonale standarden for nærmere informasjon. Men ambisjonsnivået er at for "vanlige" brukere som ønsker å modellere i UML skal denne standarden være tilstrekkelig. Derimot, de som skal implementere standardene i egne systemer og/eller er ansvarlige for kvalitetssikringen av UML-modeller må også har kjennskap til de internasjonale standardene. Dette gjelder også de deler hvor vi har liten erfaring så langt og hvor det er referanse til de respektive internasjonale standardene for nærmere beskrivelser.

SOSI del 1 Regler for UML-modellering beskriver regler for modellering på et konseptuelt nivå, dvs. uavhengig av hvilke plattform eller system som modellene skal implementeres i. Tilpasninger

av f.eks. en konseptuell datamodell til en proprietær datamodell i en forvaltningsløsning er ikke en del av SOSI standarden. Kapittel 13 beskriver tagged values som benyttes i forbindelse med generering av plattformspesifikke realiseringer, slik som SOSI-format og GML.

Se også ISO/TC 211 Best practice https://github.com/ISO-TC211/UML-Best-Practices. Her er det nyttig informasjon om god diagramdesign, verktøy og annet referansemateriell.

2 Historikk og endringslogg

Versjon	Dato	Utført av	Grunnlag for endringen
5.0	2016-02-09	Prosjektgruppe SOSI del 1	Dette er et dokument som er basert på flere fysiske møter samt en rekke telefonmøter. Utgangspunktet er Geodataloven og reviderte ISO standarder.
5.1	2020-01-27	SOSI gruppe 1	Bakoverkompatible forbedringer basert på tilbakemeldinger

Aktuell ansvarlig:
Statens kartverk
Standardiseringssekreteriatet
Kartverksvn. 21, 3511 Hønefoss
Tlf 32 11 80 00

Faglig ansvar:
Statens kartverk
IT-avdelingen - Standarder og teknologiutvikling
Kartverksvn. 21, 3511 Hønefoss
Tlf 32 11 80 00

2.1 Endringslogg

Dette er en mindre revisjon av "Regler for UML-modellering versjon 5.0 fra 2016".

Kapittel 10 - Generelt om UML-modellering og kapittel 11 er en revisjon av SOSI del 1 versjon 4.0 Regler for UML-modellering. Disse to kapitlene er basert på de reviderte versjonene ISO 19103: Conceptual schema language og ISO 19109: Rules for application schema. Disse standardene gir mer konkrete krav til UML-modellering.

Denne standarden er basert på UML versjon 2.

Denne versjonen har delt konformitetsklassen Basisregler i to alternative klasser. Den ene konformitetsklassen er bakoverkompatibel med Regler for UML modellering versjon 5.0. det vil si at ingen krav er strammet inn, og modeller som var gyldige etter versjon 5.0 fortsatt er gyldige i versjon 5.1. Den andre konformitetsklassen beskriver nasjonale tilpassinger for modeller som ikke klarer å innfri alle internasjonale krav, men som likevel kan være basis for implementasjoner.

3 Omfang

3.1 Omfatter

Standarden beskriver regler for modellering av brukertilfeller (”use case”), forretningsmodeller, modellering av tjenester samt geografiske objekter i UML, på en hensiktsmessig måte ut fra ulike brukerbehov, i henhold til internasjonale standarder i regi av ISO/TC 211. Standarden beskriver internasjonale konsepter med eksempler fra Norge og norske modeller.

3.2 Målsetting

Denne standarden inneholder regler for hvordan UML-modeller skal utvikles. UML-modeller spesifiseres på ulike modellnivåer (abstraksjonsnivåer), hovedfokuset ligger på et konseptuelt nivå, dvs. uavhengig av hvilke plattform og teknologi som modellen skal implementeres i.

Standarden inneholder imidlertid føringer for implementasjonsspesifikke modeller. En oversikt over modellnivåer er nærmere angitt i kapittel 7.10.

Regler for hvordan et konseptuelt skjema mappes til et implementasjonsspesifikt skjema beskrives i andre SOSI standarder. For UML-applikasjonsskjema er dette beskrevet i

- SOSI del 1 – Realisering i SOSI-format 5.0
- SOSI del 1 – Realisering i GML-format 5.0

Tilsvarende kan det etterhvert komme "mapping"-regler til ulike plattformer for tjenester.

For UML-applikasjonsskjema (informasjonsmodeller) er det lagt stor vekt på at de modelleringsprinsippene som legges til grunn, skal kunne være gjenstand for automatisk generering av ulike typer representasjon eller dokumentasjon, f.eks. utvekslingsformat. Eksempler her er GML applikasjonskjema, for enkle modeller kan det også automatisk genereres SOSI-syntaks spesifikasjon. Fremover vil det være sterkere fokus på semantisk web (RDF og Linked Data), som vil kunne avledes direkte fra de samme UML-applikasjonsskjemaer.

For tjenestemodeller er det lagt vekt på at modellene skal kunne mappes til ulike teknologier, dvs. at en ikke trenger å modellere tjenesten på nytt ved overgang fra f.eks. Java til en Web service.

Tilsvarende er det også lagt større vekt på registre, både forvaltningen av dem samt tilgang til verdier i eksisterende kodelister. Selve modellene skal være tilgjengelig i et eget versjonert modellregister.

3.3 Bruksområde

Standarden skal sikre at modellene som ligger til grunn for data og tjenester i vår nasjonale infrastruktur beskrives på en slik måte at brukerbehovene tilfredsstilles, både med tanke på innhold men også utvekslingsformat.

For lettere å forholde seg til standarden er det lagt inn en oversikt over hvilke deler av standarden som er relevant for de ulike målgrupper:

Tabell 3.1 Målgrupper og relevante kapitler.

Målgruppe	Arbeidsområde/hensikt	Kapitler i denne standarden
Fasilitator	Den som leder et prosjekt som innebærer modellering, f.eks. en SOSI-prosjektgruppe.	7, 8
UML-editorer - applikasjonsskjema	Den som har ansvar for UML-modellering av et applikasjonsskjema	8, 10, 11

Målgruppe	Arbeidsområde/hensikt	Kapitler i denne standarden
UML-editorer - tjenestemodeller	Den som har ansvar for UML-modellering av en tjeneste	9, 10
Domeneeksperter	Fageksperter innenfor ulike fagområder	7, (10, 11)
Systemutviklere	Systemleverandører som skal implementere SOSI	Alle, samt ATS i normativt refererte standarder.
Systemarkitekter	Den som har ansvar for forretningsarkitektur	7, 8
Allmennheten	Vanlige brukere av SOSI, som vil gjøre direkte søk mot et modellregister via et web innsyn, slik som for eksempel: https://objektkatalog.geonorge.no/	7, 10.1

4 Konformitetsklasser

Denne standarden dekker ulike typer modellering. Kravene til ulike typer modellering vil fremkomme i ulike konformitetsklasser. De modeller/komponenter som hevder at de skal være konforme med denne standarden må være konform med en eller flere av konformitetsklassene. Der det er ønskelig kan det angis hvilke subsett av konformitetsklasser og krav som skal følges eller er fulgt.

Denne standarden inneholder 12 konformitetsklasser, fordelt på følgende:

- Basisregler
  - Basisregler for UML med realisering av internasjonale standarder
  - Basisregler for UML med nasjonale tilpassinger (innstramminger/utelatelser)
- UML-applikasjonsskjema
  - UML-applikasjonsskjema
  - Enkle geometriske primitiver
  - Andre geometriske primitiver
  - Geometriske komplekser
  - Geometriske aggregater
  - Topologiske primitiver
  - Topologiske komplekser
- Tjenester
- TjenestemodelleringRegistre
  - SOSI-modellregister
  - Kodelister

Denne versjonen av standarden har ingen krav knyttet til brukstilfeller og forretningsprosesser, og har følgelig ingen konformitetsklasser for dette.

Nærmere beskrivelse av konformitetsklassene med tilhørende konformitetstester er beskrevet i Vedlegg A.

5 Normative referanser

EIF	European Interoperability Framework http://ec.europa.eu/idabc/en/document/2319/5644.html
EIRA	European Interoperability Reference Architecture https://joinup.ec.europa.eu/asset/eia/description
IETF RFC 5646	Språkkoding
INSPIRE	INSPIRE D2.5_v3 http://inspire.jrc.ec.europa.eu/documents/Data_Specifications/D2.5_v3.4.pdf
ISO 639-1:2002	Codes for the representation of names of languages -- Part 1: Alpha-2 code
ISO 639-3:2007	Codes for the representation of names of languages -- Part 3: Alpha-3 code for comprehensive coverage of languages
ISO 8601:2004	Data elements and interchange formats — Information interchange — Representation of dates and times https://www.standard.no/no/Nettbutikk/produktkatalogen/Produktpresentasjon/?ProductID=124527
ISO 19103:2015	Conceptual schema language (CSL) http://www.standard.no/no/Nettbutikk/produktkatalogen/Produktpresentasjon/?ProductID=791977
NS-EN ISO 19107:2005	Modell for å beskrive geometri og topologi (ISO 19107:2003) http://www.standard.no/no/Nettbutikk/produktkatalogen/Produktpresentasjon/?ProductID=144442
NS-EN ISO 19108:	Modell for å beskrive tidsaspekter (ISO 19108:2002) http://www.standard.no/no/Nettbutikk/produktkatalogen/Produktpresentasjon/?ProductID=144443
ISO/FDIS 19109:2015	Geographic information- Rules for application schema
NS-EN ISO 19115-1:2014	Metadata - Del 1: Grunnprinsipper (ISO 19115-1:2014) http://www.standard.no/no/Nettbutikk/produktkatalogen/Produktpresentasjon/?ProductID=702321
ISO 19119: 2015	Geographic information – Services http://www.standard.no/no/Nettbutikk/produktkatalogen/Produktpresentasjon/?ProductID=210119
ISO 19123:2007	Modell for overdekkende tematisk representasjon (ISO 19123:2005) http://www.standard.no/no/Nettbutikk/produktkatalogen/Produktpresentasjon/?ProductID=269727 Ytterligere informasjon finnes også i INSPIRE D2.5_v3.4 kapittel 10.5.
NS-ISO 19136:2009 (GML 3.2.1)	Geografisk markeringsspråk (GML) (ISO 19136:2007) Annex-E. http://portal.opengeospatial.org/files/?artifact_id=26765
ISO 19136-2:2015 (GML 3.3)	Geography Markup Language (GML) -- Part 2: Extended schemas and encoding rules https://portal.opengeospatial.org/files/?artifact_id=46568

NS-EN ISO 19156:2013	Observasjoner og målinger (ISO 19156:2011) http://www.standard.no/no/Nettbutikk/produktkatalogen/Produktpresentasjon/?ProductID=657670
NS-EN ISO 19157: 2013	Datakvalitet (ISO 19157:2013) http://www.standard.no/no/Nettbutikk/produktkatalogen/Produktpresentasjon/?ProductID=682968
NS-EN ISO 19135-1: 20xx	Registerfunksjonalitet (ISO 19135-1:20xx) http://www.standard.no/no/Nettbutikk/produktkatalogen/Produktpresentasjon/?ProductID=xxx
ISO/IEC 19505-2:2012	Information technology — Object Management Group Unified Modeling Language (OMG UML) — Part 2: Superstructure NOTE Unified Modeling Language (UML), *version 2.4.1*, http://www.omg.org/spec/UML/2.4.1/
OCL 2.3.1	OMG *Object Constraint Language, version 2.3.1*,available at http://www.omg.org/spec/OCL/2.3.1
RM-ODP	ISO/IEC 10746-1:1998, *Information technology — Open Distributed Processing — Reference model: Overview* *ISO/IEC 10746-2:2009, Information technology — Open Distributed Processing — Reference model: Foundations*
SOSI Nettverk og lineære referanser	SOSI generell del - Nettverk og lineære referanser, versjon 5.0, februar 2016
TOGAF	The Open Group Architecture Framework https://www.opengroup.org/togaf/

6 Termer, definisjoner, forkortelser og notasjon

6.1 Termer og definisjoner

I dette kapitlet er de fleste ord og termer definert, noen både på norsk og engelsk. Første linje inneholder termen på norsk, i uthevet skrift. I noen tilfeller er det også et synonym, dvs. at det er flere termer for det samme. Deretter følger norsk definisjon, neste linje er den engelske termen med engelsk definisjon i kursiv.

Hensikten med de engelske termene er å lettere kunne relatere begrepene i dette dokumentet til internasjonale standarder/dokumenter.

Denne standarden beskriver modelleringsregler for UML. Termer knyttet til modelleringen (klasse, arv, assosiasjon, subtype, etc.) er forklart i standarden, og inngår ikke i dette kapittel.

applikasjonsskjema
konseptuelt skjema for data som skal brukes i en eller flere applikasjoner

application schema
conceptual schema for data required by one or more applications [ISO 19101]

brukstilfeller
beskrivelse av (del-)funksjonaliteten i et system sett fra et eksternt perspektiv.
Merknad: I stedet for å beskrive detaljerte egenskaper med systemet, er hensikten å vise funksjonalitet på et overordnet nivå

mapping
beskrivelse av overgang mellom et konsept på en plattform til et tilsvarende konsept på en annen plattform.

Merknad: Beskrives ofte i form av regler, til nytte for de som skal forstå samt programmere disse overgangene.
Eksempel: Skjematransformasjon

metadata
informasjon som beskriver et datasett

Merknad: Hvilke opplysninger som inngår i metadataene, kan variere avhengig av datasettets karakter. Vanlige opplysninger er innhold, kvalitet, tilstand, struktur, format, produsent og vedlikeholdsansvar.

objekt
forekomst (instans) av en objekttype

feature Instance
abstraction of real world phenomena

NOTE A feature may occur as a type or an instance. Feature type or feature instance should be used when only one is meant.

objektkatalog
definisjon og beskrivelse av objekttyper, objektegenskaper samt relasjoner mellom objekter, sammen med eventuelle funksjoner som er anvendt for objektet
Eksempel: SOSI

feature Catalogue
catalogue containing definitions and descriptions of the feature types, feature attributes and feature relationships occurring in one or more sets of geographic data, together with any feature operations that may be applied

objekttype
en klasse av objekter med felles egenskaper, forhold mot andre objekttyper og funksjoner

geografisk objekttype
klasse med stereotype «FeatureType» som er en abstraksjon av et fenomen i den virkelige verden

Merknad: Instanser av slike klasser skal ha sin egen identitet og kan ha egenskaper med geometritype eller andre geografiske tilknyttinger, for eksempel topologiske forhold eller lineær referanse.

feature type
abstraction of real world phenomena [ISO 19101]

plattformuavhengige modeller
spesifikasjoner for systemets funksjonalitet og data som er plattformuavhengig.

Merknad: I henhold til Object Management Group (OMG), benevnes disse PIM – Platform Independent Models

plattformspesifikke modeller
spesifikasjoner for systemets funksjonalitet og data som er tilpasset en spesifikasjonsplattform (f.eks XML), fortrinnsvis automatisk generert fra en plattformuavhengig modell

Merknad: I henhold til Object Management Group (OMG), benevnes disse PSM – Platform specific Models

subversion
versjonskontrollsystem som gjør det mulig å lagre versjoner av datafiler på en sentral server, og samtidig organisere endringer i datafilene uten at det blir konflikt mellom brukere som eventuelt endrer samtidig.

tagged value (engelsk)
en navnet verdi som knyttes til et modellelement, disse brukes ofte til å muliggjøre automatisk mapping til ulike platformer

tjeneste
funksjonalitet gitt av en enhet gjennom et grensesnitt

tjenesteskjema
(tjenestemodell)

6.2 Forkortelser

ATS	Abstrakt TestSuite
BPMN	Business Process Model and Notation
CSL	Conceptual schema language
DCP	Distributed Computing Plattform
DOK	Det offentlige kartgrunnlaget
EA	Enterprise Architect – verktøy for UML-modellering
EIF	European Interoperability Framework
EIRA	European Interoperability Reference Architecture
ELF	European Location Framework (EU prosjekt)
FE	Filter Encoding
GeoJSON	JSON med rudimentær geometri

GFM	General Feature Model
GIS	Geografisk InformasjonsSystem
GML	Geography Markup Language, XML Encoding Specification for geo-related data
IETF	Internet Engineering Task Force
IFC	Industry Foundation Classes
INSPIRE	Infrastructure for Spatial Information in the European Community
ISO	International Standardization Organization
JRC	Joint Research Centre
JSON	JavaScript Object Notation
JSON-LD	JavaScript Object Notation - Linked Data
MDA	Model Driven Architecture (OMG)
OCL	Object constraint language
OGC	Open Geospatial Consortium
OMG	Object Management Group
OWL	Web Ontology Language
PIM	Platform Independent Model
PSM	Platform Specific Model
RDF	Resource Description Framework
REST	Representational State Transfer
RFC	Request for Comments
RM-ODP	Reference Model of Open Distributed Processing
RPC	Remote Procedure Call
SKOS	Simple Knowledge Organization System
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SOS	Sensor observation service
SOSI	Samordnet Opplegg for Stedfestet Informasjon
SVN	Subversion
TIN	Triangulated Irregular Network
TOGAF ADM	The Open Group Architecture Framework Architecture Development Model
TopoJSON	GeoJSON med topologi
UC	Use case – brukstilfelle
UML	Unified modeling language
UoD	Universe of Discourse
WCS	Web Coverage Service
WFS	Web Feature Service
WMS	Web Map Server
WSDL	Web Service Description/Definition Language
XMI	XML Metadata Interchange
XML	Extensible Markup Language
xsd	XML Schema Definition

7 SOSI-metoden

7.1 Introduksjon

SOSI-metoden er en samling av metodebeskrivelser, standarder, fellesressurser (f.eks. SOSI-modellregister) og verktøy (f.eks. SOSI-kontroll) som utgjør teknologi-delen av den norske infrastrukturen for geografisk informasjon. SOSI-metoden fastslår hvordan generelle metoder og teknologier skal brukes i den norske infrastrukturen for geografisk informasjon.

SOSI-standarden er en samling norske standarder og spesifikasjoner for geografisk informasjon og hvordan geografisk informasjon skal utveksles mellom ulike aktører. Spesifikasjonene og standardene for geografisk informasjon omfatter definisjon av:

- objekter og fenomener fra den virkelige verden som er direkte eller indirekte stedfestet
- objektenes egenskaper
- relasjoner mellom objekter.

Spesifikasjoner og standarder for utveksling av geografisk informasjon omfatter beskrivelser av:

- dataformater
- tjenester.

SOSI-standarden er direkte forankret i internasjonale standarder tilpasset norske forhold, norske lover og beste praksis i Norge.

SOSI-standarden inneholder også regler og retningslinjer for hvordan SOSI-standarder og ulike spesifikasjoner skal utvikles.

SOSI-metoden er en samling oppskrifter som skal brukes for å utvikle norske standarder og spesifikasjoner for geografisk informasjon, herunder standardiserte:

- datamodeller
- tjenestemodeller
- formater for utveksling av geografiske data.

Oppskriftene inneholder krav, anbefalinger og metoder for hvordan standarder og spesifikasjoner skal utvikles, dokumenteres og implementeres.

Innholdsmessig kan en dele SOSI-metoden i to ulike sett med oppskrifter:

- datamodellering
- tjenestemodellering.

Tjenestemodellering benytter også oppskrifter og regler som er definert under datamodellering.

7.2 Bakgrunn om IT rammeverk og modellering

7.2.1 IT rammeverk

RM-ODP - **Reference Model of Open Distributed Processing** - er viktig rammeverk for standardisering av datautveksling mellom distribuerte systemer ved at den definerer flere sentrale konsepter for standardiseringsarbeid, f.eks. objekt orientering, utvikle spesifikasjoner fra ulike synsvinkler (viewpoints) og bruk av formelle beskrivelsesmetoder.

TOGAF ADM - The Open Group Architecture Framework – Architecture Development Method er prosessen som brukes for å analysere en hel organisasjon eller et spesifikt fagområde for å komme fram de spesifikke forretnings- og virksomhetsbehov.

EIRA - European Interoperability Reference Architecture er en referansearkitektur for leveranser av digitale offentlige tjenester mellom sektorer. Klassifikasjon og organisering av byggeklosser brukt i utviklingen og leveranse av digitale tjenester innen offentlig administrasjon, knyttet til EIF (European Interoperability Framework).

[EIRA beskriver 4 grupper med byggeklosser (juridisk, organisatorisk, semantisk og teknisk. Denne SOSI-standarden forholder seg til de semantiske og teknologiske byggeklossene]

EIF - European Interoperability Framework: Europeisk interoperabilitetsrammeverk v2.0

7.2.2 Konsepter

7.2.2.1 Modelldrevet arkitektur

Objektorientering er en tilnærming hvor en beskriver informasjonen i form av objekter. Objektene kan inneholde både data og funksjonalitet. Objekter tilbyr informasjon i form av tjenester.

Modelldrevet Arkitektur (MDA) - som er utarbeidet av Open Management Group (OMG) - er en metode innen systemutvikling som:

1. Skiller spesifikasjonene for funksjonalitet og data, fra spesifikasjonene for implementasjon av funksjonene og database.
2. Spesifikasjonene beskrives som modeller. Spesifikasjonene for systemets funksjonalitet og data som er plattform-uavhengig, benevnes som PIM – Platform Independent Models. Her brukes det norske ordet plattformuavhengige modeller.
3. MDA legger opp til, i mest mulig grad, automatisk generering av plattformspesifikke spesifikasjoner og om mulig også kode for implementasjon. Disse spesifikasjonen benevnes som PSM -Platform Specific Models. Her brukes det norske ordet plattformspesifikke modeller.

Den modelldrevne tilnærmingen følger konseptene utviklet i den modelldrevne arkitektur (MDA-Model Driven Architecture) definert av OMG, og beskriver en åpen, leverandørnøytral tilnærmelse til endringer i teknologi og forretningsprosesser.

En applikasjons plattformuavhengig modell modellert i UML og/eller andre OMG-modelleringsstandarder kan realiseres gjennom MDA på enhver plattform, åpen eller proprietær, inkludert WebServices, .NET, CORBA, J2EE, etc. Disse plattformuavhengige modellene dokumenterer funksjonalitet og oppførsel separat fra den teknologispesifikke kodingen av en implementasjon basert på en spesiell teknologi.

Levetiden til en teknisk implementering er kortere enn levetiden til informasjonen den behandler. Dette gjør det nødvendig å beskrive informasjonen på en slik måte at den tillater nye teknikker og implementasjonsmiljøer.

Denne arkitektoniske linjen fokuserer på tre hovedspørsmål. Disse er portabilitet, interoperabilitet og gjenbruk. Sentralt i MDA står også XML Metadata Interchange (XMI) for utveksling av UML-modeller mellom ulike plattformer og verktøyer. Primære mål er gjenbrukbarhet, flyttbarhet og samhandling for objektbasert programvare i et distribuert, heterogent miljø. Tilpassing til disse spesifikasjonene vil gjøre det mulig å utvikle heterogene applikasjonsmiljøer på tvers av alle store maskinvareplattformer og operasjonssystemer.

Automatisk mapping – automatisk generering av plattformspesifikke skjema (PSM) fra de konseptuelle modellene (PIM).

Mapping regler er ulike typer regler for generering av ulike dokumenter fra de respektive modellene. Eksempler: XML skjema, dokumentasjon, RDF, Excel, etc.

Service Oriented Architecture (SOA) er en teknikk for design av enkle og komplekse datasystemer. Systemet bygges opp av flere uavhengige komponenter. Hver komponent tilbyr tjenester som de andre komponentene kan benytte uten å måtte implementere funksjoner og database selv.

Det er mange fordeler med systemer som følger SOA prinsippene sammenlignet med komplekse og sterkt integrerte systemer:

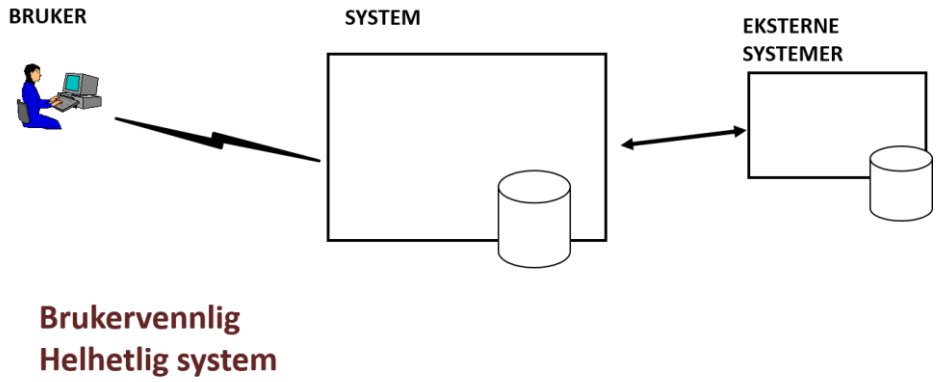
- raskere å utvikle
- lettere å bygge spesialiserte delsystemer (f.eks. GIS, Digitalt dokumentarkiv, Sakssystem)
- ett delsystem kan erstattes av et nytt uten å endre de andre delsystemene
- ett delsystem kan være del av flere systemer.

Av viktige prinsipper nevnes:

1. brukerperspektivet
2. den modulære systemarkitekturen
3. interoperabilitet/kommunikasjon.

7.2.2.2 Brukerperspektivet

Et eksempel på brukerperspektivet er vist i Figur 7.1 Brukerperspektivet.



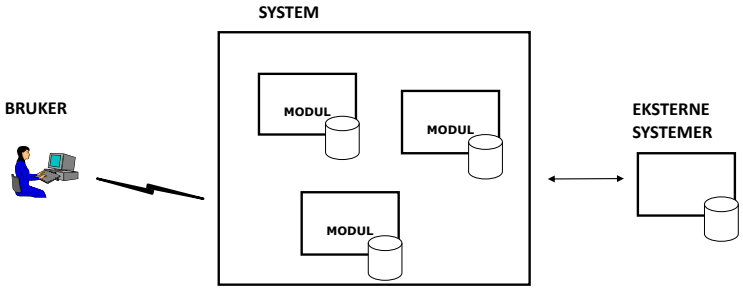
Figur 7.1 Brukerperspektivet

Brukeren skal som i et hvert system oppleve systemet som:

- brukervennlig
- enhetlig
- intuitivt brukergrensesnitt (f.eks. web basert)
- online aksess til databasen
- online aksess til ekstern informasjon.

7.2.2.3 Modulær systemarkitektur

Et eksempel på modulær systemarkitektur er vist i Figur 7.2 Modulær systemarkitektur.

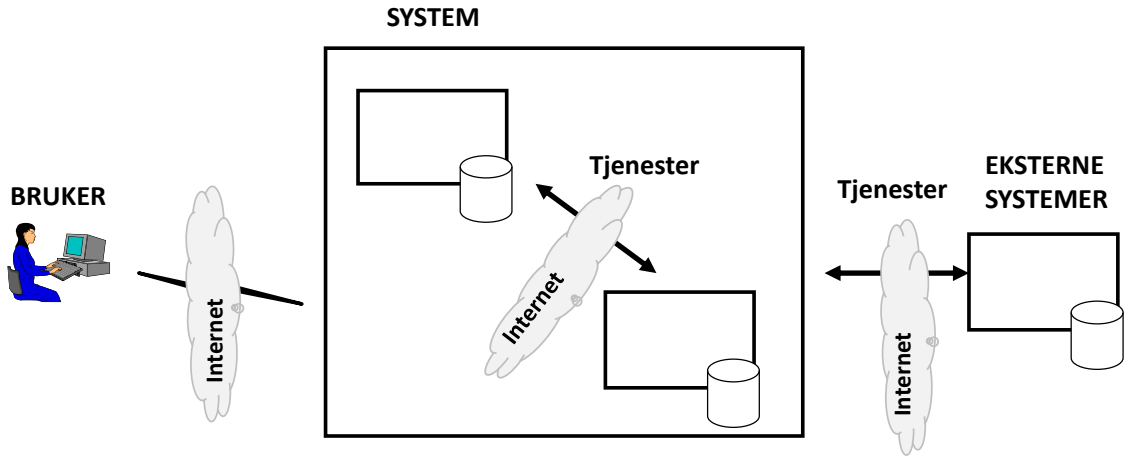


Figur 7.2 Modulær systemarkitektur

System bygges opp med uavhengige og essensielle moduler (delsystemer). Hver modul er eksklusiv ansvarlig for en definert del av data i systemet. En modul har ikke tilgang til database i andre delsystemer.

7.2.2.4 **Interoperabilitet/kommunikasjon**

Et eksempel på kommunikasjon gjennom tjenester er vist i Figur 7.3 Kommunikasjon gjennom WEB-tjenester som bruker http protokoll.



Figur 7.3 Kommunikasjon gjennom WEB-tjenester som bruker http protokoll

All kommunikasjon mellom systemene skjer ved bruk av standardiserte protokoller (f.eks. http/https, internett). All utveksling av data mellom modulene skjer ved tjenester.

7.2.3 Perspektiver (Vinklinger/viewpoints)

ISO19xxx-standardene bygger på konsepter som er definert i RM-ODP, som også er en ISO-standard. Se Tabell 7.1 Perspektiver i RM-ODP. Grunnleggende i RM-ODPs rammeverk er å skille mellom:

- konseptuelle spesifikasjoner for et systems funksjoner og data
- spesifikasjoner for implementasjon av funksjoner og databaser.

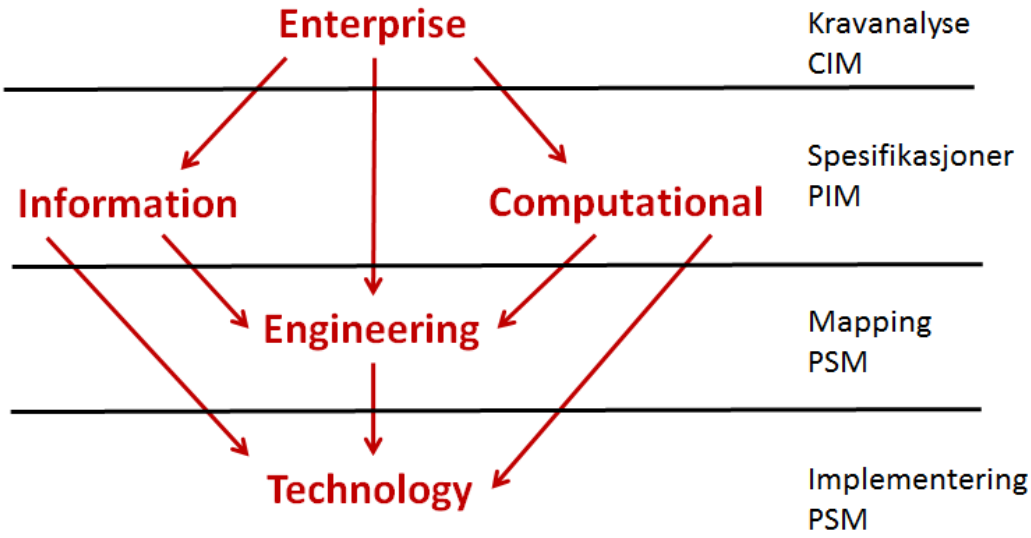
Ofte er selv spesifikasjonene på konseptuelt nivå svært komplekse og vanskelig å ha oversikt over. Ofte har forskjellige personer spesielt interesse for deler av spesifikasjonene. RM-ODP har følgelig introdusert bruk av perspektiver (viewpoints) ved utarbeidelse av system-spesifikasjoner. Konseptet med perspektiver går ut på å bryte ned komplekse spesifikasjoner i et sett av separate deler, som likevel henger sammen.

Tabell 7.1 Perspektiver i RM-ODP

RM-ODP-perspektiv (viewpoint)	**Forklaring**
Enterprise (Virksomhet)	Fokus på hensikt, omfang og policy for et informasjonssystem innenfor en virksomhet
Information (Data)	Fokus på hvilken informasjon som skal håndteres i systemet, informasjonens semantikk, definisjoner, begrensinger og prosessering

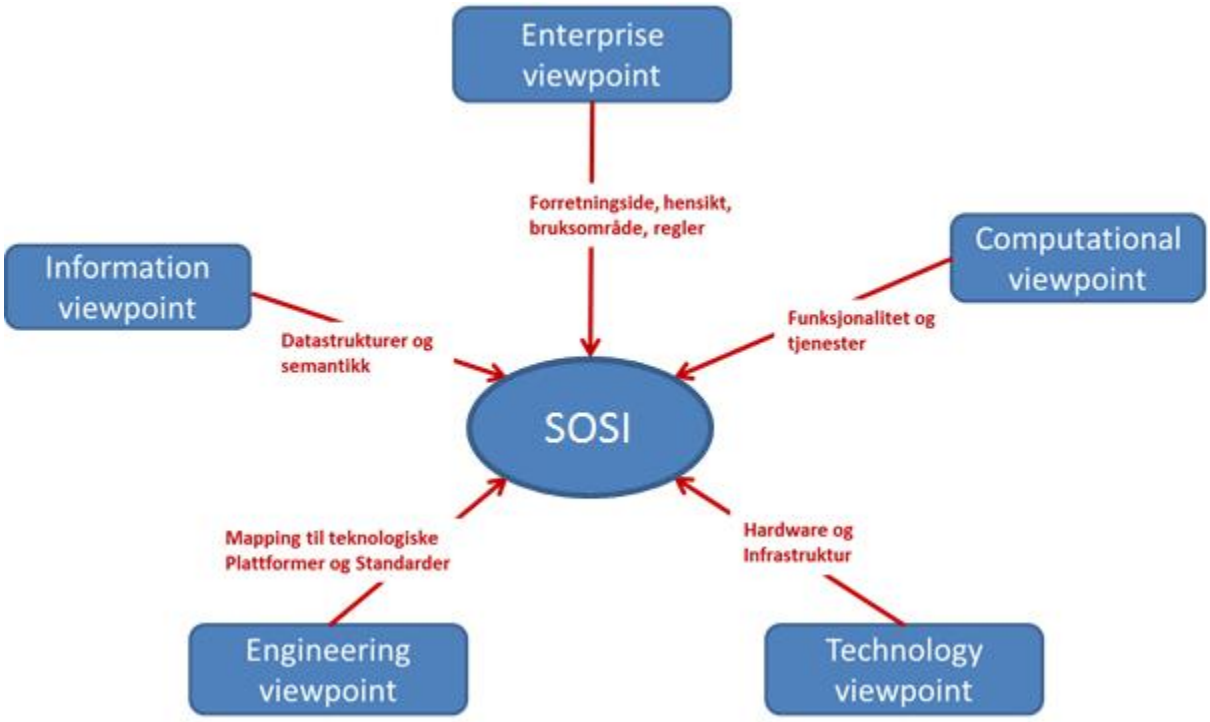
Computational/Services (Funksjoner/Tjenester)	Fokus på funksjonalitet, distribusjon og utveksling av informasjon mellom systemets komponenter, via definerte grensesnitt
Engineering (Mapping)	Fokus på mekanismer og standarder for distribusjon av informasjon, og mapping fra konseptuelle spesifikasjoner
Technology (Implementasjon)	Fokus på implementasjon av tjenester og distribusjon av data basert på tilgjengelig hardware og software

Figur 7.4 viser forholdet mellom de fem perspektiver som RM-ODP har definert. Det første nivået er å analyse en virksomhets forretningsprosesser. Deretter vil en spesifisere de nødvendige data og tjenester på et plattformuavhengig nivå. Gjennom "engineering"- og "technology"-vinklingen fokuseres det på mappingen ned til valgt teknologi for implementasjon.



Figur 7.4 Forholdet mellom de fem perspektivene som RM-ODP har definert

Figur 7.5 viser de fem perspektiver som RM-ODP har definert. (Mer om RM-ODPs "viewpoints" i ISO 19119 Services) eller ISO/IEC 10746-1:1998, Information technology — Open Distributed Processing — Reference model: Overview.
.



Figur 7.5 Forholdet mellom SOSI og RM-ODP vinklinger

SOSI benytter RM-ODPs tenkemåte med perspektiver både i analysesammenheng og ved utarbeidelsen av standardene.

Tabell Tabell 7.2 Oversikt over kapitler som handler om RM-ODPs perspektiver.viser de ulike perspektiver samt hvilke kapittel i standarden disse omhandles i, evt. referanse til andre SOSI-standarder.

Tabell 7.2 Oversikt over kapitler som handler om RM-ODPs perspektiver.

RM-ODP-perspektiv (viewpoint)	**Aktivitet**	**Kapittel i standarden SOSI del 1 - Regler for UML-modellering, evt. andre.**
Enterprise (Virksomhet)	Innhenter fagkunnskap om innhold og bruk av data. Analyserer bruk av data og behov for tjenester	Kapittel 8
Information (Data)	Etablerer konseptuelle fagmodeller og produktspesifikasjoner	Kapittel 10 og 11
Computational/Services (Funksjoner/Tjenester)	Etablerer konseptuelle tjenestemodeller	Kapittel 10 og 9
Engineering (Mapping)	Definerer ”mapping” fra konseptuelle modeller til aktuelle plattformer for datautveksling, samt ”mapping” fra konseptuelle tjenestemodeller til	Kapittel 13 (registere over tagged values) for ”mapping”). SOSI del1 - Realisering i SOSI.

	plattformspesifikke web tjenester	SOSI del 1- Realisering i GML.
Technology (Implementasjon)	Teknologiplattformer/utvekslings formater	Kapittel 12 [Denne standarden beskriver i liten grad implementasjons-teknologien med unntak av SVN for SOSI-modellregister]

7.3 Forankring

SOSI er forankret i norske lover, internasjonale standarder, samt internasjonalt anerkjent rammeverk og metodikk for analyser og utvikling av informasjonssystemer.

7.3.1 Lover

Geodataloven (2010) er den viktigste loven som styrer oppbyggingen av den geografiske infrastrukturen i Norge. Loven skal sikre alle god tilgang til offentlig geografisk informasjon.

Geodataforskriften (2012) gir detaljerte regler om hvordan offentlige etater skal organisere sine internettjenester for å kunne gi effektiv tilgang til geografisk informasjon.

INSPIRE-direktivet (2007) er et europeisk samarbeid for en felles geografisk infrastruktur som Norge har sluttet seg til. Direktivet har som mål å kunne utveksle standardiserte geografiske data mellom deltakerlandene. Direktivet er implementert i norsk lov gjennom Geodataloven og Geodataforskriften.

7.3.2 Standarder

ISO 191xx – geografisk informasjon - er en serie med internasjonale standarder som definerer, beskriver og håndterer geografisk informasjon. Normative referanser for denne standarden er:

- ISO 19103 Generelle regler for modellering med UML
- ISO 19109 Regler for å lage UML-applikasjonsskjema
- ISO 19119 Regler for å lage tjenestemodeller
- ISO 19107 Beskrivelse av geometri og topologi
- ISO 19108 Beskrivelse av mer komplekse temporale aspekter
- ISO 19115-1 Metadata
- ISO 19136 Geografisk markeringsspråk - GML (XML)
- ISO 19136-2 Utvidelser av GML med tanke på raster (Coverage)
- ISO 19156 Regler for modellering av observasjoner og målinger
- ISO 19157 Datakvalitet

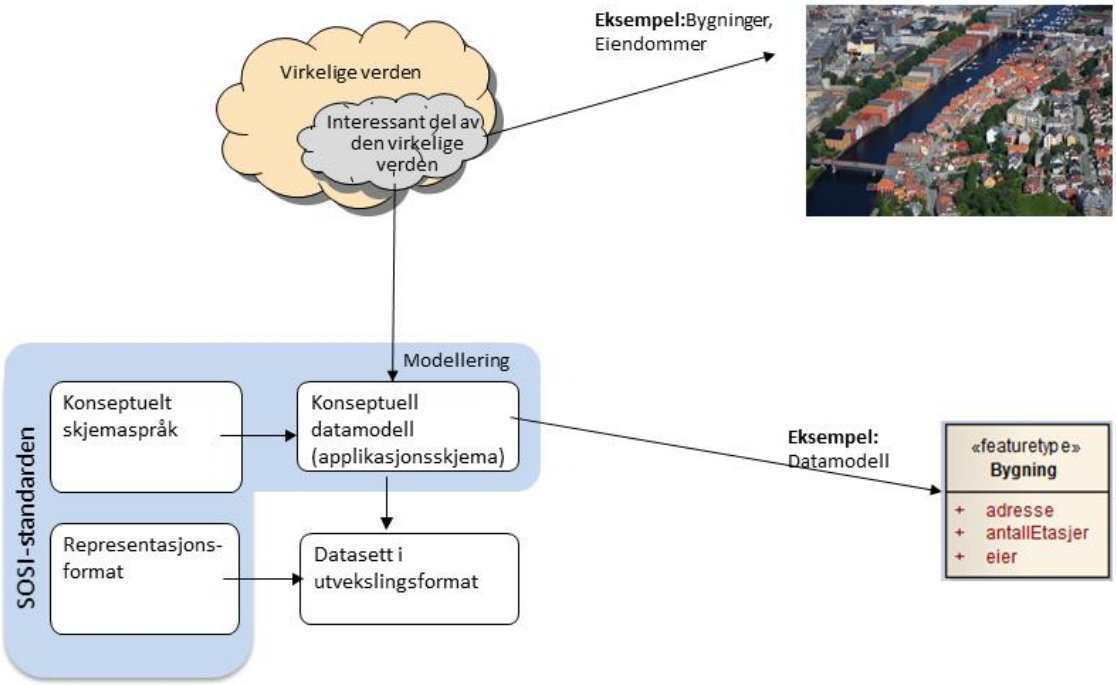
INSPIRE Data Product Specifications er produktspesifikasjoner fordelt på 34 temaer.

INSPIRE Network Services er spesifikasjoner av tjenester.

7.4 Datamodellering

Datamodellering etter SOSI-metoden er vist i Figur 7.6 Datamodellering. I korte trekk går den ut på å beskrive datastrukturen for en utvalgt del av den virkelige verden.

1. Velger ut den ”interessante del” av virkelige verden.
2. Identifiserer objekttyper, egenskaper til og relasjoner mellom objekttypene
3. Modellering. Informasjonen om objekttypene beskrives formelt med et konseptuelt skjemaspråk (UML), som resulterer i en konseptuell datamodell (UML-applikasjonsskjema). Modellering skjer ved bruk av dataverktøy.
4. Ut fra den konseptuelle datamodellen kan en etablere spesifikke representasjonsformater til bruk i datautveksling.



Figur 7.6 Datamodellering

Konseptuelle datamodeller (Applikasjonsskjema)
SOSI-metoden adresserer to typer konseptuelle datamodeller:

- fagmodeller (domene modeller)
- produktspesifikasjoner.

Fagmodellene er modellene som skapes ved prosedyren som er beskrevet ovenfor. Fagmodellene beskriver alle objekttypene, deres egenskaper og relasjoner for et fagområde (”interessant del av virkeligheten”). Fagmodellen vil således være standard forståelsesmodell for fagområdet, og er dokumentert i SOSI del 2 Generell objektkatalog.

Produktspesifikasjoner inneholder konseptuelle modeller som avledes av fagmodellene. Produktspesifikasjonene representerer ofte en avgrensning av den totale informasjonen i fagmodellene, men ofte også en presisering/utvidelse og beskrivelse av spesifikke begrensninger i forhold til fagmodellene.

Eksempel: En produktspesifikasjon for ledningsinnmåling inneholder bare informasjon som måles inn med landmåling (dvs. bare en del av det som en fagmodell kan inneholde), men inneholder også spesifikke krav til nøyaktighet og metoder for innmåling.

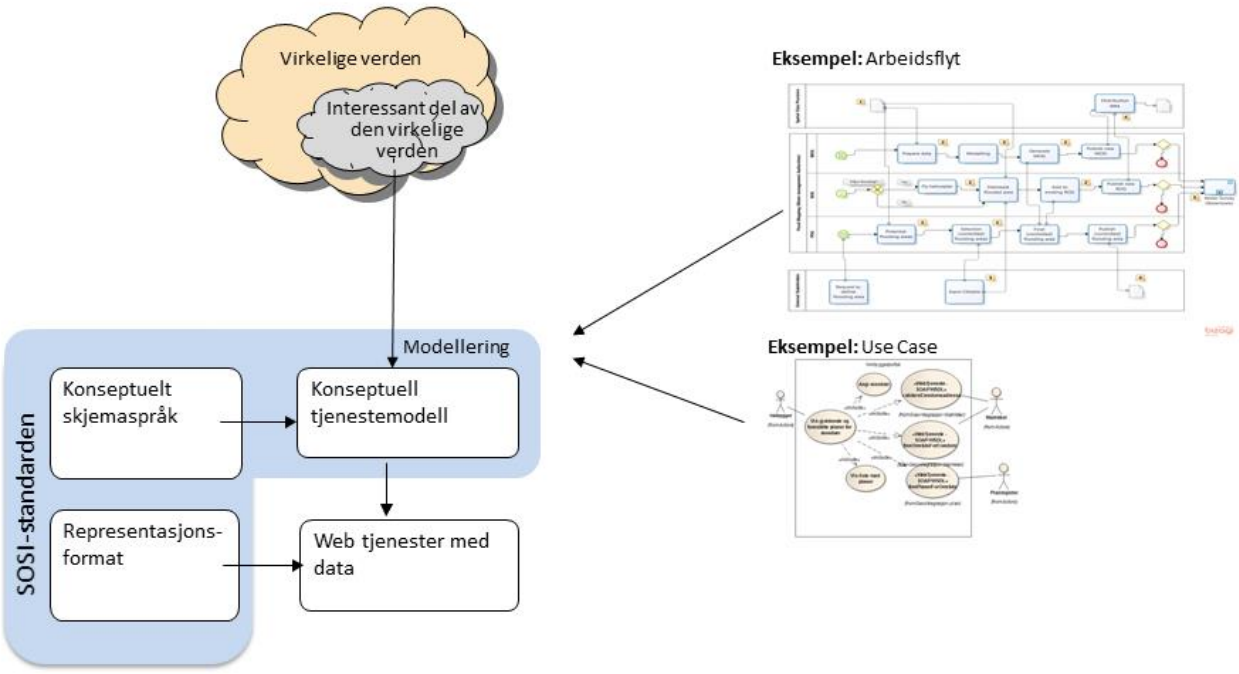
For nærmere detaljer henvises til SOSI del 1 versjon 5 - SOSI produktspesifikasjoner - krav og godkjenning.

Datautvekslingsformat (representasjonsformat) er dataformatet som beskriver datainstansene i en datafil. Det vil etableres minst ett representasjonsformat for hver produktspesifikasjon.

7.5 Tjenestemodellering

Tjenestemodellering etter SOSI-metoden er vist i Figur 7.7 Tjenestemodellering. Hovedtrekkene er som for datamodellering, men den "interessante del av den virkelige verden" går ut på å studere hvordan dataene for et fagområde blir brukt, og ut fra det definere tjenester som skal standardiseres.

1. Studerer hvordan de geografiske data blir brukt i en virksomhet og utveksles mellom virksomheter.
2. For å modellere og dokumentere bruk av dataene anbefales det å benytte prosessdiagrammer (arbeidsflytdiagrammer/workflow diagram) eller "use case"-diagrammer der dette er hensiktsmessig. Hensikten med denne modelleringen er å komme fram til hvilke tjenester som skal spesifiseres. Ref. Kapittel 8.
3. De valgte tjenestene modelleres som en konseptuell tjenestemodell med samme konseptuelle skjemaspråk (UML) som for datamodellene. I tjenestemodellen spesifiseres hvilke data som skal følge med når tjenesten kalles og hvilke data som tjenesten skal avlevere.
4. Ut fra den konseptuelle tjenestemodellen kan en etablere plattformspesifikke tjenestekall med representasjonsformat for tilhørende data.



Figur 7.7 Tjenestemodellering

Konseptuell tjenestemodell er en modell som spesifiserer:

- alle tjenester knyttet til en fagområde
- tjenestenes tilhørende datastrukturer:
  - parametere som styrer bruk av tjenesten
  - data som tjenesten avleverer

Web tjenester spesifiseres for hver aktuelle systemplattform, samt at tjenestens tilhørende data etablerer et representasjonsformat på samme måte som for datamodellering.

SOSI-standarden inneholder standardiserte tjenester innen flere fagområder/ applikasjonsområder som dokumenteres med:

- tekstlig beskrivelse av tjenesten som omfatter identifikasjon, hensikt, bruksområde, etc
- formell UML-modell som spesifiserer tjenestens grensesnitt.

7.6 Konseptuelt modellspråk

SOSI-metoden benytter Unified Modeling Language (UML) som formelt modelleringsspråk å beskrive modellene.

Av UMLs ulike diagramteknikker benyttes hovedsakelig:

- UML klassediagrammer
- UML pakkediagrammer.

og utgjør en stor del av den formelle beskrivelsen av standardene og spesifikasjonene. Dette er nærmere beskrevet i kapittel 10 og 11.

I tillegg brukes

- prosessdiagrammer (f.eks. BPMN 2.0, BPEL, UML)
- UML ”use case”-diagrammer
- UML sekvensdiagrammer
- UML objektdiagrammer
- Object Constraint Language (OCL) (For å beskrive restriksjoner i UML).

til å understøtte og forklare beskrivelsene i standarden når en vurderer dette som nødvendig.

7.7 Implementasjonsplattformer

SOSI-metoden foreskriver automatisk mapping fra plattformuavhengige modeller til plattformspesifikke implementasjonsmodeller. Dette gjelder for utvalgte:

- datautvekslingsformater
    - SOSI-format (prikkformat)
    - GML
    - andre som følger forutsetningene for å bli en del av standarden
- webtjenester
    - REST/OpenAPI
    - SOAP/WSDL
    - WFS/FE
    - Atom feed
    - WCS
    - SOS
    - andre som følger forutsetningene for å bli en del av standarden.

7.8 SOSI-modellregister for geografiske data

En sentral komponent i SOSI-metoden er SOSI-modellregisteret. SOSI-modellregisteret er et SVN-basert register som inneholder alle de viktigste modellene som inngår i den nasjonale geografiske infrastrukturen. For nærmere informasjon, se kapittel 12.

7.9 Harmonisering med andre standarder/spesifikasjoner

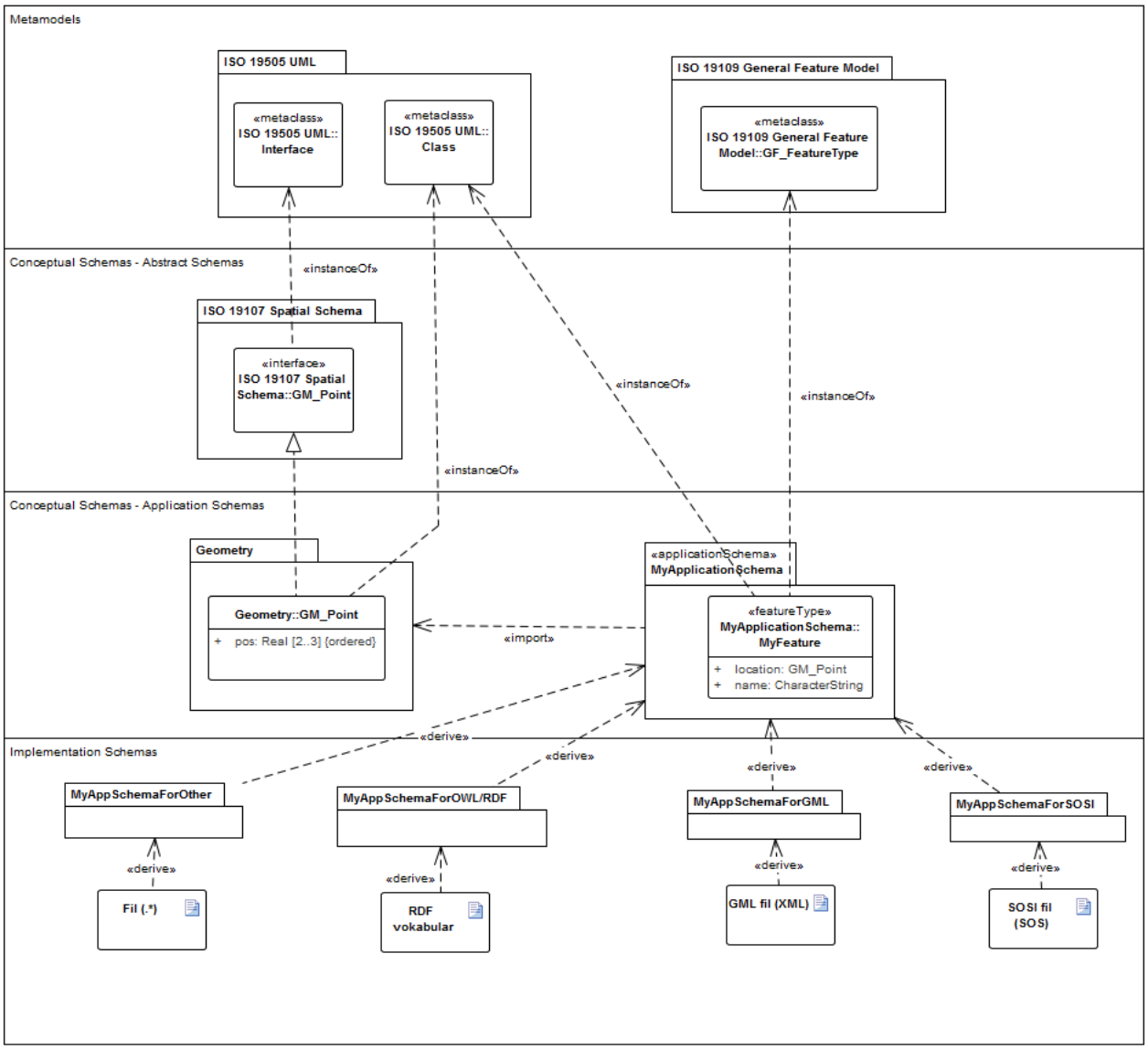
I strategi for det videre arbeidet med SOSI henvises det til at SOSI-modellregister skal være det foretrukne register for all geografisk informasjon. En harmonisering med disse vil på sikt hindre at det fortsatt bygges sektorspesifikke løsninger og sikre gjenbruk av dataelementer og at det er interoperabilitet mellom/med andre standarder/spesifikasjoner, f.eks:

- INSPIRE Annex I - III slik dette er angitt i Geodataloven
- IFC (BuildingSMART)

Tilsvarende vil det være behov for harmonisering med standarder/spesifikasjoner på fagsiden (støydirektivet, vannrammedirektivet, bredbåndsdirektivet, etc.).

7.10 Modeller på ulike abstraksjonsnivåer

Modeller for data og tjenester foreligger på ulike nivåer. Figur 7.8 viser ulike abstraksjonsnivåer i henhold til ISO 19103: 2015 Conceptual schema language med tanke på modellering og implementasjon av UML-applikasjonsskjemaer. Tilsvarende vil det være for tjenester. I forbindelse med SOSI-standardisering benytter vi modeller eller konsepter beskrevet i modeller på alle disse nivåene. Det nederste nivået viser også mappingen videre til data i form av ulike typer filer eller vokabularer, slik som SOSI, GML og RDF.



Figur 7.8 Abstraksjonsnivåer for ulike typer modeller, fra ISO 19103.

I kapittel 10 beskriver vi generell bruk av UML (ISO 19505) og i kapittel 11 beskriver vi bruken av "General Feature Model" (ISO 19109), begge som metamodeller.

I kapittel 11 beskriver vi hvordan vi benytter andre standarder som byggeklosser i et UML-applikasjonsskjema. Figur 7.8 viser også geometriske og topologiske egenskaper jfr. ISO 19107 Spatial schema i et UML applikasjonsnivå, men vil også gjelde for andre typer predefinerte egenskaper, slik som tid og temporale objekter, stedfesting ved hjelp av rasterdata (coverage) etc. Selve standarden (ISO 19107) er definert på et abstrakt konseptuelt skjemanivå med type/interface. Disse realiseres som datatyper på applikasjonsskjema nivå.

UML-applikasjonsskjema inngår i en fagområdestandard og i en produktspesifikasjoner. Det er her fageksperten sammen med en UML-editor blir enige om objekttyper med tilhørende egenskaper, assosiasjoner, andre relasjoner og operasjoner som inngår.

UML-applikasjonsskjema i en fagområdestandard vil utgjøre en fullstendig beskrivelse av innenfor et fagområde. En produktspesifikasjon vil utgjøre et subset av denne samt ytterligere detaljering. En produktspesifikasjons UML-applikasjonsskjema er det som i Figur 7.8 er betegnet MyApplicationSchema: MyFeature, og vil være utgangspunkt for realisering på ulike plattformer.

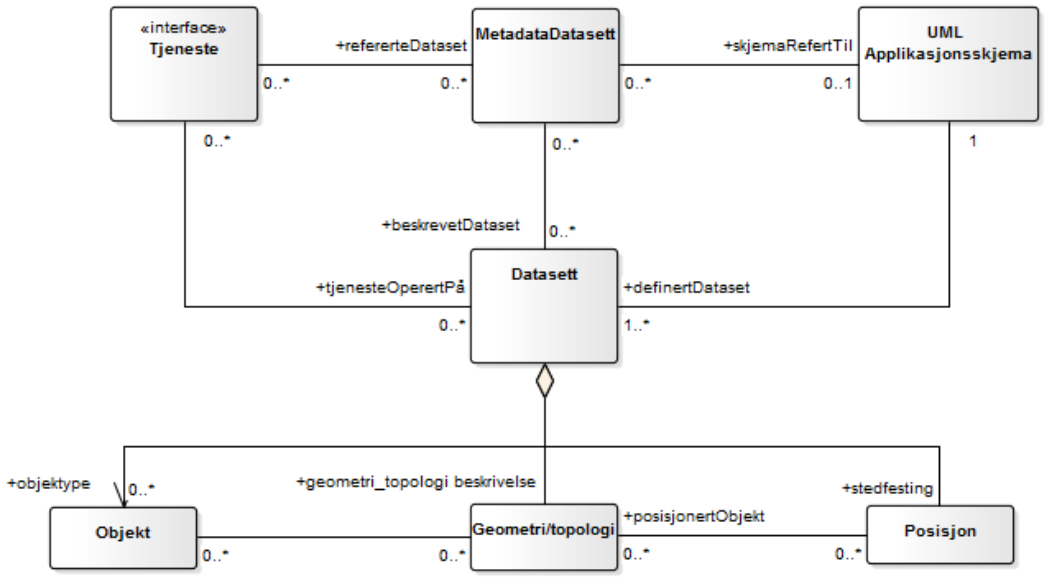
Fram til og med SOSI versjon 4 har vi modellert på en slik måte at UML-applikasjonsskjema til en produktspesifikasjon kan realiseres i både SOSI-format og GML. Av denne grunn har vi ikke fokusert på de implementasjonsspesifikke skjemaene for hver plattform. For SOSI versjon 5 ser en at byggeklossene vil være vesentlig utvidet, og at ikke alle applikasjonsskjema kan representeres i SOSI-format realiseringen, og det vil være behov for å utdype hvilke deler av et produkt som en kan få i SOSI-formatet. Dette gjøres gjennom det vi i Figur 7.8 kaller MyAppSchemaForSOSI. Dersom subsettet i MyAppSchemaForSOSI kan beskrives i form objekttyper som ikke kan leveres, kan dette angis i produktspesifikasjonen under kapittel "Leveranseinformasjon". Dersom det er et mer komplisert subset må en i noen tilfeller lage et implementasjonsspesifikt UML-applikasjonsskjema.

Et UML-applikasjonsskjema kan dokumenteres i henhold til krav i standarden SOSI produktspesifikasjoner - krav og godkjenning, samt enklere dokumentasjon, slik som faktaark.

7.11 Forholdet mellom UML-applikasjonskjema, datasett, metadata og tjenester

Figur 7.9 viser hvordan et UML-applikasjonsskjema inngår i et oversiktsdiagram over de viktigste komponentene som inngår i data og tjenester.

Denne modellen er basert på ISO 19101-1:2014 Reference Model -Part 1: Fundamentals.



Figur 7.9 ISO 19101-2_2015 Reference model - Par1:2014 Fundamentals

Figur 7.9 inneholder:

- **Datasett,** som inkluderer:
  - **objekter** med tilhørende egenskaper, relasjoner (assosiasjoner) og operasjoner
  - **geometri/topologi** for beskrivelse av en objekttypes romlig egenskaper, i form av vektor- eller rasterdata
  - beskrivelse av **posisjon** for geometri/topologibeskrivelsen, herunder hvilket geodetisk referansesystem som posisjonene er basert på.
- **UML-applikasjonsskjema**, som beskriver objekter som inngår i datasettet. Applikasjonsskjema identifiserer hvilke geometri og/eller topologityper samt referansesystem som er nødvendig for å gi en komplett beskrivelse av den geografiske informasjonen i datasettet.
- **Metadata-datasett**, som gir brukere mulighet til å søke, evaluere, sammenligne og bestille (laste ned) geografiske data og/eller påkalle tjenester.
- **Tjeneste**, implementert som tjenesteapplikasjon som opererer på geografiske data som finnes i datasettet. Kan benytte metadata for å sikre korrekt utførelse av tjenestene.

8 Modellering av brukstilfeller og forretningsprosesser

8.1 Innledning

Utvikling av modeller i SOSI-standarder, produktspesifikasjoner og andre dokumenter krever stor forståelse for hvordan data blir brukt i ulike virksomheter. Følgelig legges det opp til en brukerorientert tilnærming for valg og spesifikasjon av de standardiserte tjenestene og applikasjonsskjemaene. I det ligger at forut for spesifikasjonsarbeidet gjennomføres en analyse og beskrivelse av aktuelle behov i form av diagrammer som illustrerer behov for samspill mellom systemer ved hjelp av tjenester, slik som:

- brukstilfeller (”use case”)
- forretningsprosesser (Business Processes)
- sekvensdiagram

Beskrivelsen av brukstilfeller og forretningsprosesser har todelt hensikt.

- Forståelse**,** diagrammene dokumenterer behovet for data og tjenester slik at utenforstående personer (f.eks. beslutningstakere, brukere, utviklere, etc.) kan forstå problemstillingen.
- Identifikasjon av aktuelle webtjenester, hvilken kategori de tilhører, og grovt sett informasjonen som utveksles.

Modeller av brukstilfeller fokuserer på funksjonalitet til systemene som støtter opp om forretningsprosessene.

Modeller av forretningsprosesser fokuserer på arbeidsflyt og dataflyt sett i forhold til virksomhetens forretning.

Et sekvensdiagram er et diagram som viser objekter som kommuniserer i et brukstilfelle og rekkefølgen i kommunikasjonen gjennom transaksjonens livsløp.

8.2 Forholdet til TOGAF

I prosjektet ”målbilde og strategier” i regi av DIFI arbeides det med å etablere en felles referansearkitektur for offentlig sektor, basert på beste praksis rammeverk. Disse rammeverkene omfatter den Europeiske referanseinfrastrukturen (EIRA), TOGAF som prosess og tilnærming, og ArchiMate som notasjon.

TOGAF er et rammeverk, en detaljert metodikk og et sett av verktøy, for spesifikasjon av en virksomhetsarkitektur.

ArchiMate er et uavhengig modelleringsspråk for virksomhetsmodellering og er sterkt knyttet opp mot TOGAF.

8.3 Brukstilfeller

8.3.1 Hensikt

SOSI-metoden anbefaler å beskrive brukstilfeller i forbindelse med spesifikasjon av tjenester og applikasjonsskjema. Før en starter med spesifikasjoner er det viktig at de forankres til noen aktuelle brukstilfeller (”use case”), hvor det framkommer at de inngår som en nødvendig komponent. Denne forankringen vil i seg selv være en god begrunnelse for å realisere tjenesten eller datamodellen. Dette forsterkes hvis samme behov går igjen i flere brukstilfeller. I sistnevnte situasjon, vil det også gi godt grunnlag for å spesifisere tjenesten eller applikasjonsskjemaet generell og robust slik at de kan gjenbrukes i flere brukstilfeller.

8.3.2 Hva er et brukstilfelle (”use case”)

Et brukstilfelle er en beskrivelse av (del-)funksjonaliteten i et system sett fra et eksternt perspektiv. I stedet for å beskrive detaljerte egenskaper med systemet, er hensikten å vise funksjonalitet på et overordnet nivå. Funksjonaliteten brytes ned i et antall oversiktlige brukstilfeller.

Eksempler. I et biblioteksystem er operasjonene knyttet til ”Å låne en bok” et brukstilfelle. I et banksystem er ”Ta ut penger” et brukstilfelle.

Et brukstilfelle er vanligvis en mer detaljert beskrivelse av en aktivitet i et prosessdiagram. Et prosessdiagram kan inneholde mange brukstilfeller.

8.3.3 Forholdet mellom brukstilfeller (”use case”) og brukerreiser.

En ”use case”-spesifikasjon har også likhetstrekk med begrepet **brukerreiser.** Brukerreiser er en kort, enkel beskrivelse av ønsket funksjonalitet sett fra brukerens perspektiv, formulert som:

> *”Som <type bruker> ønsker jeg å gjøre <funksjon 1> og <funksjons2> slik at jeg oppnår <mål>”*

Utgangspunktet for en ”use case”-model er likt med brukerreiser, men en ”use case”-modell analyserer brukerbehovet nærmere og beskriver i mer detalj logisk realisering av brukstilfellet. Brukerreiser har derimot ingen spesiell grafisk notasjon og er ment å være selvforklarende.

8.3.4 Beskrive brukstilfeller i SOSI-metoden

Beskrivelser av brukstilfeller i SOSI-metoden skal vise typiske eksempler på aktuell brukerfunksjonalitet hvor de spesifiserte tjenestene og dataene inngår. Beskrivelsen skal:

- vise systemet som bruker tjenestene på et oversiktlig nivå
- vise brukere og systemer som er tjenesteytere til systemet (aktører)
- vise tjenestene som egne ”use case” i beskrivelsen.

Brukstilfeller kan beskrives i form av:

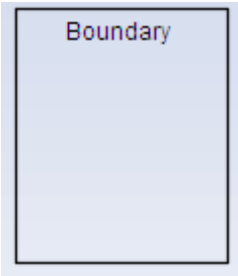
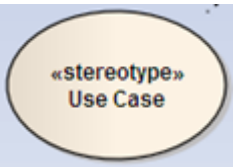
- tekstlig beskrivelse
- grafisk beskrivelse som UML ”use case”-diagram.

8.3.5 UML ”use case”

UML tilbyr egen grafisk notasjon for ”use case”-diagrammer. Et ”use case”-diagram i UML har fire hovedelementer, vist i Tabell 8.1.

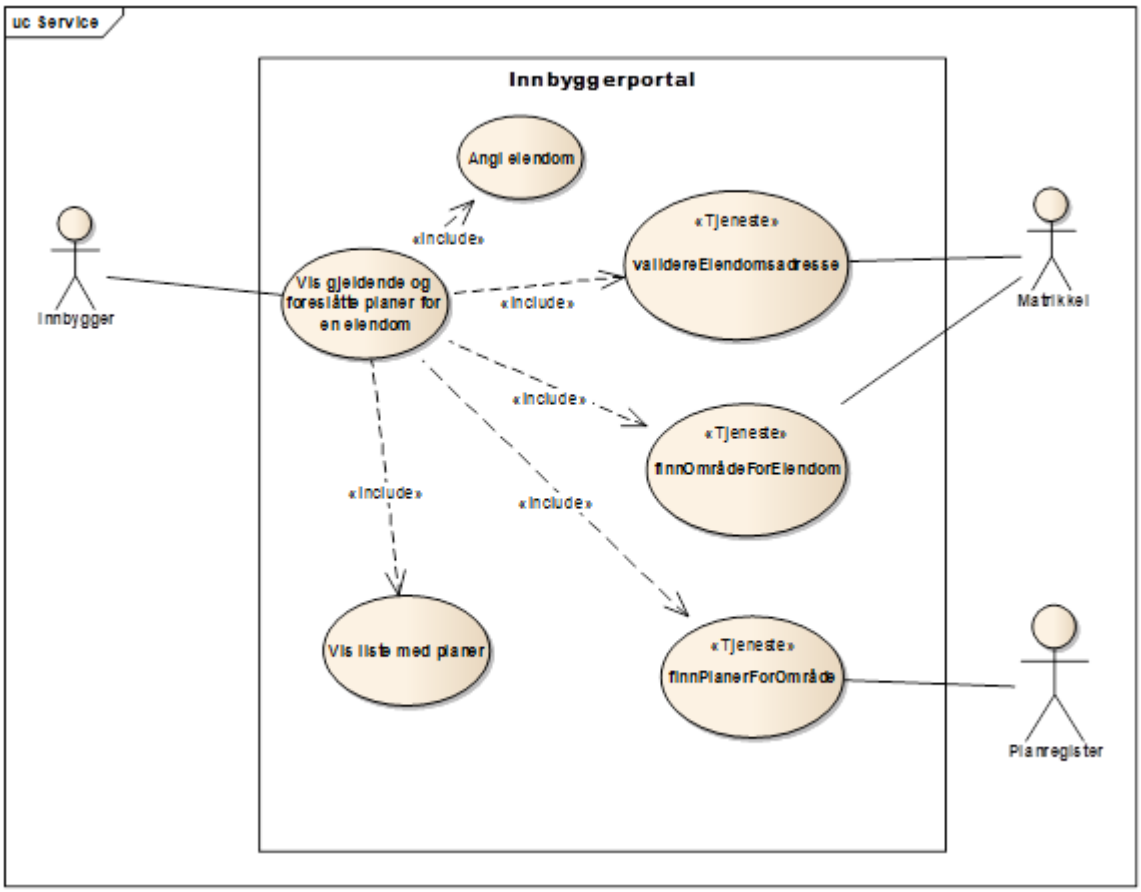
Tabell 8.1 Elementer i UML- ”use case”-diagrammer

Actor		Representerer en bruker eller et system som ligger utenfor det aktuelle system som håndterer brukstilfellet. Aktør angis med navn

Boundary		Viser systemets avgrensning. Alle aktører plasseres utenfor avgrensingen, mens alle brukstilfeller ligger innenfor
"Use case"		Representerer en avgrenset funksjonalitetet og angis med et navn. Hvis brukstilfellet gjelder en tjeneste som leveres av en aktør, angis <<Tjeneste>> som stereotype. Brukstilfeller som i helhet utføres innenfor systemet gis ingen stereotype.
Relationship		Relasjon mellom aktør og brukstilfelle, samt mellom brukstilfeller. Ulike typer:
		*Use* relasjonen brukes for å vise forholdet mellom aktør og brukstilfelle
		*Include* relasjonen brukes for å vise nedbryting av et brukstilfelle i et antall mer detaljerte brukstilfeller, som alle blir brukt minst en gang i hvert brukstilfelle
		*Extend* relasjonen brukes for å vise nedbryting av et brukstilfelle i et antall spesialiserte brukstilfeller, hvorav bare en blir brukt i hvert brukstilfelle

Eksempel: Figur 8.1 viser brukstilfelle der en "Innbyggerportal" gir brukere mulighet til å få en liste over alle planer mot Matrikkelen og Planregisteret. Brukerreisen kan være formulert slik: *"Som Innbygger ønsker jeg å benytte Innbyggerportalen til å få oversikt over alle reguleringsplaner som berører min eiendom"*

Tjenestene er angitt som egne "use case" og merket med stereotype "Tjeneste".



Figur 8.1 Brukstilfelle for innbyggerportal.

8.3.6 Anbefalinger

Anbefaling: BrukstilfelleGrafisk	UML versjon 2, ”use case” - diagramteknikk, bør benyttes for å beskrive den grafiske prosessen som identifiserer de overordnede brukerkravene.

I SOSI har vi tatt utgangspunkt i en mal (template) som var laget for CEN-TR 15449-4 – Spatial Data Infrastructure – Part 4 Service centric view. Denne er igjen videreført i regi av ISO 19119 Services. I tillegg ser vi at det også vil være behov for enklere maler, med utgangspunkt i INSPIRE.

For de fleste ”use case” er det ikke tilstrekkelig med et UML-diagram. Anneks B beskriver to maler for brukstilfeller, i form av tabeller.

Anbefaling/BrukstilfelleMal	Det anbefales å bruke en av malene beskrevet i Vedlegg B for dokumentasjon av «use case»

8.4 Forretningsprosesser

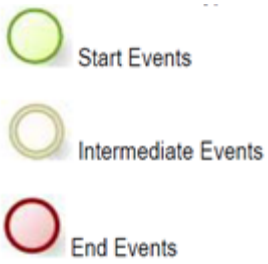
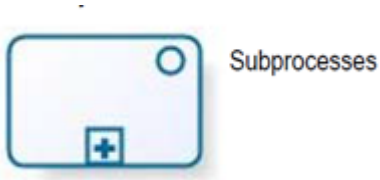
En **forretningsprosess (Business Process)** er en samling relaterte aktiviteter i en eller flere virksomheter som tilsammen produserer en spesifikk tjeneste eller produkt for en kunde. En aktivitet er en avgrenset arbeidsoppgave innenfor forretningsprosessen. Aktiviteter med komplekse arbeidsoppgaver kan dekomponeres og beskrives som en sub prosess med egne aktiviteter. Arbeidet innenfor en aktivitet støttes ofte av datasystemer som krever kommunikasjon og datautveksling med andre systemer som brukes i andre aktiviteter. Meldinger og datautveksling mellom aktiviteten er ofte kandidater til å bli realisert som tjenester.

Formelle beskrivelser av forretningsprosesser vil være en god måte å forklare, begrunne og understøtte valg av tjenester, og hvordan tjenestene er tenkt brukt.

8.4.1 BPMN 2.0 prosessdiagrammer - notasjon

BPMN 2.0 er en standard for beskrivelse av forretningsprosesser. Den definerer en grafisk notasjon for å beskrive forretningsprosesser. Metoden har mye til felles med tradisjonelle flytskjema som er og har vært viktig teknikk for å beskrive handlingsforløpet i et dataprogram. De grafiske symbolene som er definert i BPMN 2.0 kan kategoriseres i 5 hovedgrupper som vist i Tabell 8.2 Grafiske symboler i BPMN 2.0.

Tabell 8.2 Grafiske symboler i BPMN 2.0.

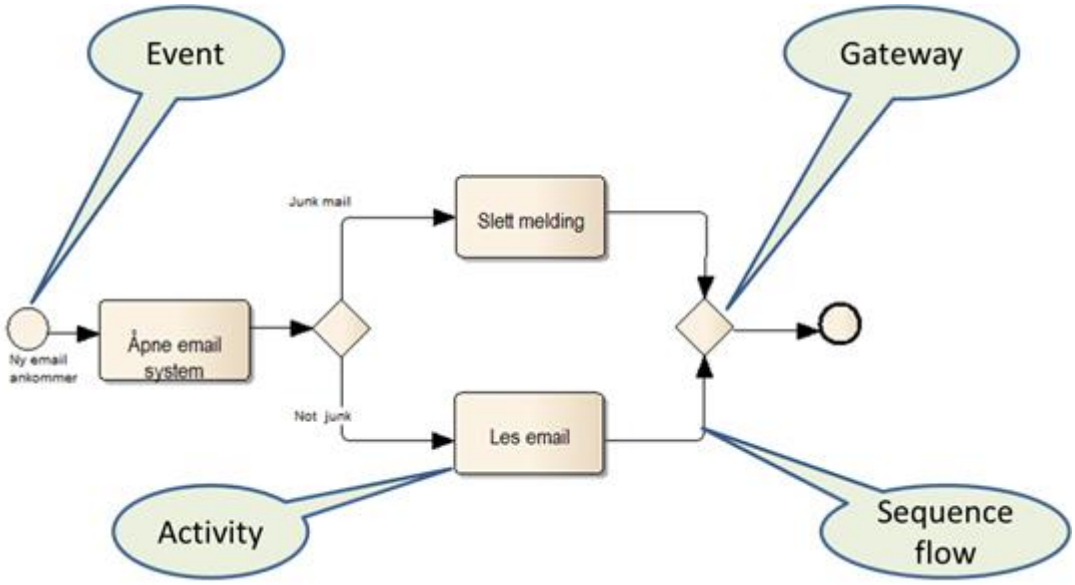
Event (hendelser)		Representerer hendelser i en forretningsprosess – at noe skjer. Start, slutt og hendelser i løpet av prosessflyten BPMN foreskriver mange måter å starte og avslutte en prosess, og det er også flere måter å angi hva som skjer under utførelsen av en prosess.
Activity (aktiviteter)		Representerer et stykke arbeid som skal utføres som en del av forretningsprosessen. Det er to hovedtyper aktiviteter • Oppgave (Task). • Subprosess (Sub process).
		Oppgave som beskrives med alle detaljer
		Aktivitet, men er beskrevet mer detaljert som eget prosessdiagram
Gateway		Modellelementer som viser/styrer oppsplitting eller sammenslåing av prosessflyten ut fra ulike betingelser. Gateway finnes i form av: • exclusive gateway • gateway based on event • parallell gateway • inclusive gateway • complex gateway.
Connection objects		Brukes til å forbinde objekter i prosessflyten

		**Process flow** viser arbeidsflyt/ prosessflyt
		**Message flow** viser meldingsutveksling mellom aktiviteter
Swim lanes		Brukes til organisere prosessflyten visuelt til ulike funksjonsområder, roller, ansvarsområder eller selvstendige virksomheter, i form av: • pool • lane

I tillegg tilbys muligheter til å inkludere tekstlige/grafiske beskrivelser i diagrammet:
- gruppering av objekter
- annotations (merknader)
- dataobjekter (som representerer databaser, datasett, datakilder etc.)

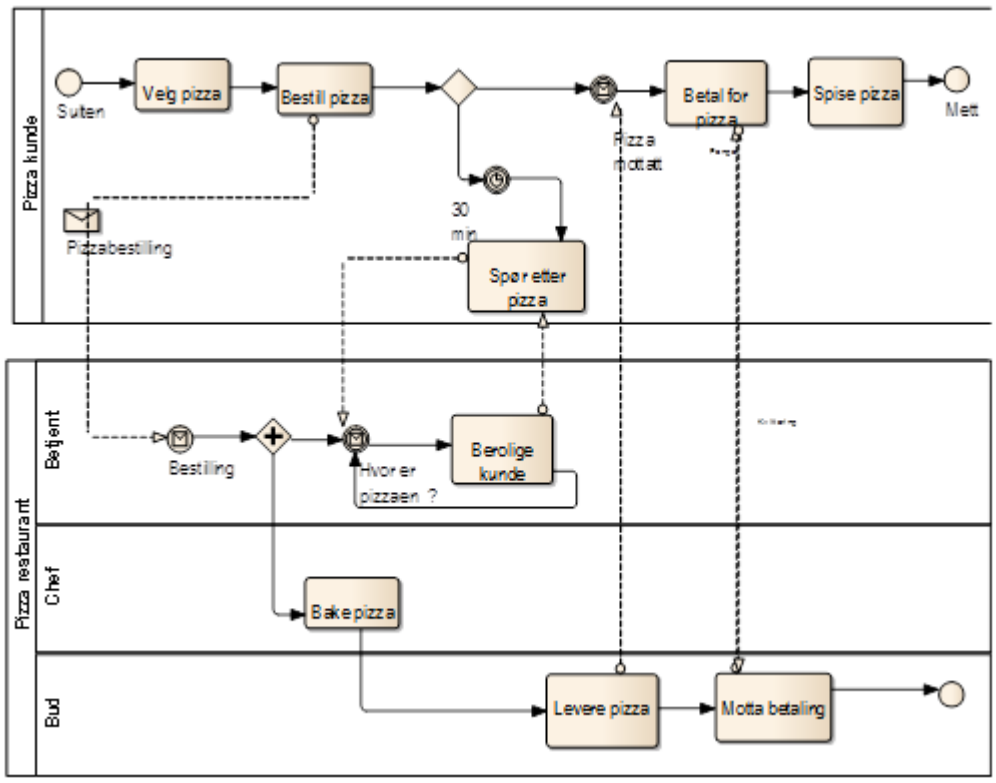
8.4.2 BPMN 2.0 prosessdiagrammer - eksempler

Eksempel 1: Figur 8.2 viser bruk av de ulike symbolene.



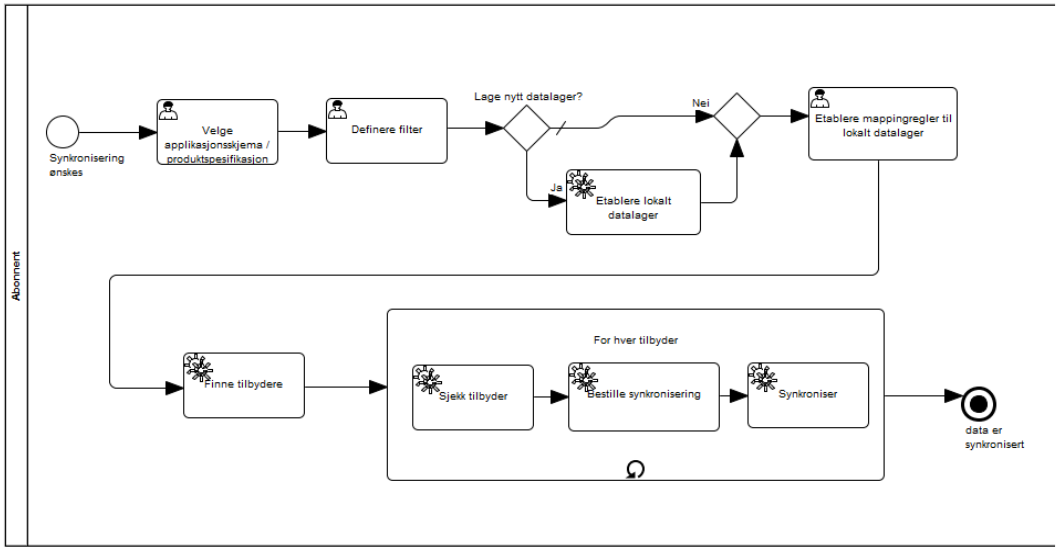
Figur 8.2 Eksempel på prosessdiagram.

Eksempel 2: Figur 8.3 er et prosessdiagram som viser prosessen for bestilling og leveranse av pizza.



Figur 8.3 Eksempel på prosessdiagram for bestilling og leveranse av pizza.

Eksempel 3: Figur 8.4 viser en oversikt over forretningsprosessen for geosynkronisering, tatt fra arbeidet med geosynkroniseringsstandarden.



Figur 8.4 Prosessdiagram for geosynkronisering.

8.4.3 BPMN 2.0 Prosessdiagrammer - anbefalinger

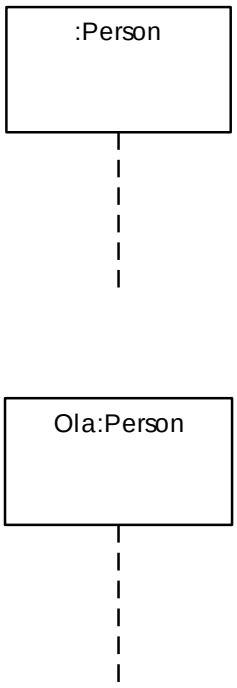
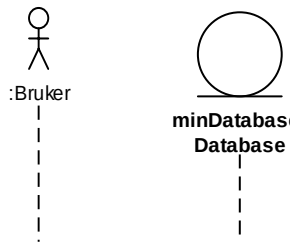
Anbefaling/forretningsprosesser	Ved modellering av forretningsprosesser anbefales det å bruke prosessdiagrammer (BPMN 2.0 eller Archimate, alternativt UML activity diagram eller sekvensdiagram). BPMN bør benyttes der en ønsker å automatisere prosesser for

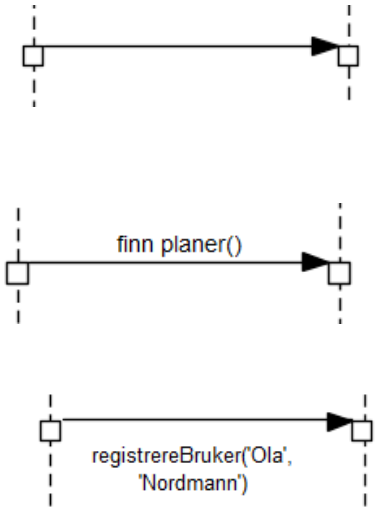
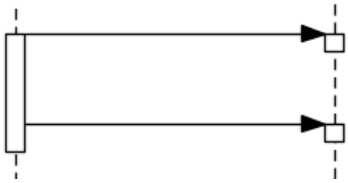
	videre bruk, f.eks gjennom BPEL (Business Process Execution Language).

8.5 Sekvensdiagrammer

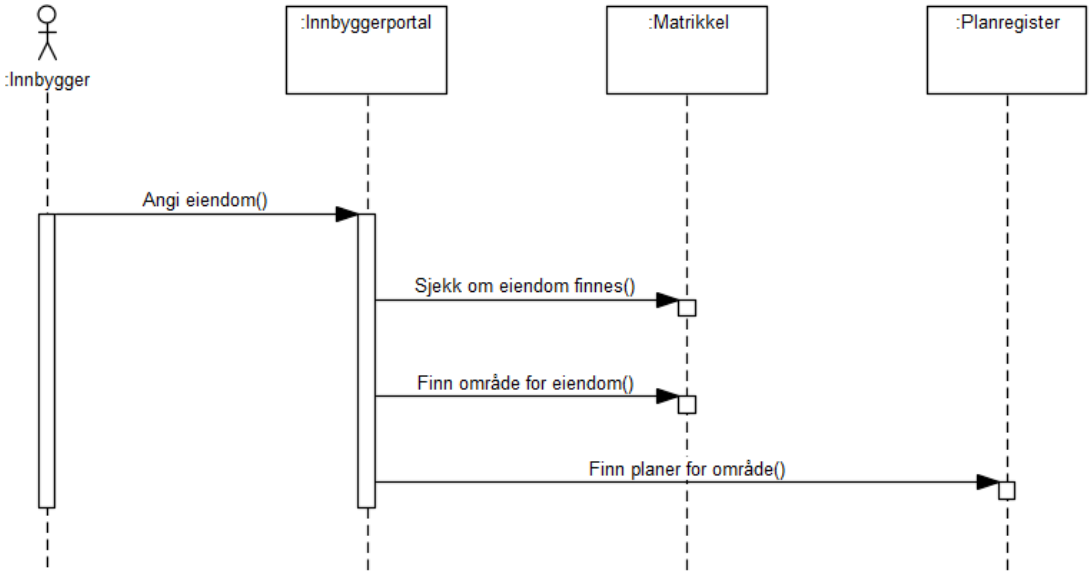
Et sekvensdiagram er et diagram som viser objekter som kommuniserer i et brukstilfelle og rekkefølgen i kommunikasjonen gjennom transaksjonens livsløp. Sekvensdiagrammer er gode til å vise hvilke objekter som kommuniserer med hvilke andre objekter, og hvilke meldinger som trigger kommunikasjonen og rekkefølgen. Sekvensdiagram er ikke ment å vise kompleks logikk. I SOSI-standarden kan et sekvensdiagram enkelt forklare bruk av tjenester. Noen av de viktigste elementene som kan brukes i et sekvensdiagram er vist i Tabell 8.3 Symboler i sekvensdiagrammer. For en fullstendig oversikt refereres det til OMG UML 2.5 (http://www.omg.org/spec/UML/2.5/PDF).

Tabell 8.3 Symboler i sekvensdiagrammer.

Lifeline		Hver deltaker i et brukstilfelle eller en transaksjon er representert med en Lifeline (livsløp). Alle deltakere i et sekvensdiagram er på instansnivå. Vanligvis vises deltakeren som et rektangel med kolon og deltakerens type (representerer en uspesifisert instans, eksempel øverst). Dersom det er behov for å spesifisere en konkret instans, kan instansnavnet skrives før kolon i rektangelet (eksempel nederst). En vertikal stiplet linje under rektangelet viser livsløpet.
		Ofte tegnes en deltakers lifeline også med symboler for brukere og datakilder. I tillegg er definert egne symboler for kontroll og grensesnitt.

Messages		Meldinger som representerer kommunikasjon mellom deltakerne vises som piler (eksempel øverst), gjerne med meldingsnavn (eksempel i midten). Representerer meldingen et operasjonskall, kan meldingsnavnet være identisk med signaturen til operasjonen samt argumenter for operasjonsparametre. Det nederste eksempelet viser en registreringsoperasjon som krever fornavn og etternavn som parametre.
ExecutionSpecification		Operasjonen framkommer som et smalt rektangel på livsløpet og viser at deltakeren er aktiv og utfører en operasjon.

Eksempel: Figur 8.5 viser et sekvensdiagram av et brukstilfelle hvor en bruker gis mulighet til å få en liste over alle planer som berører hans eiendom ved å kommunisere med Matrikkelen og Planregisteret.



Figur 8.5 Sekvensdiagram av et brukstilfelle.

8.5.1 Anbefalinger

Anbefaling/Sekvensdiagram	Det anbefales å bruke UML sekvensdiagram dersom det er behov for ytterligere presisering av et brukertilfelle, for eksempel rekkefølge på prosesser.

Et gjennomgående eksempel på bruk av prosessdiagram, "use case" - diagram og klassediagram er vist i Vedlegg C.

9 Modellering av tjenester

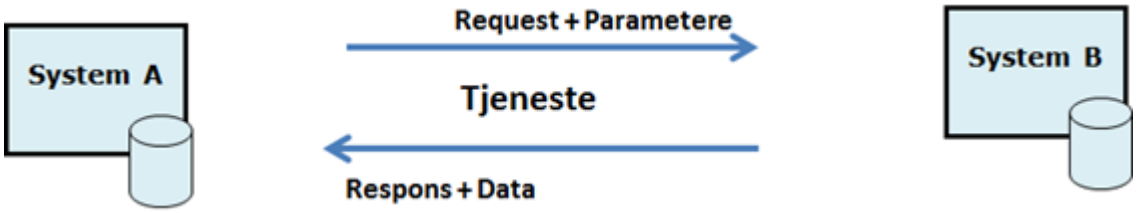
9.1 Introduksjon

Den raske utviklingen innen geografisk informasjonsteknologi, og også bruken av disse data innenfor andre domener, har resultert i utviklingen av en rekke tjenester, både for å hente data, integrere data i forretningsprosesser og for å utføre predefinerte prosesser/analyser på data.

En tjeneste er en funksjon og/eller dataleveranse som en applikasjon (system) tilbyr en annen applikasjon via et veldefinert grensesnitt og som er tilgjengelig for mange applikasjoner. Systemet som er leverandør av tjenesten vet nødvendigvis ikke hvem som er konsument av tjenesten.

En tjeneste består av en eller flere funksjoner som et system tilbyr andre systemer via et definert grensesnitt. En tjeneste er prinsipielt en maskin til maskin kommunikasjon – interoperabilitet mellom systemer/applikasjoner.

Figur 9.1 viser s*ystem A* som påkaller s*ystem B* med et tjenestekall (request), og får svar (respons).



Figur 9.1 Tjeneste med tjenestekall og svar.

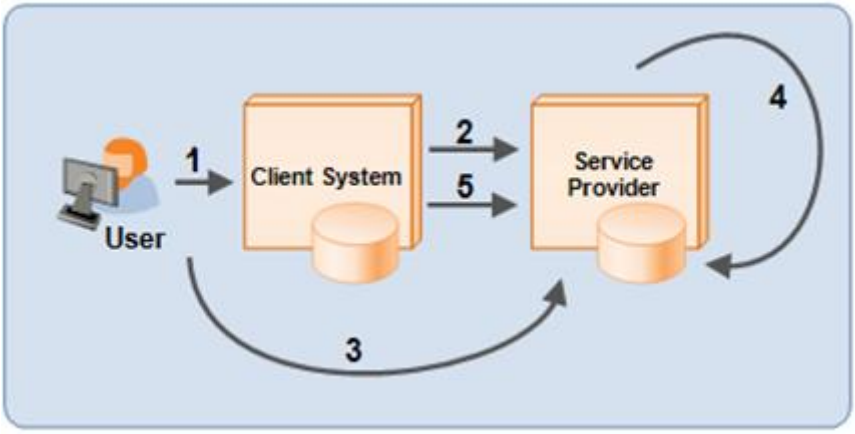
Parametrene i ”request” inneholder:
- hvilken funksjonalitet (metode) som skal utføres
- inputdata til tjenesten.

Data i responsen inneholder:
- opplysning om tjenesten ble vellykket utført eller ikke
- resultatene fra tjenestens operasjoner.

Det skilles mellom to typer enkle, enveistjenester:
- tjenester under kontroll
- tjenester med kontrolloverføring.

Tjenester under kontroll betyr at System A sender en ”request” og venter til det får respons fra System B. Eksempel: Brukeren på System A skal fylle inn en adresse, og taster inn de første bokstaven i navnet. En tjeneste i System B gir en liste over aktuelle adresser.

Tjenester med kontrolloverføring er når System A påkaller System B og samtidig overfører kontroll til System B. System B vil vanligvis tilby et eget brukergrensesnitt hvor brukeren kan utføre operasjoner og etablere informasjon i System B. Vanligvis kalt i form av en URL med parametere. Ofte vil det være behov for å overføre resultater fra brukerfunksjonene i System B til System A. Eksempel: System A har behov for å registrere geografisk posisjon for et objekt. System B kan gi denne opplysningen ved at brukeren peker på posisjonen i brukergrensesnittet for System B.



Figur 9.2 – Tjeneste med kontrolloverføring

Figur 9.2 viser prinsippene for bruk av tjenester med kontrolloverføring, og beskriver følgende:

1. Brukeren arbeider initielt på System A (Client System).
2. I løpet av sesjonen, ønsker System A (Client System) data fra System B (Service Provider) som må velges ut under brukerkontroll. System B (Service Provider) påkalles ved en tjeneste med kontrolloverføring.
3. System B (Service Provider) vil åpne en bruker-sesjon parallelt med System A (Client System), hvor brukeren har aksess til å søke, velge ut og produsere resultater.

Hvis nye data er generert i System B (Service Provider),

4. Resultatene lagres i System B (Service Provider).
5. System A (Client System) må aktivt forespørre resultatene ved egen tjeneste under kontroll.

Ettersom System B (Service Provider) ikke ”vet” hvilken applikasjon som har startet tjenesten, må ”request” parameteren inneholde unik identifikasjon for System A (Client System) for å ivareta brukerintegriteten.

9.2 Forankring

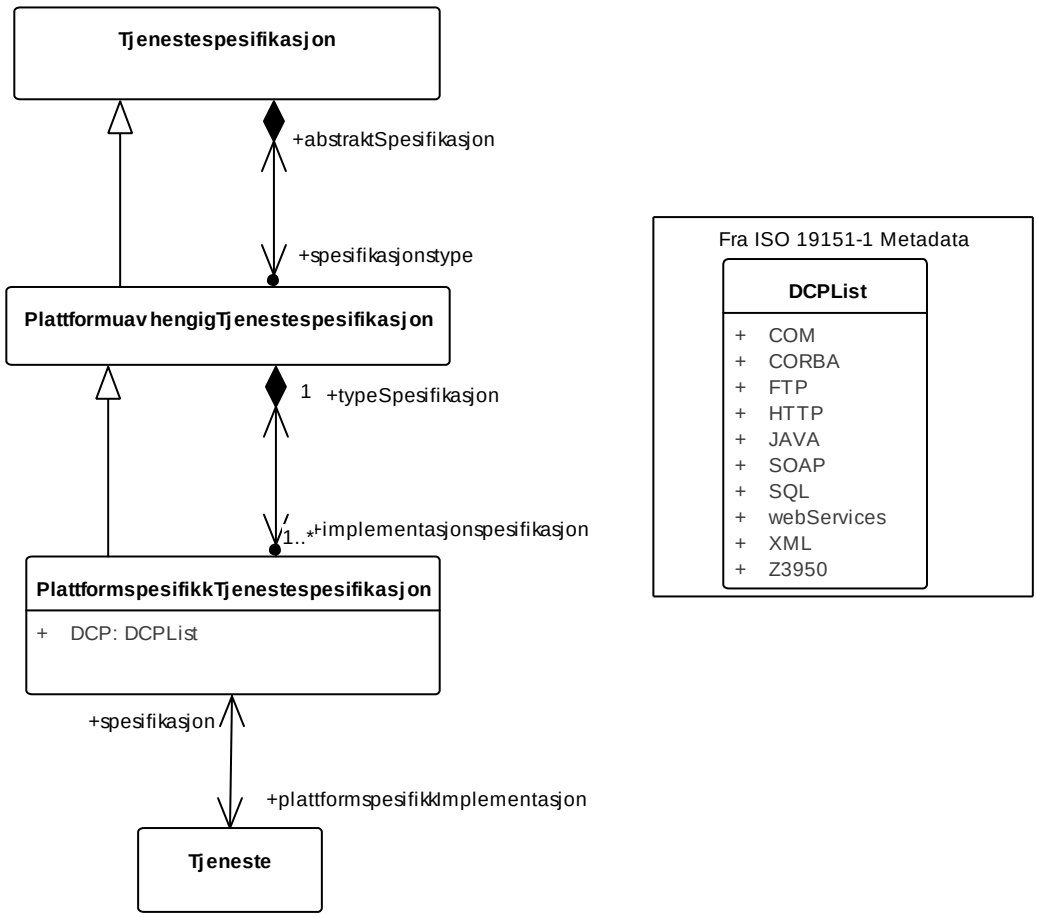
Dette kapitlet er basert på ISO 19119:2015 Services og Geointegrasjonsstandarden, og beskriver et rammeverk for plattformuavhengige(plattformnøytrale) og plattformspesifikke tjenester, som vil gjøre brukere i stand til å hente, prosessere og håndtere geografiske data fra ulike kilder, også ulike distribuerte plattformer (DCP - Distributed Computing Platform).

ISO 19119:2015 Services skiller spesifikasjonene på tre nivåer, gitt i Tabell 9.1. Figur 9.3 – Fra plattformuvhengig modell til implementasjon viser forholdet mellom disse i form av en UML-modell.

Tabell 9.1 Nivåer beskrevet i ISO 19119 Services.

Plattformuavhengige spesifikasjoner	En tjeneste som spesifiserer en spesifikk geografisk tjeneste på en plattformnøytral måte, i samsvar med kravene for ”Enterprise-”, ”Computational-” og ”Information viewpoint”.
Plattformspesifikke spesifikasjoner	Plattform og språkspesifikke tjenester som avledes fra plattformnøytrale tjenestespesifikasjoner, i samsvar med kravene i ”Engineering” og ”Technology viewpoints”.
Plattformspesifikke implementasjoner	En aktuell implementasjon av en tjeneste i henhold til kravene i ”Technology viewpoint” og som kan

	konformitetstestes mot plattformspesifikke tjenestspesifikasjoner.



Figur 9.3 – Fra plattformuvhengig modell til implementasjoner

/req/computational viewpoint/interfaces	Tjenester skal beskrives på en plattformuavhengig måte. Tjenester skal modelleres som UML-klasser med stereotypen ”interface”.

Tabell 9.2 viser eksempel på tjeneste fra et adresseregister.

Tabell 9.2 Eksempel på en tjeneste fra et adresseregister.

Tjeneste	«interface» **Adressetjeneste**	Tjenester fra et adresseregister modellert som en UML-klasse med stereotypen ”interface”.

Spesifikasjonen av tjenester i SOSI-standarden følger i hovedtrekk ISO19119, men avgrenses til enveistjenester. I denne versjonen er ikke toveistjenester og lenkede tjenester beskrevet. For mer avanserte tjenester (f.eks tjenestekjeding eller choreography) henvises til ISO 19119 (2015) Services. Det forventes at revisjoner av SOSI del 1 versjon 5 vil utvides til også å handtere denne type tjenester, med eksempler.

På konseptuelt nivå er prosedyren for tjenesteutvikling i SOSI harmonisert med den norske tjenestestandarden Geointegrasjon.

9.3 Beskrivelse av tjenesten

Følgende opplysninger inngår i spesifikasjonene for tjenester:

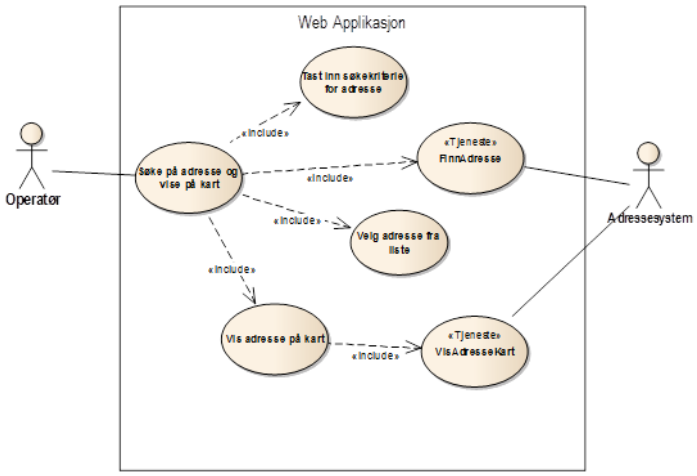
- Overordnet tekstlig beskrivelse av tjenesten og dens funksjoner (Enterprise ViewPoint).
- Detaljert beskrivelse av tjenestens funksjonalitet (operasjoner) (Computational ViewPoint).
- Detaljert beskrivelse av data (input parameter, resultatdata) knyttet til tjenesten.

Overordnet tekstlig beskrivelse av tjenesten legges inn som tekstlig beskrivelse i UML-klassen, og skal omfatte:

/req/enterpriseviewpoint/ servicename	Tjenesten skal inneholde navn som tekststreng
/req/enterpriseviewpoint/ servicetypes	Tjenesten skal inneholde kategori = En-veis tjeneste
/req/enterpriseviewpoint/ purpose	Tjenesten skal inneholde hensikt med tjenesten
/req/enterpriseviewpoint/ scope	Tjenesten skal inneholde bruksområde som dekkes av tjenestens ulike funksjoner, samt hvilke bruksområder som ikke dekkes
/req/enterpriseviewpoint/ community	Tjenesten skal inneholde liste over potensielle brukere av tjenesten
/rec/enterpriseviewpoint/ capabilities	Tjenesten bør inneholde liste over tjenestens operasjoner (metoder)
/rec/enterpriseviewpoint/ scenarios	Tjenesten bør inneholde prosessdiagram eller ”use case” diagram som viser typisk bruk av tjenestens funksjoner

Eksempel: Tabell 9.3 viser at tekstlig beskrivelse av tjenesten er lagt inn i UML-klassens beskrivelsesfelt. Beskrivelsen er støttet med et diagram av et brukstilfelle.

Tabell 9.3 Beskrivelse av en tjeneste, med brukstilfelle.

Tjeneste fra adresse-registeret.	«interface» **Adressetjeneste** **Adressetjeneste : public Class** Author: b511770 Phase: 1.0 Version: 1.0 Navn Adressetjeneste Type tjeneste En-veis Hensikt Tilby brukere av adressedata mulighet for å søke, verifisere og posisjonere adresser Funksjoner - Fritekst og strukturert søk på adresser - Finn nærmeste adresser til en posisjon - Hent adresser innenfor et angitt område - Hent adressekart Målgruppe Alle systemer som håndterer adresser Scenario Referanse til prosessdiagram	Tekstlig beskrivelse av tjenesten er lagt inn i UML-klassens beskrivelsesfelt.
Tjeneste fra adressereg.		Beskrivelsen er støttet med et diagram av et brukstilfelle.

9.4 Operasjoner

/req/computationalviewpoint/ operations	Tjenestens funksjonalitet skal modelleres som operasjoner i UML-klassen som representerer tjenesten.

Tabell 9.4 viser et eksempel på visning av operasjoner

Tabell 9.4 Visning av operasjoner for en tjeneste.

Operasjoner	«interface» **Adressetjeneste** + finnAdresse(): void + visAdresseKart(): void + finnNærmesteAdresse(): void + hentAdresseIOmråde(): void	• finnAdresse – adressesøk • visAdresseKart – vise adresse på kart • finnNærmesteAdresse – nærmeste adresse(r) til en gitt posisjon • hentAdresseIOmråde – hent alle adresser i et område

9.5 RPC og dokument/meldingssentrert parameterstil

Det er to populære modellteknikker for webtjenester i dag, RPC (Remote Procedure Call) og meldingssentrert parameterstil. WSDL 1.1 spesifikasjonen beskriver to SOAP meldingssentrerte parameterstiler, RPC og dokument, som korresponderer med RPC og meldingssentrert parameterstil. Av denne grunn brukes betegnelsen dokument/meldingssentrert parameterstil.

Det er ikke alltid et like klart skille mellom disse modellteknikkene. For eksempel kan en bruke en RPC program modell for å sende og motta dokument parameterstil meldinger. En kan også sende en RPC stil SOAP melding ved å bruke noen XML API (f.eks. DOM eller XmlReader) og deretter sende den mottatte responsmeldingen til andre rutiner for videre prosessering.

På tross av dette er det en sterk tendens blant enkelte leverandører å ta hensyn til meldingsformatet definert i tjenestenes WSDL i selve modelleringen av tjenesten.

I SOSI avgrenser vi ikke disse metodene, men stiller spesielle krav til modeller som er basert på meldingssentrert parameterstil.

/req/informationviewpoint/ operation input/output parameters	Inn/ut parametere til tjenesten i tjenestegrensesnitt skal beskrives som regulære UML-typer for klassisk RPC (Remote Procedure Call) eller sentrert parameterstil modelleres som UML-klasser med «MessageType» som stereotype for dokument-/meldingssentrert parameterstil, begge basert på modelleringsregler i henhold til ISO 19103 og ISO 19109.

For tjenestemodeller som ikke er dokument-/meldingssentrert er andre klasser med stereotyper (f.eks. FeatureType) tillatt som input-/output-parameter.

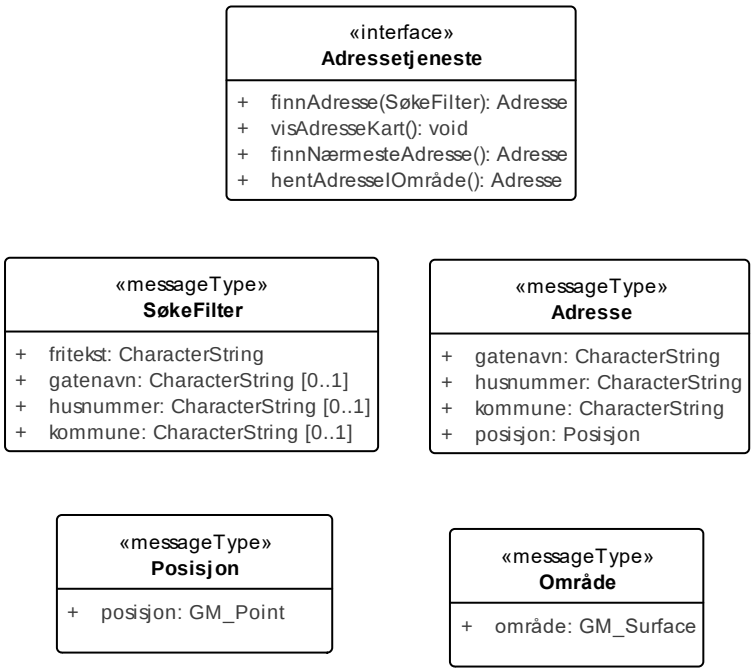
Tabell 9.5 viser eksempler på bruk av «MessageType».

Tabell 9.5 Tjeneste som bruker «messageType»

Parameterstil	«interface» **Adressetjeneste** + finnAdresse(SøkeFilter): Adresse + visAdresseKart(Adresse): void + finnNærmesteAdresse(Posisjon): Adresse + hentAdresseIOmråde(Område): Adresse «messageType» **SøkeFilter** «messageType» **Adresse** «messageType» **Posisjon** «messageType» **Område**	Dokumentasjon/ meldingssentrert parameterstil med bruk av stereotype «messageType».

9.6 Datastrukturer for input/output

Figur 9.4 – Eksempler på datastrukturer for input/output viser et eksempel på operasjoner (input/output) for en adressetjeneste.



Figur 9.4 – Eksempler på datastrukturer for input/output

9.7 Navnekonvensjoner

For navnekonvensjoner på egenskaper, roller, operasjoner og parametere henvises til anbefaling 11 i kapittel 10.

Hvis en tjeneste hører til en av følgende kategorier, anbefales følgende navnekonvensjon for prefikset:

- **finn***Xxx* - søk som resulterer i liste med instanser

- **hen**t*TjenesteNavn* - henter fram data for en instans
- **vis***TjenesteNavn* - viser informasjonen i tjenesteyters GUI
- **ny***TjenesteNavn* - lagrer data for ny instans hos tjenesteyter
- **oppdater***TjenesteNavn* - oppdaterer data for eksisterende instans hos tjenesteyter
- **sjekk***TjenesteNavn* - sjekker om verdi eksisterer
- **slett***TjenesteNavn* – sletter lagret instans

/anbefaling/navnekonvensjon	Hvis en tjeneste hører til en av kategoriene beskrevet under 9.7 anbefales det å bruke disse prefiksene for operasjoner modellert i en klasse

9.8 Sekvens/rekkefølge av tjenester og tjenestenes funksjoner

/req/computationalviewpoint/behaviour	UML sekvensdiagram brukes for å vise sekvens/rekkefølge av tjenester og tjenestenes funksjoner.

9.9 Modellering av restriksjoner

/rec/computationalviewpoint/ pre_and_post_conditions	Restriksjoner/betingelser knyttet til tilstander før og etter bruk av tjenesten kan uttrykkes med OCL.

9.10 Kategori av tjenester

/req/servicetaxonomy/ service type – architecture	En tjeneste skal klassifiseres til å tilhøre en eller flere (for aggregerte tjenester) tjeneste arkitektur typer for geografiske tjenester. • human/boundary interaction • model/information management • workflow/task management • processing (spatial, thematic or temporal) • Communication • management/security

Dette siste kravet er også i henhold til COMMISSION REGULATION (EC) No 1205/2008 of 3 December 2008, implementing Directive 2007/2/EC of the European Parliament and of the Council as regards metadata (clause D.4).

10 Generelt om UML-modellering av datastrukturer

Dette kapittelet beskriver i hovedsak regler og anbefalinger fra ISO 19103:2015 Conceptual schema language. Nasjonale regler og anbefalinger har navnemønster /krav/nnn.

10.1 Hvordan forstå en UML-modell for geografisk informasjon

UML er et grafisk språk for objektorientert modellering.
UML er forkortelse for Unified Modeling Language.
UML utvikles av Object Management Group (www.omg.org)
UML versjon 2.4.1 er en internasjonal standard med navn ISO 19505:2012.

Det grafiske språket består av diagrammer med faste grafiske modellelementer. I tillegg må en i modelleringsverktøy lagre tekstlige definisjoner på modellelementene. Dersom det grafiske språket ikke er tilstrekkelig kan en der også legge inn nøkkelord og restriksjoner i egne presise tekstlige språk.

UML har mange ulike typer diagrammer, for å beskrive informasjon, oppførsel, brukstilfeller og komponenter.

Geografisk informasjon beskrives hovedsakelig med 3 diagrammer for statisk struktur:

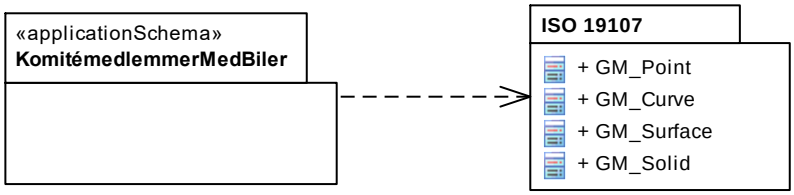
- **Pakkediagram** viser pakker og forhold mellom disse.
- **Klassediagram** viser klasser og assosiasjoner mellom disse. Klassediagram viser også klassenes navn og egenskaper.
- **Objektdiagram** kan vise reelle instanser som beskriver den virkelige verden.

Diagrammene viser ulike typer modellelementer grafisk.

Alle modellelementer i geografisk informasjon skal ha forståelige navn.

Alle modellelementer i geografisk informasjon skal ha tilstrekkelig definisjon.

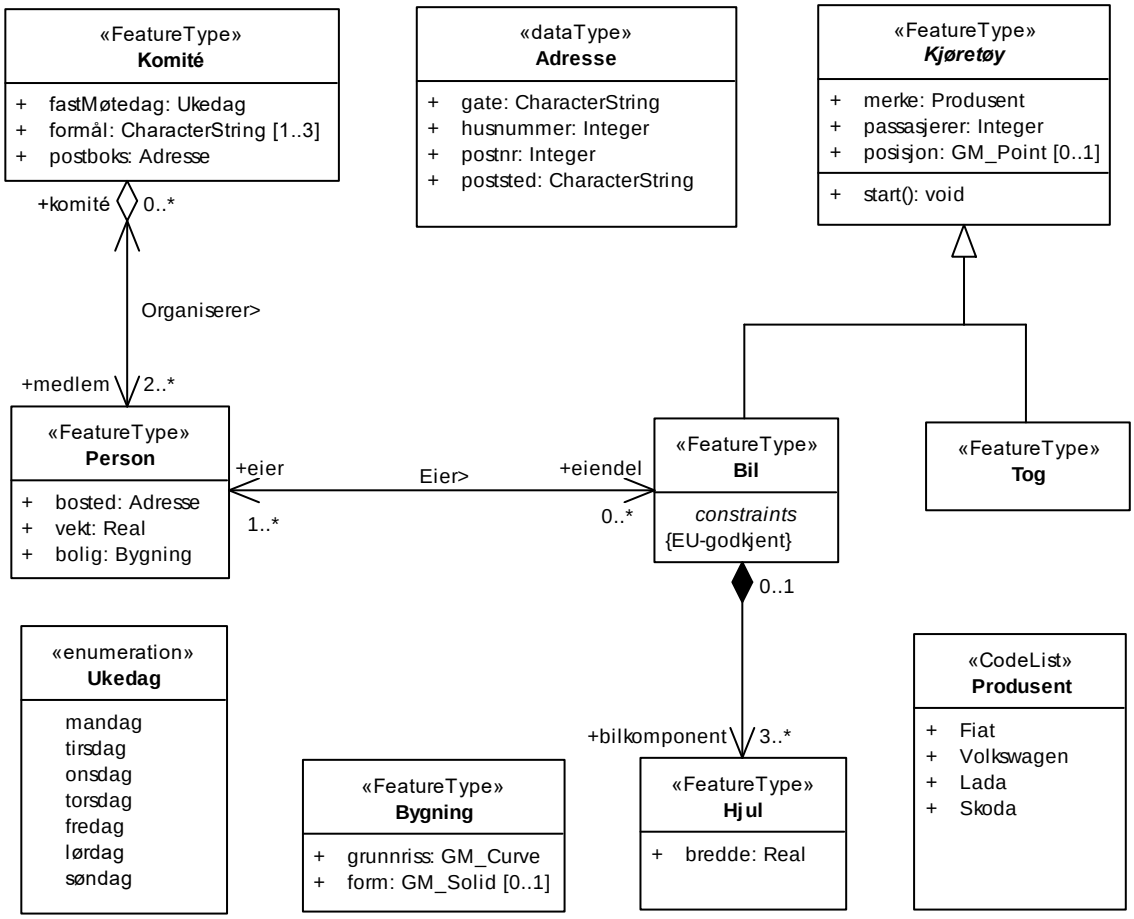
10.1.1 De viktigste modellelementene i pakkediagram



Figur 10.1 De viktigste modellelementene i pakkediagram.

Pakker med stereotypen «ApplicationSchema» inneholder objekttyper.
Pakkeavhengighet viser at noen klasser i en pakke trenger klasser fra en annen pakke.

10.1.2 De viktigste modellelementene i klassediagram



Figur 10.2 – De viktigste modellelementene i klassediagram

Figur 10.2 viser de viktigste modellelementene i klassediagram.

Klasser vises som firkantede bokser med felt for navn, egenskapsliste, og operasjoner.
Noen klasser kan også ha felt for navnede restriksjoner(*Constraints*) og tagged values.

Rett foran navnet på klassen kan det stå et stereotypenavn, med «» rundt.
(Stereotypenavn er ikke case-sensitive, så «featureType» er likeverdig med «FeatureType».)

Klasser med stereotypen «interface» eller «type» er konseptuelle klasser.
Disse kan ikke brukes direkte i datasett, men må realiseres i instansierbare klasser.

Klasser med stereotypen «FeatureType» er geografiske objekttyper.
Disse kan ha egenskaper med geometritype, eller ha andre geografiske tilknytninger.

Klasser med stereotypen «dataType» er identitetsløse samlinger av egenskaper.
Disse kan kun oppstå som egenskapstyper eller komponenter i andre klasser.

Klasser med stereotypen «Union» inneholder liste med typer hvorav kun én er mulig.

Klasser med stereotypen «enumeration» er lukkede lister av navnede koder.

Klasser med stereotypen «CodeList» er utvidbare lister av navnede koder.

Assosiasjoner mellom klasser vises som streker mellom de firkantede boksene.
De kan ha egne assosiasjonsnavn, og kan angi hvilken retning navnet indikerer.

Assosiasjonsender med pil betyr at assosiasjonen er navigerbar i pilens retningen.

Rollenavn og multiplisitet ([min..maks]) skal stå ved alle navigerbare ender.

Assosiasjoner med åpen diamant viser at dette er en samling av selvstendige deler.
Dette kalles aggregering i UML.

Assosiasjoner med fylt diamant viser at instanser av klassen på diamantsiden eier komponentene sine. Dette kalles komposisjon i UML.

Arv mellom klasser vises som streker med åpen trekant mot klassen det arves fra.

Abstrakte klasser har navnet i kursiv. Disse klassene kan ikke instansieres.

Noter er firkantede bokser med et nedbrettet hjørne. Der kan en vise en eller annen fri tekst inni. (Eldre modeller viste ofte restriksjonsbeskrivelser i noter.)

Noter kan kobles til modellelementer via en stiplet strek.

Klasser inneholder en boks med liste over egenskaper. Disse er normalt beskrevet på formen:
+ egenskapsnavn:Egenskapstype [min..maks].

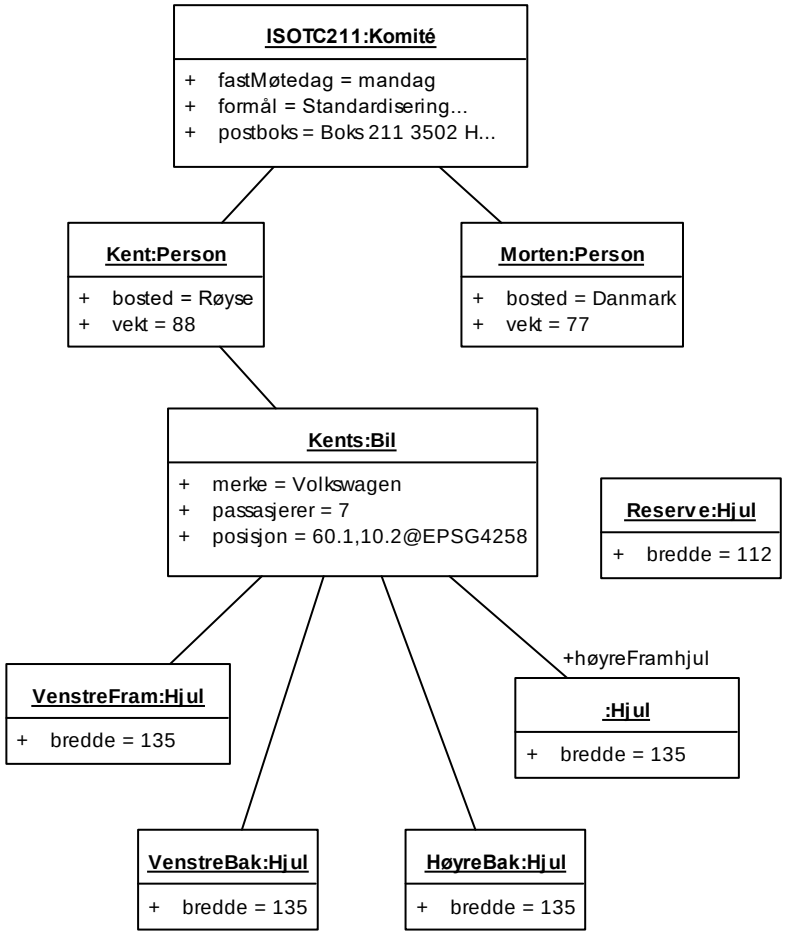
Tegnet + foran egenskapsnavn og rollenavn betyr at det ikke er kun et internt navn.
Tegnet / foran egenskapsnavnet betyr at egenskapsverdien avledes fra andre verdier.

Klasser kan også inneholde en boks med liste over klassens operasjoner.

Klasser kan også inneholde en boks med liste over klassens restriksjoner, denne boksen er merket "*constraints*".

Klasser kan også inneholde en boks med liste over klassens tagged values, denne boksen er merket "*tags*".

10.1.3 De viktigste modellelementene i objektdiagram



Figur 10.3 – Eksempel på de viktigste elementene i objektdiagram

Objekter er firkantede bokser med felt for egennavn:klassenavn med understreking under, og med felt for liste med egenskapsnavn=verdi. Du og jeg er således to objektinstanser av klassen Person, vi har ulike egennavn, og vi har sannsynligvis ulik vekt.

Link mellom objekter viser ulike reelle relasjoner mellom objektene.
Vi kan for eksempel være medlemmer av samme komité, mens bare jeg har bil. Noen hjul kan enten være løse eller de kan sitte fast på kun én bil.

10.2 Generelle krav og anbefalinger for modellering med UML

For geografisk informasjon generelt er det etablert et sett med krav og anbefalinger, og en utvidelse av UMLs modellelementtyper.

Krav og anbefalinger i dette kapitlet som har opphav i ISO 19103 er merket som nummerert. En unik navning av disse kravene etableres ved å sette et URI-navnerom foran. Eksempel: http://skjema.geonorge.no/SOSI/UML-modellering/5.1/krav/1.

Krav og anbefalinger fra ISO 19109 har biter av en slik URI-struktur i navnet. Eksempel: http://skjema.geonorge.no/SOSI/UML-modellering/5.1/req/multi-lingual/package.

Krav og anbefalinger som er spesielle norske innstramminger har denne strukturen: http://skjema.geonorge.no/SOSI/UML-modellering/5.1/krav/flerspråklighet/pakke

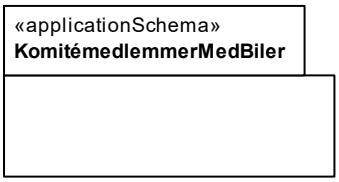
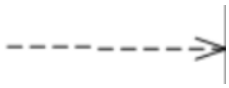
/krav/1	Det konseptuelle UML-skjema skal modelleres konformt med UML 2.

/krav/3	Alle modeller skal ha forståelige definisjoner av alle klasser, egenskaper, navigerbare assosiasjonsroller, operasjoner og datatyper.

/krav/definisjoner	Definisjonen av alle pakker, klasser, egenskaper, roller, operasjoner og restriksjoner skal registreres i modelleringsverktøyets primære dokumentasjonsfelt, dersom dette kan eksporteres. Sekundære beskrivelser og informative noter kan registreres i en tagged value med navn "description". Hvis modellelementet er hentet fra en eksisterende objektkatalog skal denne refereres til i en tagged value med navn "catalogue-entry". Fra ISO 19109:2015 /req/uml/documentation

10.2.1 Krav til pakker

Tabell 10.1 Pakker i UML.

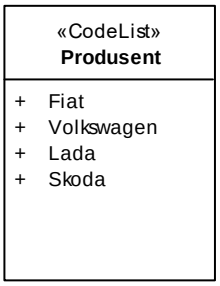
Pakke		Symbolet for pakke (Package) vises som en sammensetning av to rektangler, med pakkenavn og eventuelt en stereotype. Symbolet kan også indikere hva pakka inneholder.
Avhengighet		Avhengighet mellom pakker vises ved en stiplet linje mellom aktuelle pakke, med pil mot den pakka man er avhengig av.

Alle modeller med stereotype ApplicationSchema på en pakke har angitt sitt abstraksjonsnivå. Disse pakkene er ment å være komplette og implementerbare på ulike plattformer (MDA). Pakker over og pakker under disse kan lages for bedre gruppering, og er da også på samme abstraksjonsnivå men de har ikke stereotype. Pakker med plattformuavhengige tjenestemodeller kan også være nivå "Application Schema level". Pakker med prosessdiagrammer og brukstilfeller er normalt på nivå "Abstract Schema level" og dette må da dokumenteres i pakkebeskrivelsen.

/krav/4	Alle modeller skal dokumentere hvilket abstraksjonsnivå de er på:	
	Metamodel level	GFM og metamodellen i UML
	Abstract Schema level	Komponentene fra iso-standardene.
	Application Schema level	UML-applikasjonsskjema
	Implementation Schema level	GML-skjemaspesifikke modeller

10.2.2 Krav til kodelister

Tabell 10.2 Kodelister

lukket kodeliste		Symbolet for en lukket kodeliste er en klasse merket «enumeration» der innholdet er en liste med de konsepter som er gruppert sammen.
åpen kodeliste	 Alternativt med visning av tagger 	Symbolet for en åpen kodeliste er en klasse merket «CodeList» der innholdet er en liste med de konsepter som er gruppert sammen. Det kan legges inn nye koder i åpne kodelister men disse bør ikke overlappe med eksisterende. En tagged value med navn codeList bør inneholde en plattformuavhengig URI som identifiserer kodelisten. En tagged value med navn asDictionary og verdi true angir at kodelista er eksternt forvaltet av en ansvarlig organisasjon, og at kodene ikke ligger som en del av skjema. NB: kan ha en tagged value med navn codeList som inneholder en URL direkte til en GML Dictionary der koder og definisjoner kan hentes fra. Andre plattformspesifikke tagged values kan også forekomme.

Innholdet i lukkede kodelister forandres ikke under modellens levetid. Innholdet i åpne kodelister kan endres uavhengig av modellens versjon, men klare regler for hvordan kodelistekoder skal kunne endres må gis av ansvarlig etat. Kodelisten kan inneholde en tagged value med navn asDictionary og verdi true for å angi at den ikke kan utvides av brukeren selv. Filer med eksternt forvaltede kodelister kan vanligvis genereres automatisk ut fra et koderegister. Se også anbefaling/4.

/krav/5	Egenskaper av lukkede kodelister som verdirom skal kun bruke en av de beskrevne kodene fra den lukkede kodelista.

/krav/6	Koder modelleres i egne klasser der egenskaper representerer kodene. Navn på koder skal være mnemoniske (forståelige/huskbare), følge navnereglene for egenskapsnavn og de skal være uten skilletegn og spesialtegn.

	Unntak: Dersom kodens navn er et egennavn eller en godt kjent forkortelse kan navnet likevel starte med stor bokstav eller siffer.

Som en nasjonal tilpassing er de to følgende krav en lemping av /krav/6 for modeller som ikke er istand til å følge dette kravet. Se detaljert beskrivelse av forholdet i Vedlegg A.

/krav/kodenavn	Koder modelleres i egne klasser der egenskaper representerer kodene. Navn på koder skal være mnemoniske (forståelige/huskbare), følge navnereglene for egenskapsnavn og de skal være uten skilletegn og spesialtegn. Unntak: Dersom kodens navn er et egennavn eller en godt kjent forkortelse kan navnet likevel starte med stor bokstav eller siffer. Dersom koder har en initialverdi/utvekslingsalias skal man kunne gjøre dokumenterte avvik fra krav om fravær av blanke og skilletegn i kodens navn.

/krav/kodebruk	Dersom koder har en initialverdi/utvekslingsalias skal denne være uten skilletegn og spesialtegn, og den skal benyttes i alle realiseringer. Dersom en kode i en kodeliste har initialverdi/utvekslingsalias så skal alle andre koder i lista også ha slike.

Skilletegn og spesialtegn som må unngås er: blank, komma, !, ", #, $, %, &, ', (,), *, +, /, :, ;, <, =, >, ?, @, [, \,], ^, `, {, |, }, ~

Et modellelementnavn kan normalt heller ikke starte med siffer, "-" eller "." (NCName) dersom det problemfritt skal kunne realiseres i ulike plattformer.

Alle norske og samiske tegn er således tillatt i alle modellelementnavn.

For ytterligere beskrivelse av bruk av tegn i navning (NCName) se:
http://www.w3.org/TR/REC-xml/#NT-NameStartChar og:
http://www.w3.org/TR/REC-xml-names/#NT-NCName

/krav/7	Alle koder er konsepter, og skal ha tilstrekkelig definisjon. Det vil si alle unntatt lister over kjente egennavn.

/krav/8	Stereotypen CodeList skal benyttes der ingen eller kun et fåtall av kodene er kjent på modellversjonens tidspunkt.

/anbefaling/1	Det anbefales å ikke bruke meningsløse initialverdier som alternativ til kodens navn.

I brukstilfeller, slik som det å kunne skrive koder rett på kart, er det viktig å være bevisst på at korte kodenavn eller initialverdier/utvekslingsalias kan gi mening ut ifra hvilke objekter de er plassert på.

Eksempler på bruk av initialverdier i koder:
europaveg = E
riksveg = R

Eksempler på meningsløse initialverdier i koder:

europaveg = 1
riksveg = 2

For å ta vare på tallkoder som tidligere er brukt i SOSI-formatet kan disse initialverdiene flyttes til en tagged value med navn SOSI_verdi. Disse vil da kunne vises som tidligere i SOSI-formatrealiseringer. Mappingen mellom koder og SOSI_verdi må gjøres tilgjengelig for konvertering av eldre data.

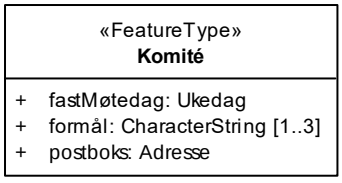
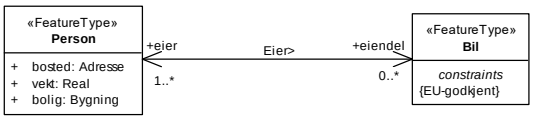
/anbefaling/2	For bedre forståelse anbefales det å ha ytterligere navn på konsepter i klart språk, og ha med definisjoner i ett eller flere språk, enten i tagged values eller knyttet til et register utenfor modellen.

/anbefaling/3	Utvidelser til kodelister bør ikke erstatte eller forandre eksisterende koder.

/anbefaling/4	En tagged value «codeList» bør inneholde en URI som identifiserer kodelisten.

10.2.3 Egenskaper og krav til og assosiasjonsroller

Tabell 10.3 Egenskaper og assosiasjonsroller.

egenskap		Symbolet for en egenskap er et navn med : etter i en navneliste i en klasse. En + foran indikerer bare at egenskapen er åpent tilgjengelig. Etter : angis navnet på datatypen til egenskapen, og til slutt kan en multiplisitet angis i [].
rolle		Symbolet for en rolle er en assosiasjonsende med et rollenavn. Assosiasjoner (i eldre modeller) helt uten navigerbarhetspiler ble tolket slik at de er navigerbare i retninger der det er et rollenavn. Selve assosiasjonen kan også ha et navn men dette er ikke viktig å ha med.

/krav/10	Alle assosiasjoner skal ha multiplisitet på navigerbare ender (med pil).

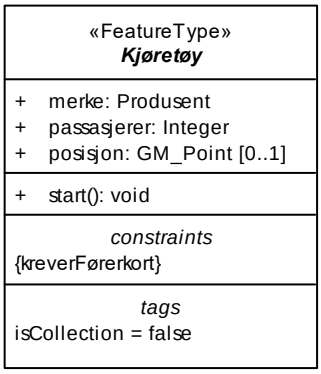
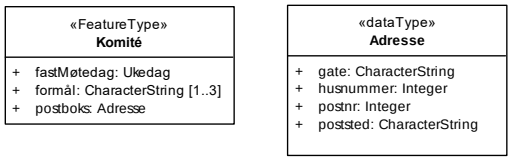
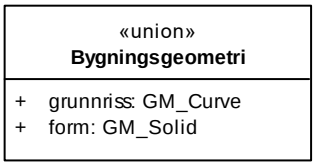
/krav/11	Alle navigerbare ender skal ha rollenavn.

/anbefaling/5	Rollenavnet bør reflektere den rollen som klassen på rollenavnenden spiller for den andre klassen.

/anbefaling/6	Assosiasjoner mellom flere enn to klasser bør unngås for å redusere kompleksitet.

10.2.4 Krav til klasser

Tabell 10.4 Klasser

objekttype		Symbolet for en objekttype er en boks med en eller flere vertikale deler, og et forståelig klassenavn i den øverste del av klasseboksen. En «stereotype» foran klassenavnet indikerer at klassen har denne spesielle betydningen. Rett under navnedelen kommer en del med egenskaper. Hvis klassen har operasjoner er de i en egen del under egenskapene. Restriksjoner og tagger kan vises i egne deler underst.
datatype		Symbolet for en datatype er en boks med en eller flere deler, og et navn i den øverste del av klasseboksen. Det skal stå «dataType» foran navnet. Datatyper skal ikke ha egen identitet. Datatyper kan ha egenskaper og assosiasjoner.
union		En klasse med stereotype «Union» beskriver et sett med mulige klasser. Kun en av klassene kan forekomme i hver instans.

/krav/12	Datatypeklasser med stereotype «dataType» mangler egen identitet og skal kun benyttes til egenskaper eller som mål i navigerbare komposisjoner.

/krav/enkelArv	Bruk av multippel arv skal ikke forekomme dersom modellen skal benyttes til implementering.

/krav/14	Generalisering er kun tillatt mellom klasser med samme stereotype og i samme abstraksjonsnivå.

/krav/15	Modeller av geografisk informasjon skal ved behov bruke en av de standardiserte stereotypene, og ikke lage egne alternative stereotyper med samme mening. a) «CodeList» åpen kodeliste som benytter tekster til å representere de potensielle verdiene. b) «dataType» sett av egenskaper som ikke har noen egen identifikator c) «enumeration» lukket kodeliste med navnede koder d) «interface» abstrakt klassifikator som kan inneholde egenskaper, assosiasjoner og operasjoner, vanlige klasser kan

	realisere dette innholdet e) «Leaf» pakke som inneholder definisjoner, og som er uten underpakker f) «Union» type som inneholder kun en av et sett mulige alternativer g) «FeatureType» klasse som er objekttype h) «ApplicationSchema» Pakke som er applikasjonsskjema (Andre stereotyper med andre betydninger kan legges til.)

/anbefaling/ styleGuide	Det anbefales å følge UML 2 style guide ved bruk av store og små bokstaver på stereotyper. Se skrivemåte i tabell over. Annen bruk av store og små bokstaver er også tillatt.

For beskrivelse av navning av stereotyper se: www.omg.org sitt dokument "UML 2.4.1 Superstructure 11-08-05.pdf" side 681 (se også side 54, 680 og 686).

10.2.5 Krav til navning og tekstlig dokumentasjon av modellelementer

Tabell 10.5 Klassers navn og definisjon.

objekttype og egenskaper med forståelige navn og tekstlig beskrivelse	«FeatureType» **Komité** + fastMøtedag: Ukedag + formål: CharacterString [1..3] + postboks: Adresse **Feature Type: Komité** Komité Definition: en gruppe personer som er oppnevnt for et spesielt formål Type: Feature Type Attribute: Name: formål Definition: hvilke oppgave komiteen skal løse Voidable: false Multiplicity: 1..3 Value type: CharacterString back to package: HjulKomité	Objekttypens definisjon, og egenskapenes, rollenes og restriksjonenes definisjoner må kunne leses i sammenheng med diagrammet der objekttypen er vist.

Alle modellelementer skal ha et navn som kan forstås og brukes av både mennesker og maskiner. De fleste modellelementer har en definisjon, men denne vises ikke i noe diagram. Derfor må det i tillegg genereres tekstlige utlistinger som står sammen med diagrammene.

/krav/16	Alle navn på modellelementer skal være case-insensitivt unike innenfor sitt navnerom, og ikke inneholde blanke eller andre skilletegn. Merknad: navnerommet til roller og egenskaper er klassen.

/anbefaling/8	Navn på alle modellelementer bør være presise og lett forståelige tekniske navn.

/anbefaling/9	Navn på modellelementer bør være korte der eierklassen også tilfører forståelse. Eksempel. Hvis en objekttype har navnet Bygning, bør ikke egenskapen hete bygningsnavn, men kun navn. Godt etablerte navn (bygningsnummer) bør kunne videreføres.

/anbefaling/10	Navn som kombinerer flere forståelige ord bør unngå skilletegn som "_" og "-".

/krav/navning	Navn på egenskaper, roller, operasjoner og parametere skal starte med liten bokstav, og så fortsette med stor bokstav på hvert nytt ord (med unntak av konstruktøroperasjoner som har samme navn som klassen). Navn på pakker, klasser og assosiasjoner skal starte med stor bokstav. (Fra ISO 19103:2015 Anbefaling 11, se også /krav/6)

/anbefaling/12	Modellelementer bør bruke dokumentasjonsfelt for å ytterligere klargjøre meningen med elementet.

/anbefaling/13	Navn på modellelementer bør være så korte som praktisk mulig. Bruk standard forkortelser hvis de er forståelige, man bør hoppe over preposisjoner og verb når de ikke tilfører vesentlig mening til navnet.

For å støtte forståelsen av hva objekttyper og andre modellelementer er, kan det lages en peker til et bilde i form av en tegning eller et foto som eksemplifiserer elementet. Denne pekeren kan være en http-URI og som peker til en fil tilgjengelig på nett. Ved generering av dokumentasjon kan denne fila da hentes og vises automatisk.

Eksempel:
http://skjema.geonorge.no/SOSI/fagområdestandard/Gravplass/4.5/gravsted.png

/anbefaling/ bildeAvModellelement	Alle modellelementer kan ha en tagged value med navn SOSI_bildeAvModellelement og med verdi i form av en URI som identifiserer et bilde som illustrerer modellelementet.

10.2.6 Krav til struktur for flerspråkelighet

Tabell 10.6 Flerspråkelighet i modellelementer.

pakker, objekttyper og andre modellelementer med navn og beskrivelse i flere språk	«FeatureType**Komité** + fastMøtedag: Ukedag + formål: CharacterString [1..3] + postboks: Adresse *tags* definition = "group of appointed persons"@en definition = "group de gens sympas"@fr description = "bør inneholde begge kjønn"@no designation = "Committee"@en	Dersom modellelementers navn, definisjon og ytterligere beskrivelser skal beskrives i alternative språk så skal dette gjøres med spesielle tagged values og med spesiell formatering av taggenes innhold.

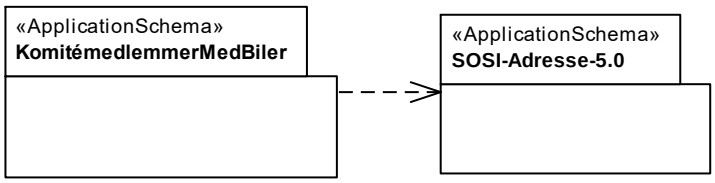
Disse kravene til taggnavn og formatering for flerspråkelighet kommer opprinnelig fra krav til pakker i ISO 19109 men er i SOSI også påkrevet for alle andre modellelementer.

/krav/flerspråklighet/ pakke	En applikasjonsskjemapakke skal ha en tagged value "language" med verdi som indikerer primærspråket for alle navn og definisjoner i applikasjonsskjemaet. Verdien skal tas fra IETF RFC 5646. (Eks. language=no, language=en) Dersom en pakke har en tagged value "designation" som inneholder pakkenavnet i et alternativt språk skal verdien formateres etter følgende mønster: "{Name}"@{language} der {Name} er navnet i det angitte språk, og {language} er språkkoden fra ISO639 etter reglene i IETF RFC 5646. NB: Fnutter er påkrevet omkring navnet. Eksempel: "CommitteeMembersWithCars"@en fra ISO 19109:2015 /req/multi-lingual/package

/krav/flerspråklighet/ element	Dersom en klasse, egenskap, rolle, operasjon eller restriksjon skal beskrive navnet eller definisjonen i alternative språk skal disse benytte en tagged value "designation" eller "definition" for å ha definisjonen i språkene. Dersom ønskelig kan man også benytte en tagged value "description" for å ha noter og ytteligere beskrivelser i alternative språk, og denne skal da benytte samme format. Alle disse tagged values kan gjentas. Verdiene skal formateres etter følgende mønster: "{Name}"@{language} Eksempel: "Committee"@en fra ISO 19109:2015 /req/multi-lingual/feature

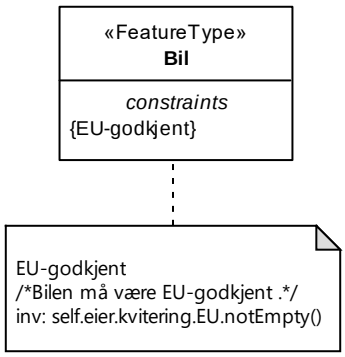
10.2.7 Krav til visning i diagrammer

Tabell 10.7 Pakkediagram

Pakkeavhengighet		Pakker med innhold som refererer til andre pakker (ikke underpakker) skal angi hvilke pakker den er avhengig av.

/krav/17	Dersom det er avhengigheter mellom applikasjonsskjemapakker skal alle disse avhengighetene vises i minst ett pakkediagram (ofte kalt pakkeavhengighetsdiagram).

Tabell 10.8 Diagram med restriksjon.

Restriksjoner kan ha forklarende navn og beskrives i klartekst, og i tillegg ha presis OCL-syntaks		Objekttypers restriksjoner kan navnes og vises i diagrammet slik at leseren intuitivt forstår meningen. Definisjonen av restriksjonen, og OCL-syntaksen vises fullstendig i den tekstlige delen. Dette kan også vises grafisk i diagrammet dersom det legges i en note.

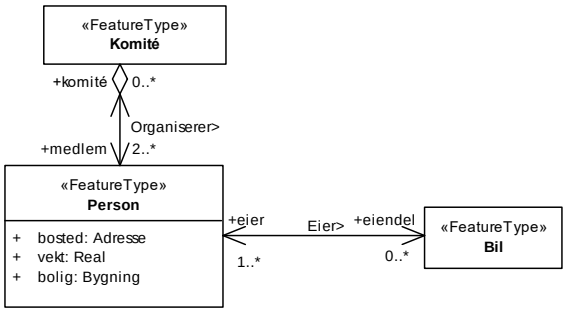
/anbefaling/ restriksjonsnavn	Restriksjoner bør ha selvforklarende navn, og navnene bør vises i diagram. I tillegg kan restriksjonens navn, definisjon og OCL-syntaks vises i en note.

/anbefaling/14	Restriksjoner bør beskrives med OCL 2.0. I tillegg bør de beskrives i klart språk.

/anbefaling/15	For å lette lesing av modeller bør fontstørrelsen på diagramtekster være lik eller større enn 8 punkt, etter at eventuell skalering er iberegnet.

/anbefaling/16	For hver større pakke anbefales det å lage et samlediagram med alle pakkens klasser, vist uten egenskapsinnhold.

Tabell 10.9 Diagram for samlet visning av innhold i klasser.

Alle klasser skal vise alt innhold samlet i minst et diagram		Alle objekttypens egne og arvede egenskaper og roller skal vises samlet i minst ett diagram. Det kreves ikke at datatyper og assosierte klasser viser sitt innhold. Ofte er et hoveddiagram tilstrekkelig for å vise alt innhold dersom en pakke kun inneholder noen få objekttyper.

/krav/18	Alle klasser og assosiasjoner skal i minst ett diagram vise alle arvede og alle egne egenskaper, roller, operasjoner og restriksjoner.

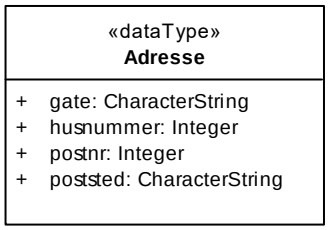
/krav/19	Alle klasser skal ha en definisjon som beskriver mening og forståelse.

/krav/20	Alle klasser, assosiasjoner, egenskaper, roller og restriksjoner skal ha tekstlig beskrivelse i tilknytning til sin grafiske visning. Dersom modellelementer inneholder viktige plattformnøytrale tagged values som SOSI_kortnavn, SOSI_presentasjonsnavn, SOSI_bildeAvModellelement eller asDictionary=true og codeList er dette også å regne som modellelementets dokumentasjon og må være med i den tekstlige beskrivelsen.

/krav/21	Pakkeavhengighetsdiagram skal vises med alle avhengigheter til eksterne pakker.

10.2.8 Basis datatyper som brukes

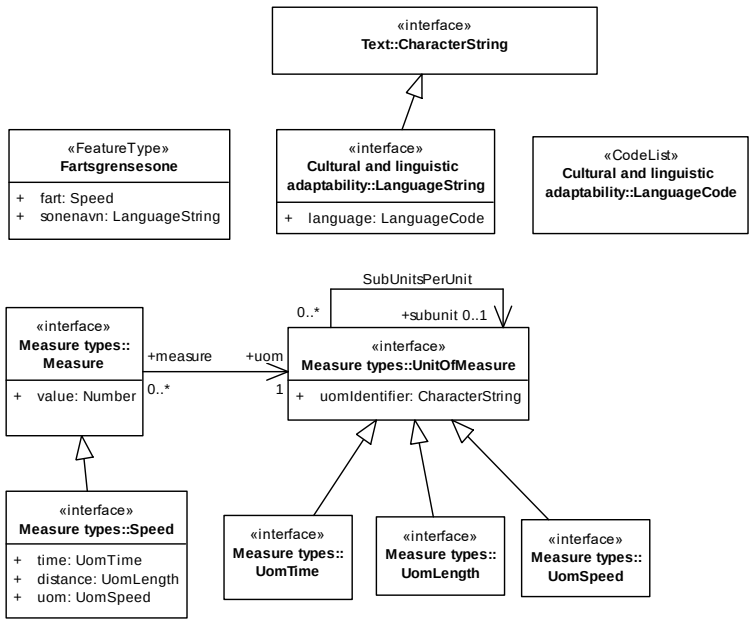
Tabell 10.10 Egenskaper

basistyper		Egenskaper har enten andre klasser som datatype, eller de kan bruke en standardisert datatype som har fast realisering i de aktuelle plattformene. De vanligst brukte basistypene er CharacterString, Integer, Real, Boolean, Date og DateTime.

/krav/22	Plattformuavhengige modeller skal benytte disse beskrevne basisdatatypene der dette er aktuelt, og ikke lage egne alternative typer med samme mening: a) Primitive typer: CharacterString, Integer, Real, Boolean, Date og DateTime,.. b) ”Collection templates”, (Set, Bag, OrderedSet, Sequence) c) Enumererte typer, (Bit, Digit, Sign) d) Navnetyper: (typer for representasjon av navnestrukturer: NameSpace, TypeName, MemberName) e) Any type, f) Record typer: typer for representasjon av generiske strukturer. (Disse basistypene er videre beskrevet i ISO 19103:2015)

10.2.9 Utvidete typer som brukes

Tabell 10.11 Utvidete typer

utvidede typer		Egenskaper kan også bruke en standardisert utvidet datatype dersom det er behov for det. Disse har også oftest en fast realisering i en plattform.

/krav/25	Plattformuavhengige modeller skal benytte de beskrevne utvidelsesdatatypene der dette er aktuelt, og ikke lage egne alternative typer med samme mening. • LanguageString - for tekststrenger der man må angi at strengen er i et spesifikt språk som ikke vil være det samme som hele datasettets språk. • Anchor, FileName, MediaType, URI, - datatyper for bruk til verdensveven. • En av subtypene til Measure eller DirectedMeasure kan brukes dersom man vil styre verdisettingen: (Length, Distance, Angle, ...) • En av subtypene til UnitOfMeasure kan benyttes dersom verdier i datasettets egenskaper skal kunne få lov til å ha ulike benevninger og enheter: UomLength, UomAngle, ... • Koder for standardenheter: StandardUnits (Disse typene er videre beskrevet i ISO 19103:2015)

10.2.10 Objektdiagram

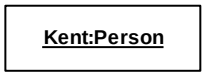
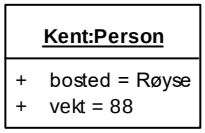
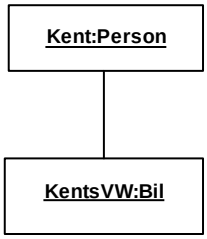
10.2.10.1 **Innledning**

Et objektdiagram er en instans av et klassediagram – en konkret forekomst av en klasse. Forskjellen er at et klassediagram representerer en abstrakt modell med klasser, egenskaper og relasjoner mellom klassene, mens et objektdiagram representerer et sett med instanser med aktuelle data. Mens elementene i klassediagrammet representerer malen, viser objektdiagrammene elementene med konkrete virkelige dataeksempler. Bruk av objektdiagrammer er vanligvis begrenset til å vise eksempler på datastrukturer. Et objektdiagram understøtter således forståelsen av et klassediagram.

10.2.10.2 **Symboler**

Objektdiagrammer lages med utgangspunkt i de samme basiselementene som for klassediagram, men på en noe annerledes form.

Tabell 10.12 Objektdiagram

Object		**Object** (Objekt) representerer instansen av en klasse, og identifiseres med en virkelig *ObjektId* og navnet på *Klassen* (Person) fra Klassediagrammet
Run Time State		**Run time State** (egenskapsverdier) ved et gitt tidspunkt angis for hver attributt som er definert i klassen
Link		**Link** (sammenhenger) mellom instanser av objekter vises med en strek

/anbefaling/ objektdiagram	Under modelleringsprosessen er det anbefalt å lage objektdiagrammer for å teste om mulighetene for instansieringen av klassene er bra nok. Når en modell er ferdigmodellert er det nyttig å lage og vise et eksemplarisk objektdiagram for å formidle hvordan man tenker seg at klassene skal instansieres.

/anbefaling/ instanseksempel	Under modelleringsprosessen, og spesielt når en modell er ferdigmodellert er det nyttig å lage eksempeldatasett med objektinstanser i et enda mer kjent format, for eksempel GML.

11 UML-modellering av applikasjonsskjema

Dette kapittel tar utgangspunkt i ISO 19109 :2015 Rules for application schemas, og beskriver regler for å modellere et UML-applikasjonsskjema. UML-applikasjonsskjema finnes i både SOSI fagområdestandarder og i SOSI produktspesifikasjoner. Dette kapittel har fokus på UML-applikasjonsskjema for produktspesifikasjoner, men har også noen regler som er spesielle for fagområdestandardenes UML-applikasjonsskjema, slik som abstrakte geometrityper for angivelse av stedfesting. Dette er omtalt i kapittel 11.5.

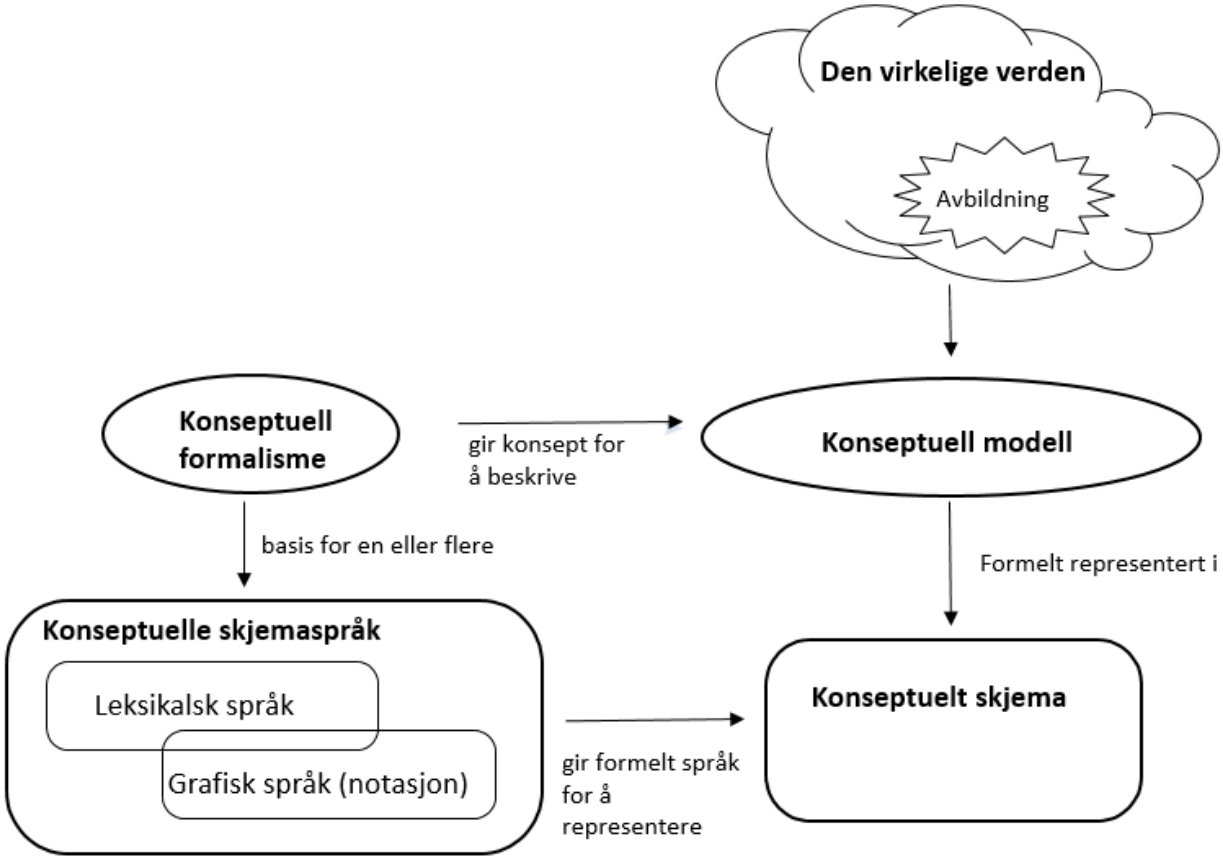
Kravene fra ISO begynner med "/req/", anbefalinger med "/rec/". Tilleggskrav og -anbefalinger begynner med hhv. "/krav/" og "/anbefaling/".

11.1 Modellering og konseptuelt modelleringsspråk

Målet med modellering av geografisk informasjon er å beskrive den virkelige verden med alle fenomener og innebygde regler på en mest mulig presis måte. Hvilke fenomener og hvilke regler fra den virkelige verden som er viktige å ta med inn i modellen, er svært avhengig av behovet brukerne av modellen har.

Ofte brukes uttrykket "Universe of discourse" (UoD) som betegnelse på den delen av virkeligheten som er av interesse for en bestemt bruk. En beskrivelse av UoD kalles en konseptuell modell. For å beskrive den konseptuelle modellen på en formell måte, trengs et konseptuelt modelleringsspråk. En konseptuell modell beskrevet med et formelt, konseptuelt modelleringsspråk kalles et konseptuelt skjema. Et skjema er altså en modell beskrevet med et formelt modelleringsspråk. I SOSI-arbeidet benyttes Unified Modelling Language (UML) som det formelle modelleringsspråket.

Det finnes en rekke modelleringsspråk i tillegg til UML, slik som OML (OPEN Modeling Language), EXPRESS (data modeling language for product data), etc. De ulike modelleringsspråk har sine styrker og svakheter, men selve avbildningen av den virkelige verden i form av modeller er felles for dem alle.



Figur 11.1 Fra virkelighet til konseptuelt skjema (ISO 19109).

UML har en metamodell (konseptuell formalisme) som beskriver en grafisk og tekstlig notasjon. Den grafiske notasjonen med tekstlig beskrivelse er beregnet for mennesker. Dette er måten vi formidler og kommuniserer konseptene på. En streng grafisk notasjon gjør det mulig å tolke en UML-figur på en veldig presis måte, og kan på ingen måte sammenlignes med f.eks. en PowerPoint-figur, hvor det ikke er noen grafisk notasjon.

Den tekstlige beskrivelse inneholder også deler av modellen som ikke vises i den grafiske notasjonen, f.eks. definisjonen av modellelementene. Det finnes ulike verktøy for å generere den tekstlige beskrivelsen av en modell. Det finnes også en standard tekstlig beskrivelse, XMI (XML metadata interchange) som gjør det mulig å utveksle modeller mellom ulike UML-verktøy. XMI er beregnet for å leses av datamaskiner.

Denne konseptuelle formalismen gjør det mulig å beskrive en avbildning av den virkelige verden i en konseptuell modell. Denne konseptuelle modellen er igjen utgangspunkt for et konseptuelt skjema, som igjen kan mappes ned til en implementasjonsplattform, f.eks. GML eller SOSI-syntaks.

11.2 Praktiske tilnærmingsmåter - fra fenomener i virkeligheten til modellelementer

For å gjøre UoD (Universe of discourse - den interessante delen av virkeligheten) håndterbar, deles den i UML-applikasjonsskjema opp i enkeltdeler. De fenomenene som finnes i virkeligheten grupperes sammen i objekter med sine objektegenskaper. Et objekt er et selvstendig identifiserbart fenomen, fenomen som en kan si "har sitt eget liv". Objekter eksisterer (i alle fall) i databaser som representasjoner av slike "identifiserbare fenomener".
For å skape system og orden i håndteringen av objekter, klassifiseres de i objekttyper. En objekttype er en samling av kjennetegn for objekter som har noe til felles. Kjennetegnene brukes til å klassifisere et objekt til rett objekttype, og også til nærmere å beskrive objektet ved et felles sett med objekttypeegenskaper.

Eksempel: Objekttypen Bygning er en samling av kjennetegn for bygninger i virkeligheten. Vi har alle en oppfatning om hva som er en bygning. Dette er noe vi har lært lenge før vi ble introdusert for modellering i geografisk informasjon, og som lett fører til at vi ikke bryr oss om å dokumentere kjennetegnene. Alle kjenner jo igjen en bygning når de ser en. En bygning beskrives nærmere ved objekttypeegenskapene byggeår, tiltenkt bruk, byggemateriale og stedfesting (som kan uttrykkes for eksempel ved et representasjonspunkt)

De ulike fagområdene har ofte fagområdespesifikke måter å dele inn virkeligheten på. Det betyr at det som i et fagområde er en naturlig objekttype, i et annet fagområde er kun en egenskap til en objekttype. Det kan også bety at samme fenomenet i virkeligheten kan klassifiseres som to ulike objekttyper.

Eksempel 1: Fartsgrense er i noen sammenhenger modellert som en egen objekttype med typisk utstrekning fra et fartsgrenseskilt til et annet. I andre sammenhenger er det en egenskap på objekttypen veglenke.

Eksempel 2: Fenomenet "Et hvitt hus med et tårn med en kraftig lyskilde på toppen, beliggende på en holme like ved innseilingen til en by", klassifiseres i en sammenheng som tilhørende objekttypen bygning, i en annen sammenheng som tilhørende objekttypen fyr.

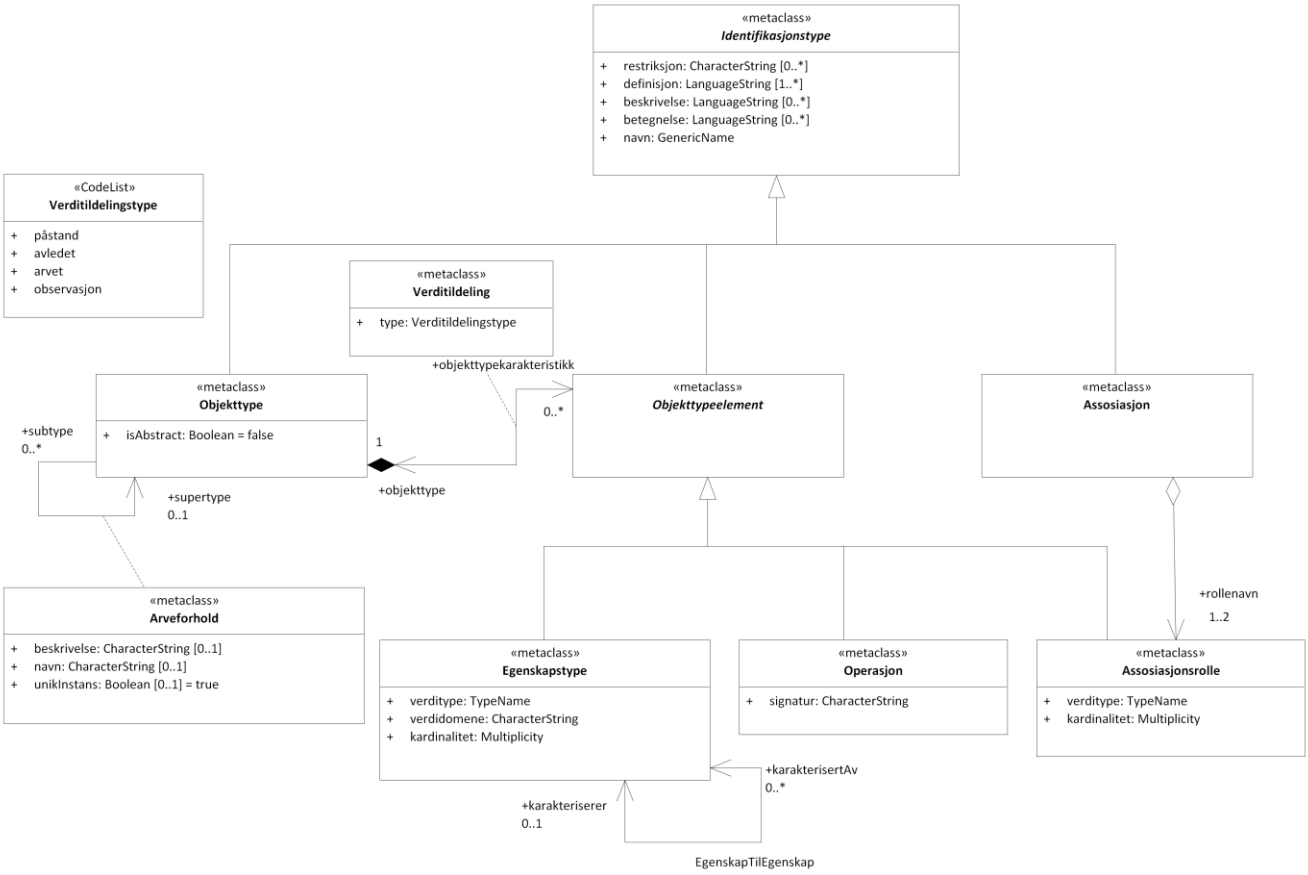
I geografisk informasjonsbehandling er det vanlig at objekter kan stedfestes. Men objekter uten direkte stedfesting er også nødvendige.

Eksempel: En eiendomsteig er stedfestet ved hjelp av eiendomsgrensene som skiller den fra nabo-teigene. Objekttypen matrikkelenhet er "kun" et juridisk fenomen uten egen naturlig stedfesting, men som stedfestes gjennom de tilhørende eiendomsteiger.

UML-reglene for å definere objekttyper er samlet i UML-metamodellen "*General Feature Model*". Den sier at objekttyper skal ha en definisjon, som gjør at vi kan kjenne igjen de objektene som tilhører objekttypen.

11.3 General Feature Model (GFM)

General Feature Model (GFM) er en modell av konsepter som er grunnleggende for å kunne beskrive et utsnitt av den reelle verdenen. GFM er en metamodell beskrevet i UML. Den definerer strukturen for hvordan vi skal klassifisere virkelighetens objekter, deres egenskaper og relasjoner mellom dem. Elementene som er nødvendig for denne klassifiseringen vises i Figur 11.2 som inneholder et subsett av GFM, SOSI-profilen. Denne profilen har noen strengere krav enn den som er beskrevet i ISO 19109 Rules for application schema.



Figur 11.2 SOSI-profil av GFM (ISO 19109 Rules for application schema).

Objekt (Feature) er den fundamentale enheten som brukes for å beskrive geografisk informasjon. Den del av virkeligheten som skal modelleres sees på som en samling objekter av ulike typer. GFM definerer konseptene som skal brukes for å beskrive objekttypene og hvordan konseptene henger sammen.

I kortform kan modellen forklares slik:

- **Objekttype**. Den del av virkeligheten som skal modelleres (f.eks. i et applikasjonsskjema), skal bestå av Objekter, definert med Objekttyper. Objekttyper kan videre være definert med supertyper/subtyper.
- **Objekttypeelement.** Et objekt defineres med sett av ulike egenskaper, operasjoner og relasjoner til andre objekter
    - **Egenskapstype** definerer en egenskap med tilhørende datatype. En egenskap kan være sammensatt av flere egenskaper
    - **Operasjon** definerer spesifikk funksjonalitet for objekttypen
    - **Assosiasjonsrolle** definerer forholdet til andre objekttyper
- **Assosiasjon** representerer relasjonen mellom objekttyper, og er bærer av en eller to Assosiasjonsroller
- **Identifikasjonstype** som er supertype til alle modellkonseptene, definerer hvordan objektene gis entydig identifikasjon og beskrivelse

Disse konseptene er grunnleggende for modellering av geografisk informasjon og uavhengig av modelleringsspråk. SOSI har valgt å anvende UML som modelleringsspråk.

11.4 Modellering av applikasjonsskjema

11.4.1 Generelle regler

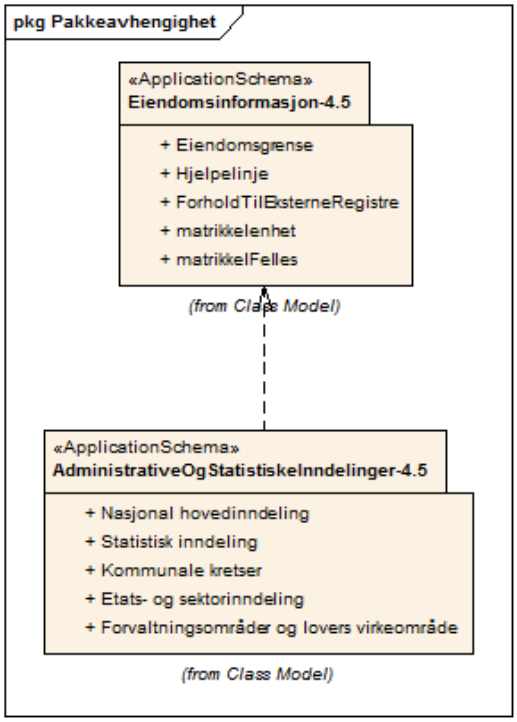
Et applikasjonsskjema er en modell av de objekttyper som er interessant for applikasjonens område.

Et applikasjonsskjema har en tosidig hensikt:

- Modellen skal vise felles og korrekt forståelse av innhold og datastruktur for det aktuelle applikasjonsområdet.
- Modellen skal uttrykkes presist i et konseptuelt skjemaspråk slik at en kan benytte automatiske metoder ved implementasjon av modellene.

/req/uml/profile	Applikasjonsskjema skal modelleres ved bruk av UML-profilen definert i ISO19103:2015, og med tillegg beskrevet i dette kapittel.
/req/uml/structure	Applikasjonens datastruktur skal modelleres i applikasjonsskjemaet. Alle klasser i et applikasjonsskjema for datautveksling skal enten være instansierbare, eller dersom de er merket som abstrakte skal de ha instansierbare subtyper. Ingen klasser i applikasjonsskjemaet skal ha stereotype «interface».
/req/general/csl	Et applikasjonsskjema skal beskrives i et konseptuelt skjemaspråk. For SOSI applikasjonsskjemaer skal det konseptuelle skjemaspråket være UML 2.
/req/uml/packaging	Applikasjonsskjema skal beskrives innenfor en pakke med stereotype «ApplicationSchema». Denne pakkens navn er navnet på applikasjonsskjemaet. Applikasjonsskjemaets versjon skal registreres i en taggedValue med navn "version".
/req/uml/documentation	Applikasjonsskjema skal dokumenteres i henhold til /krav/definisjoner i kapittel 10.2.
/req/general/integration	Applikasjonsskjema som bygger på eksisterende objektkataloger skal dokumentere disse avhengighetene eksplisit i modellen.
/req/uml/integration	UML pakkeavhengighet skal beskrive avhengigheter til andre pakker, slik at alle elementer i applikasjonsskjemaet blir komplett beskrevet. Pakker med stereotype «ApplicationSchema» skal ikke inneholde andre pakker med stereotype «ApplicationSchema». Pakkeavhengigheter skal kun være mellom applikasjonsskjemapakker, eller også til standard pakker. Pakker inne i, eller utenpå applikasjonsskjemapakker skal ikke ha pakkeavhengigheter. Pakker skal ikke ha gjensidige avhengigheter, og ikke inneholde sykliske avhengigheter. Avhengigheter (stipla strek med pil) til andre pakker kan ha stereotype «import».
/req/general/integration	Applikasjonsskjema som bygger på eksisterende objektkataloger skal dokumentere disse avhengighetene eksplisit i modellen.

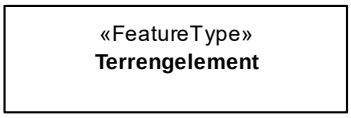
Tabell 11.1 Kopling mot andre pakker.

Knytning mot eksisterende objektkataloger		AdministrativeOgStatistiske Inndelinger benytter elementer fra Eiendomsinformasjon. Tilhørighet er merket med fulle tagger (Fully qualified Namespace) der dette er naturlig ut fra lesbarhet i modellen.

11.4.2 Hovedregler for å implementere GFMs konsepter i UML-applikasjonsskjema

11.4.2.1 Objekttyper

Tabell 11.2 Objekttyper

Objekttype		Her ser vi et eksempel på en objekttype, en klasse med stereotype «FeatureType». Denne objekttypen har fått navnet Terrengelement.

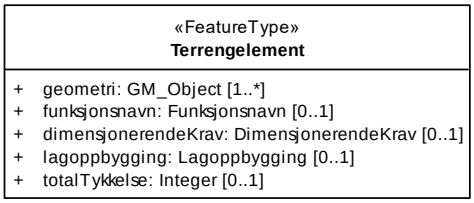
/req/general/feature	Objekttyper skal modelleres som instanser av metaklassen Objekttype. Objekttyper skal ikke modelleres som spesialiseringer av GM_Object eller TM_Object. Objekttypene skal ha navn som er unike innen applikasjonsskjemaet.
/req/uml/feature	Instanser av metaklassen Objekttype (jfr Figur 11.2 GFM) skal implementeres som klasser. Klassene skal ha stereotypen «FeatureType». Alle klassenavn skal være unike innen applikasjonsskjemaet. Den tekstlige definisjonen av klassen skal registreres i modelleringsverktøyets dokumentasjonsfelt dersom denne informasjonen kan eksporteres.

/rec/uml/property-name	Navnet på egenskaper, operasjoner og assosiasjonsroller bør være unike innenfor applikasjonsskjemaet, eller alle med like navn bør ha identiske definisjoner.

11.4.2.2 Egenskaper

Egenskaper karakteriserer klasser og bidrar til å skille ulike konsepter som er representert i ulike klasser fra hverandre.

Tabell 11.3 Egenskaper

Egenskap		Objekttype Terrengelement har fem egenskaper (geometri, funksjonsnavn, dimensjonerendeKrav, lagoppbygging og totalTykkelse). Egenskapene består av de obligatoriske elementene navn (f.eks. "totalTykkelse"), type (f.eks. "Integer"), multiplisitet (f.eks. "[0..1]") og definisjon (vises ikke i eksempelet).

/req/general/attribute	Egenskaper skal modelleres som instanser av metaklassen Egenskapstype i henhold til Figur 11.2 (GFM).
/req/uml/attribute	En instans av en Egenskapstype skal implementeres som UML Egenskap.
/req/uml/attribute	En instans av (metaklassen) Egenskapstype (GFM) representeres som en egenskap, dersom den ikke er en egenskap knyttet til en annen egenskap [se /req/uml/attributeOfAttribute]. Egenskapen eller assosiasjonsrollen kan ha stereotype «estimated» hvis verdien er tilordnet ved bruk av en observasjonsprosedyre. [ISO 19156:2011].
/req/uml/attribute-of-attribute	Hvis en instans av Egenskapstype er kompleks, skal verditypen referere til en klasse med stereotype «dataType» som inneholder definisjon av de enkelte egenskaper.
/req/uml/attributeOfAttribute	En instans av (metaklassen) Egenskapstype som oppstår som rollen "karakterisertAv" i en "Egenskap til egenskap" assosiasjon skal instansieres som en klasse. Denne klassen skal enten være datatypen til en egenskap, eller pekes til fra en assosiasjonsrolle. Egenskapen som er egenskap til en annen egenskap skal lages ved å: steg 1: Opprett en ny klasse som skal representere den egenskapen som skal karakteriseres. Bruk egenskapsnavnet som klassenavn, men med stor forbokstav. Merk klassen med stereotype «dataType»; steg 2: Sett inn en egenskap i denne klassen for å representere den originale egenskapen, bruk dennes navn direkte; steg 3: Sett inn egenskaper i denne klassen for å karakterisere den originale egenskapen; steg 4: Bruk denne klassen som datatype til den originale datatypen, eller slett den originale egenskapen og opprett en assosiasjon til den nye klassen med egenskapsnavnet som rollenavn.

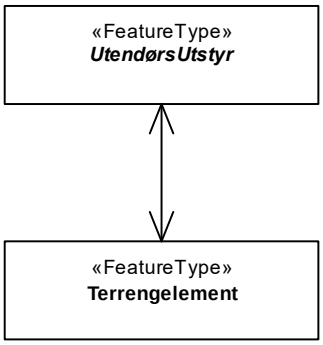
11.4.2.3 **Assosiasjoner**

Relasjoner mellom objekttyper modelleres ved hjelp av assosiasjoner. GFMs konsept **"**Assosiasjon» kan innta en av følgende subtyper:

- vanlig assosiasjon
- aggregering

- komposisjon
- topologisk
- temporal

Tabell 11.4 Assosiasjoner

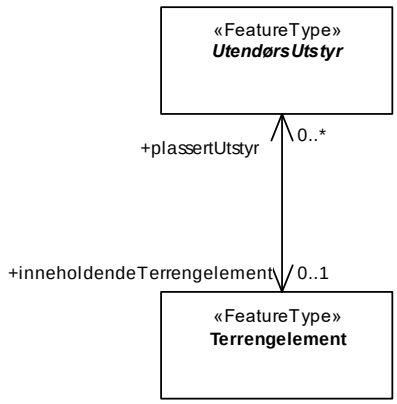
Assosiasjon		For å vise at det finnes en relasjon mellom objekttype Terrengelement og objekttype UtendørsUtstyr, ble det lagt til en vanlig assosiasjon mellom dem. Pilene viser at denne assosiasjonen er navigerbar mot begge endepunktene til assosiasjonen slik at begge objekttyper "vet" om relasjonen til den andre objekttypen. For flere eksempler og forklaring på aggregering og komposisjon se 10.1.2.

/req/uml/association	En instans av en Vanlig assosiasjon skal implementeres som en UML Assosiasjon mellom aktuelle objekttyper. Hvis assosiasjonen har egenskaper, skal disse defineres som egenskaper i en Assosiasjonsklasse.
/req/uml/aggregation	En instans av en assosiasjon av type aggregering (tilhører) skal implementeres som UML Aggregering (åpen diamant). Medlemmer kan eksistere uavhengig av aggregatet. En instans av en assosiasjon av type komposisjon (del av) skal implementeres som UML Komposisjon (fylt diamant).
/req/general/feature-association	Assosiasjoner skal modelleres som instanser av metaklassen Assosiasjon i henhold til Figur 11.2 (GFM).

11.4.2.4 **Assosiasjonsroller**

En assosiasjonsrolle sier noe om hvilken rolle en objekttype har overfor en annen objekttype.

Tabell 11.5 Assosiasjonsroller

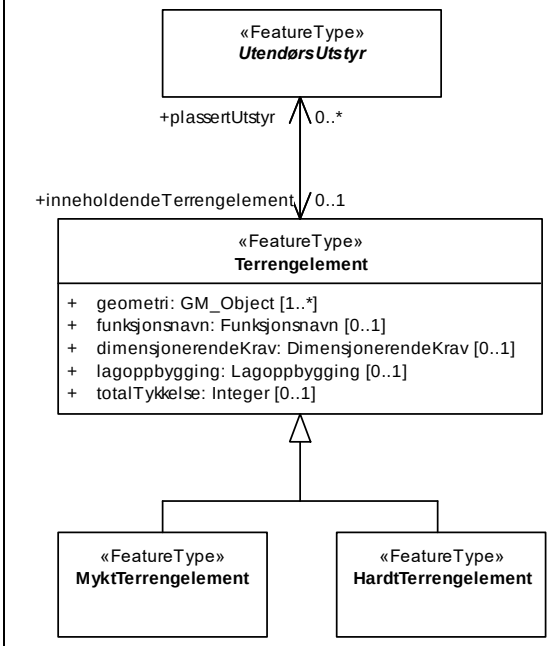
Assosiasjonsrolle		En assosiasjonsrolle består av elementene rollenavn og multiplisitet som begge legges til på den enden av assosiasjonen der det er pil. Eksempelet viser at flere instanser av objekttypen UtendørsUtstyr kan ha rollen plassertUtstyr i forbindelse med maksimal én instans av objekttypen Terrengelement. Terrengelementinstanser som er en del av denne relasjonen har rollen inneholdendeTerrengelement overfor utstyrselementer.

/req/uml/role	En instans av (metaklassen) Assosiasjonsrolle skal implementeres i henhold til GFM med et rollenavn ved korrekt ende av en assosiasjon som representerer en assosiasjon mellom objekttyper. Assosiasjonsrollen kan ha stereotype «estimated» hvis verdien er tilordnet ved bruk av en observasjonsprosedyre. [ISO 19156:2011].
/req/general/association-role	Assosiasjonsroller skal modelleres som instanser av metaklassen Assosiasjonsrolle i henhold til Figur 11.2 (GFM).

11.4.2.5 **Arv (generalisering/spesialisering)**

Arveforholdet representerer spesialisering og generalisering, og er et viktig konsept i objektorientert arkitektur. Egenskaper, operasjoner, assosiasjoner og restriksjoner blir overført fra supertypen (klassen på et generalisert nivå) til subtypen (klassen på et mer spesialisert nivå).

Tabell 11.6 Arv

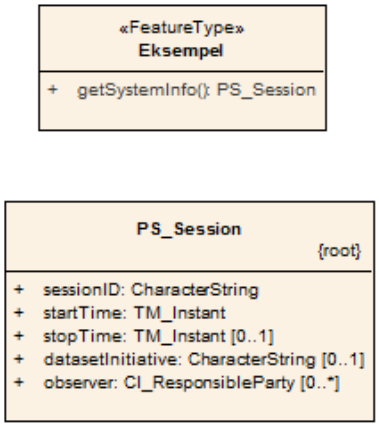
Arv		Objekttypene MyktTerrengelement og HardtTerrengelement arver egenskapene fra klassen Terrengelement, samt assosiasjonen til objekttype UtendørsUtstyr.

/req/general/inheritance	Arv skal modelleres som instanser av metaklassen Arveforhold.
/req/uml/inheritance	En instans av (metaklassen) "Arveforhold" skal representeres som en generalisering, med følgende mulige tillegg: tilfelle 1: Hvis *unikinstans* er *true* (default verdi) kan {disjoint} constraint knyttes til arverelasjonen; tilfelle 2: Hvis *unikinstans* er *false* skal {overlapping} constraint knyttes til arverelasjonen. Se også Figur 11.2 GFM.

11.4.2.6 **Operasjoner**

Oppførselen av objekttyper beskrives ved hjelp av operasjoner som kan utføres på eller av instanser av objekttypen der operasjonen er definert.

Tabell 11.7 Operasjoner

Operasjon		Objekttypen Eksempel har en operasjon Navn: getSystemInfo. Parametere: Ingen (det som er inne i parantesen) Respons: PS_Session (det som operasjonen returnerer. PS_Session er en egen klasse.

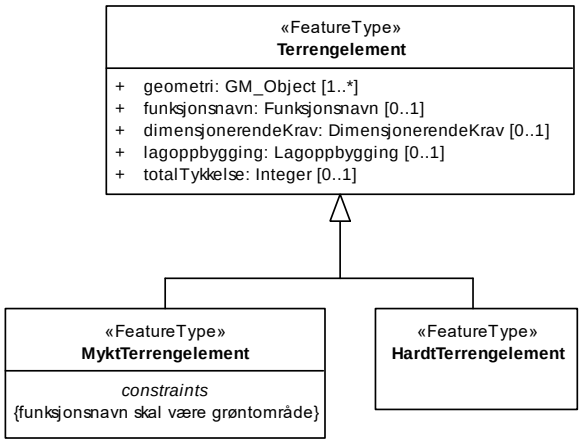
/req/general/operation	Operasjoner skal modelleres som instanser av metaklassen Operasjon i henhold til Figur 11.2 (GFM).
/req/uml/operation	En instans av metaklassen Operasjon skal implementeres som UML Operation i klassen som representerer objekttypen.

11.4.2.7 **Restriksjoner**

Elementer i en UML-modell kan ha restriksjoner som definerer regler utover det som er mulig å modellere eksplisitt med språket selv. Restriksjoner er en viktig komponent for å bidra til økt dataintegritet og består av en tekstlig forklaring og en maskinlesbar del. Den maskinlesbare delen må angis med en spesiell syntaks, OCL (Object Constraint Language).

Dessverre er restriksjonene ofte godt skjult i modellen og kan ikke i alle tilfeller vises i diagrammene. Derfor er det viktig å ha en dokumentasjon av modellen som tar vare på både tekst og OCL-syntaks.

Tabell 11.8 Restriksjoner

Restriksjon		Objekttypen MyktTerrengelement har fått en restriksjon der den tekstlige beskrivelsen vises i constraints-delen av klassen: "funksjonsnavn skal være grøntområde". OCL syntaks for denne restriksjonen ser slikt ut: inv: self.funksjonsnavn = "grøntområde"

/req/general/constraint	Restriksjoner i et applikasjonsskjema som ikke kan uttrykkes på andre måter med modellelementene beskrives med et restriksjonsspråk som er egnet for CSL. Der man har valgt UML som CSL, er restriksjonsspråket vanligvis OCL.
/req/uml/constraint	Restriksjoner beskrives i OCL og i klart språk i klassen til det modellelementet det skal begrense.
/krav/applikasjonsskjema/ dokumentasjon/restriksjoner	Tekstlig beskrivelse og OCL syntaks av alle restriksjoner knyttet til modellelementer i et applikasjonsskjema skal vises i dokumentasjonen av modellen.

11.4.2.8 **Verditildeling**

Verdier kan tildeles egenskaper på mange ulike måter. I mange applikasjoner eller for mange egenskaper er ikke metoden kjent eller det er ikke tilstrekkelig interesse i å modellere denne eksplisitt i modellen. I prinsippet er det en prosess knyttet til tildeling av verdi for alle egenskaper. Verditildeling er en assosiasjonsklasse (metaklasse) for enhver prosess hvor verdier blir tildelt egenskaper.

Verditildeling benyttes ofte i forbindelse med modellering av observasjoner. (Se senere kapittel).

11.4.3 Modellering av geometri og topologi

11.4.3.1 **Introduksjon**

Denne standarden beskriver hvordan stedfestingen av geografiske objekter skjer i form av geometri- og topologiegenskaper. **Hovedregelen er å ikke bruke mer avanserte mekanismer enn det en trenger for å oppfylle brukerkravene**. På den annen side, en skal være forsiktig med å moderere brukerkravene slik at de kan løses ved enkle primitiver.

Denne standarden omhandler de geometri- og topologitypene som er beskrevet i NS-EN ISO 19107 Modell for å beskrive geometri og topologi schema og som kan inngå i et applikasjonsskjema. Flere av de geometri og topologitypene som er angitt ligger langt utenfor det som er praktisk anvendt i SOSI i dag. Men på den annen side, angivelse av geometri- og topologiprimitiver i SOSI del1 versjon 5 skal ikke hindre brukermiljøene i å ta i bruk de nødvendige mekanismer der brukerkravene tilsier det og software finnes eller skal utvikles. Av denne grunn gir standarden anbefalinger på bruk uten å forby andre og mer avanserte mekanismer.

Anvendelse av geometri- og topologityper må også avpasses den utvekslingsplattform som en ønsker å legge til grunn. Alle de geometrityper som er angitt her kan realiseres i GML. De geometritypene som er støttet i SOSI-realisering (dvs. SOSI prikkformat) er angitt i kapittel 11.4.3.3.6 og vist i Tabell 11.15. Hvis en ønsker å legge andre utvekslingsplattformer til grunn, slik som f.eks. RDF må en selv ta ansvar for å sjekke om det finnes standard vokabularer i f.eks. RDF for de mekanismer en ønsker å bruke.

Stedfestingsdata beskrives i form av geometri eller topologi, angitt hver for seg eller i kombinasjon, for å beskrive den eller de romlige egenskap(er) til en objekttype. Det høyeste nivået av geometri er GM_Object. Denne er en abstrakt supertype for alle geometrityper. Tilsvarende er det høyeste nivået for topologi TP_Object. Det at de er abstrakte betyr at de ikke kan implementeres direkte, og at alle subtyper med tilhørende interpolasjonsmetoder kan forekomme i dataene, og likevel være konforme med spesifikasjonen. Av denne grunn brukes ikke disse direkte i UML-applikasjonsskjema som skal realiseres direkte.

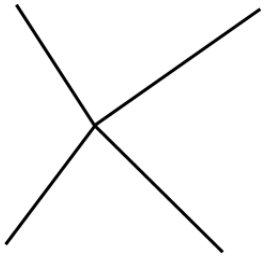
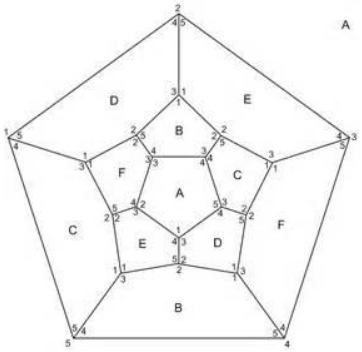
Geometrien brukes for å angi den kvantitative beskrivelsen i form av koordinater og matematiske funksjoner, slik som dimensjon (0d til 3d), posisjon, størrelse, form og orientering. Det som kjennetegner geometri i forhold til topologi er at geometri endres ved transformasjon fra et geodetisk referansesystem eller koordinatsystem til et annet, mens topologien er uendret.

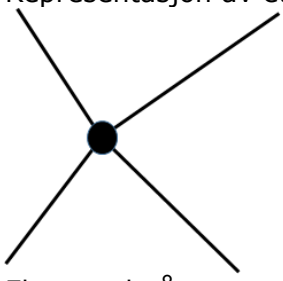
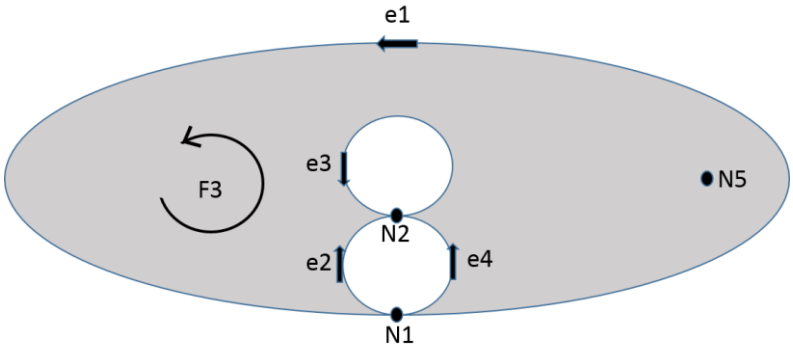
Topologien beskriver karakteristikken ved geometriske figurer som blir uendret ved endring i "rommet", f.eks. ved transformasjon mellom ulike referansesystemer/koordinatsystemer. Topologien inneholder informasjon om sammenhengen mellom geometriske primitiver som kan avledes fra den underliggende geometri. Konseptet som ligger til grunn for topologi er å tillate organisering av sammensatte geometriske objekter i en topologisk struktur uavhengig av strukturen til deres romlige egenskaper.

Det henvises til NS-EN ISO 19107:2005 Modell for å beskrive geometri og topologi, annex C og D for nærmere beskrivelse av geometri og topologi samt eksempler på hvordan disse er satt sammen.

Standarden skiller mellom 4 nivåer i beskrivelsen av geometri og topologi, i stor grad basert på kompleksitet, gitt i Tabell 11.9.

Tabell 11.9 Nivåer for geometri og topologi.

Nivå	**Forklaring**
Geometriske primitiver	Geometriske primitiver er objekter som beskriver geometrisk konfigurering, dvs. et enkelt, sammenknyttet og homogent element i rommet. De inkluderer punkt (Point), kurver (Curves), flater (surfaces) og legemer (Solid). Ustrukturerte samlinger av geometriske primitiver går ofte under betegnelsen "spagetti" data.  Eksempel på geometriske primitiver i form av kurver (linjer), som er en 1-dimensjonal geometrisk primitiv.
Geometriske komplekser	Samling geometriske primitiver, som sammen danner et geometrisk kompleks.  Eksempel: Dersom en vil samle all geometridata om et vegnettverk i et enkelt objekt "Vegnett" og dette da skal inneholde den geometriske plasseringen av alle lenker og kryss som inngår i veinettet, blir dette en kompleks geometri.

Topologiske primitiver	Representasjon av et enkelt, ikke oppdelt, topologisk element.  Eksempel på sammenhengen mellom geometriske primitiver beskrevet med node.
Topologiske komplekser (med geometrisk realisering)	 Det grå område i figuren viser et eksempel på kompleks topologi. F3 (Face) har en ytre avgrensning e1 (edge) og indre avgrensninger e2, e3 og e4. Det er også en isolert N5 (node). +e1 og -e3, former "boundary ring" i F3. Tilsvarende også +e1 og +e2 -e4. N5 (node) er en isolert node. sets{-e3}, {+e2, -e4}, {+e5, -e5} er alle indre avgrensninger. (ref OGC Topology_Investigation)

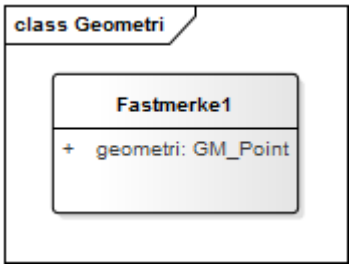
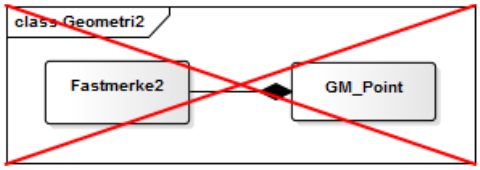
Dersom et UML-applikasjonsskjema krever eksplisitt topologisk informasjon utvides geometriske komplekser til å inkludere strukturen til topologiske komplekser. "Chain-node" topologi er et 1-dimesjonalt topologisk kompleks. Det som vanligvis kalles full topologi i et kartografisk 2D miljø er et 2-dimensjonalt topologisk kompleks realisert med geometriske objekter i et 2D koordinatsystem.

Denne standarden fokuserer på bruk av geometriske primitiver og komplekser, men hindrer ikke at topologiske objekter kan benyttes der brukerkravene tilsier dette. Merk at topologien ofte bygges opp av geometriske objekter i et systems interne struktur, og trenger ikke nødvendigvis å overføres som en del av utvekslingsformatet. Spesielt geografiske informasjonssystemer har ofte denne funksjonaliteten innebygd. Andre systemer, som ikke har mulighet til å bygge opp en topologi, men har støtte for romlige operatorer, vil kunne ha nytte av å få topologien overført.

11.4.3.2 **Angivelse av geometri/topologi i en UML-modell**

Geometrien angis som en datatype til egenskapen.

Tabell 11.10 Angivelse av geometri i en UML-modell.

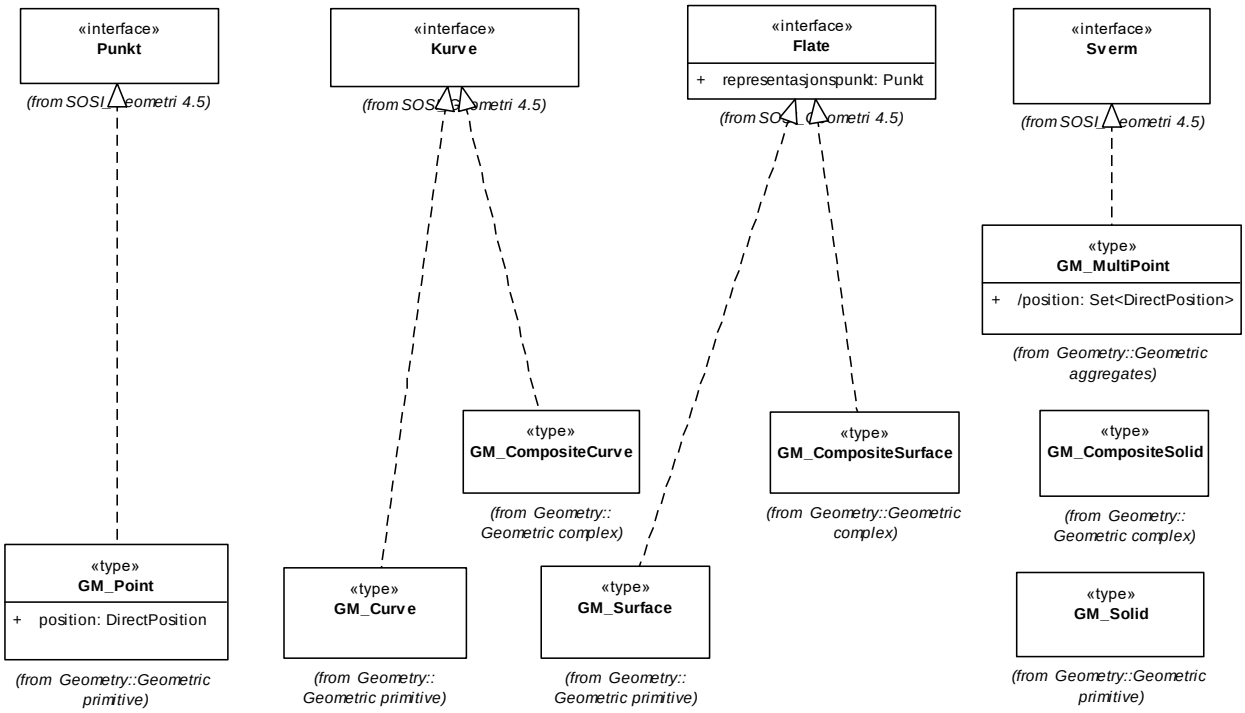
Geometri-egenskap		Geometritypen angis som en datatype knyttet til egenskapen
		Geometrien er modellert som en assosiasjon til et geometrisk objekt.

/req/spatial/object	Geometri og topologi skal angis som en egenskap til en objekttype. Merknad: Dette er en innsnevring for objekttyper i forhold til ISO 19109:2015, som også tillater at dette kan modelleres som en assosiasjon til et geometrisk objekt.
/req/spatial/schema	Geometri- og topologityper skal være i henhold til NS-EN ISO 19107, samt SOSI geometriegenskapene Punkt, Kurve, Flate og Sverm.

11.4.3.3 **Geometri**

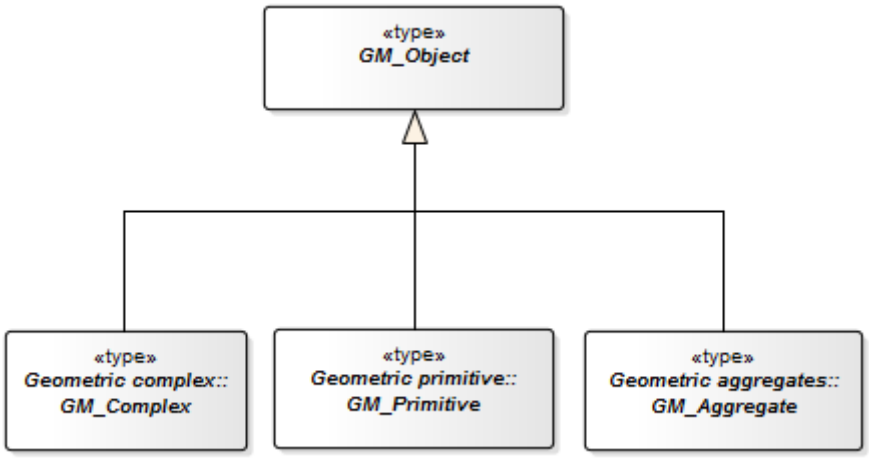
Figur 11.3 viser hvordan geometrier modellert med de tradisjonelle interface-typene Punkt, Kurve og Flate må realiseres til iso-typer for å kunne kontrollere implementeringen i GML eller SOSI-format. For kurver og flater ser man at man da må velge om man vil mappe til heleid eller delbar (GM_Composite) type. Og man må velge hvordan man eventuelt skal modellere for å ta vare på flaters sentralpunkt.

Dersom man kun beholder den tradisjonelle typen i modellen vil den for GML automatisk konverteres til sin heleide primitiv-variant, og til flater uten sentralpunkt.



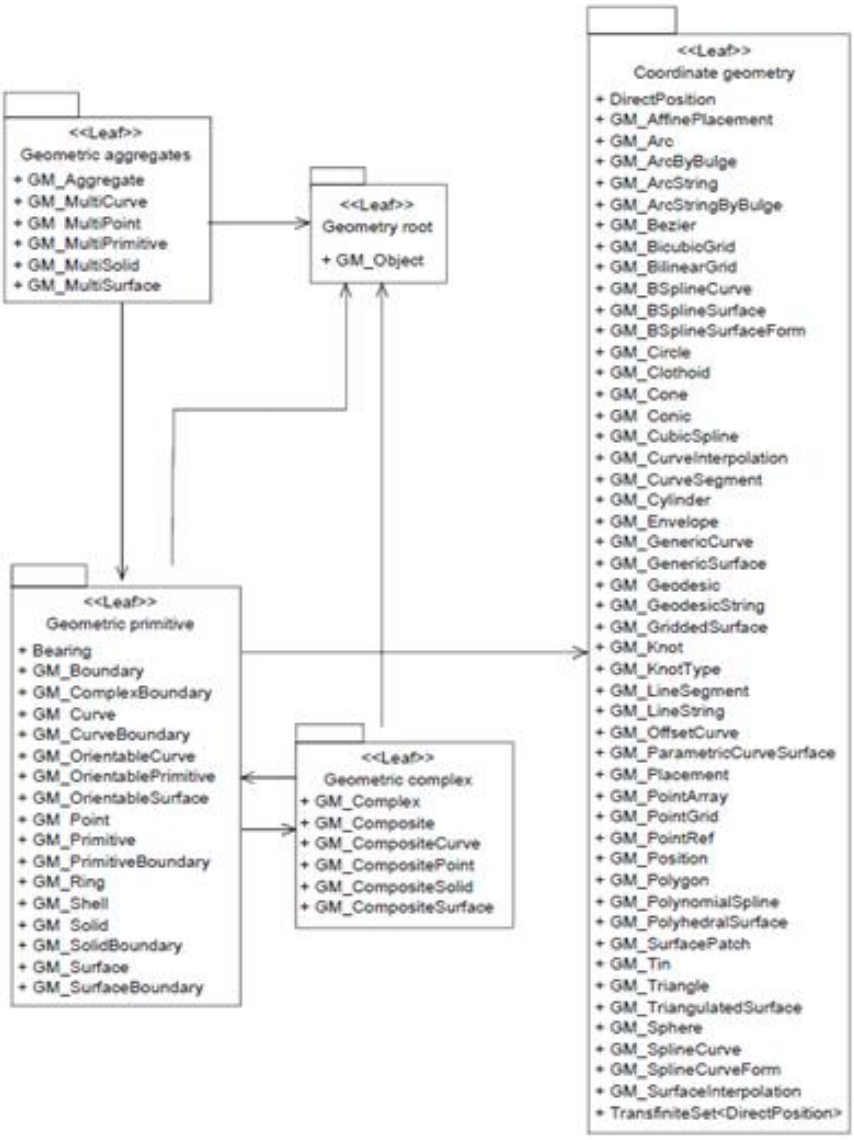
Figur 11.3 Realisering av norske geometrityper til ISO-geometrityper

Figur 11.4 viser de abstrakte hovedklassene av ISO geometrimodell slik denne er spesifisert i NS-EN ISO 19107 modell for å beskrive geometri.



Figur 11.4 ISO-geometrimodell jfr. NS-EN ISO 19107 (forenklet).

Figur 11.5 viser alle ulike subtyper av GM_Object. De er alle abstrakte. Med abstrakt menes at disse ikke vil finnes i en produktspesifikasjons UML-modell, her vil en bruke instansierbare geometrityper, eventuelt supplert med angivelse av segmenttyper.



Figur 11.5 ISO-geometri og segmenttyper.

Tabellene i kapitlene under viser de ulike geometrityper som kan inngå i en UML-modell. Den siste kolonnen viser hvilke av disse ISO 19109 beskriver som kan brukes i et UML-applikasjonsskjema. ISO 19109 legger klare begrensninger på hvilke geometrityper som er tillatt brukt i et UML-applikasjonsskjema.

Tabellen under viser hvilke geometriske objekter og typer som kan benyttes i et UML-applikasjonsskjema.

Tabell 11.11 Geometrityper

Geometriske primitiver	Geometriske komplekser	Geometriske aggregater
GM_Point GM_Curve GM_Surface GM_Solid	GM_CompositePoint GM_CompositeCurve GM_CompositeSurface GM_CompositeSolid (GM_Complex)	(GM_Aggregate) GM_MultiPoint GM_MultiCurve GM_MultiSurface GM_MultiSolid

Note: Tabellen lister bare opp høyest nivå, subtyper kan også brukes i et UML-applikasjonsskjema.

/anbefaling/geometrinivå	Bruk ikke mer komplekse geometrityper enn det som er nødvendig for å angi geometri. De enkleste og mest vanlige er GM_Point, GM_Curve, GM_Surface, GM_Solid.

11.4.3.3.1 Geometriske primitiver

De geometriske primitivene utgjør de enkleste modellelementer for geometri.

/krav/enkleGeometriskePrimitiver	Geometriske primitiver angis i form av GM_Point (Punkt), GM_Curve (Kurve), GM_Surface (Flate) og GM_Solid.

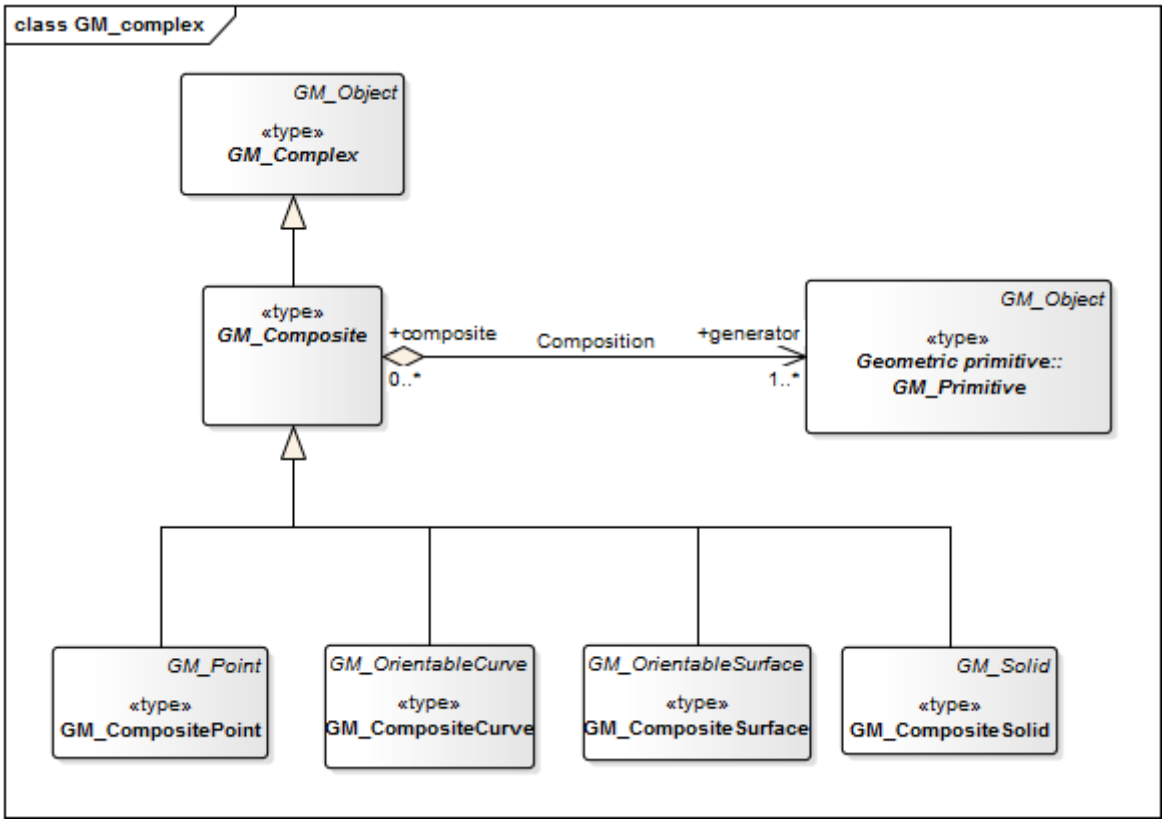
Det finnes også et sett geometriske primitiver som ytterligere detaljerer geometrien.

/krav/andreGeometriskePrimitiver	Dersom en har behov for andre geometriske primitiver benyttes GM_OrientableCurve, GM_OrientableSurface, GM_PolyhedralSurface, GM_TriangulatedSurface, GM_Tin.

Merknad: Disse geometriske primitivene bør kun brukes der brukstilfellene krever en slik geometri.

11.4.3.3.2 Geometriske komplekser

Geometriske komplekser brukes for å representere stedfestingen til en objekttype som et sett sammenhengende geometriske primitiver. Instanser av GM_Complex tillater geometriske primitiver å bli benyttet av geometrien til flere objekttyper, slik tilfellet er ved delt geometri mellom flater.



Figur 11.6 Geometriske komplekser

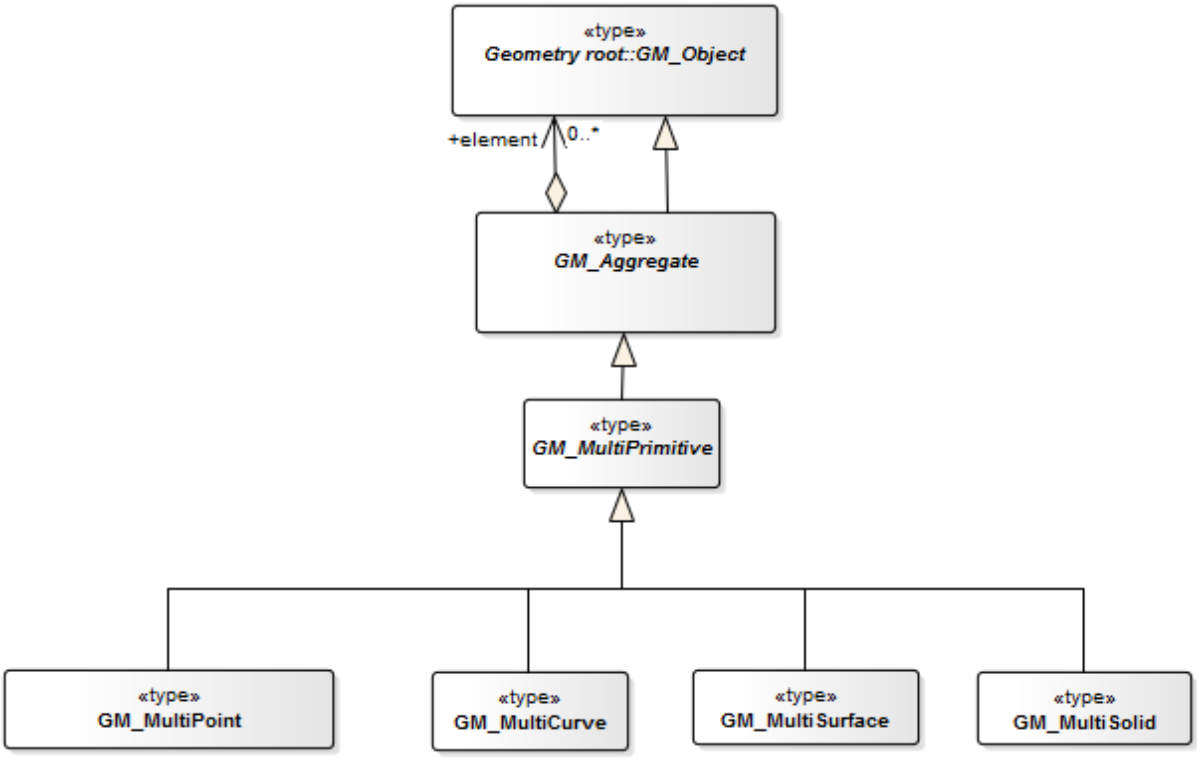
Figur 11.6 forklarer sammenhengen mellom GM_Composite, GM_CompositePoint, GM_CompositeCurve, GM_CompositeSurface og GM_CompositeSolid.

For nærmere angivelse av krav til bruken av GM_Complex henvises til ISO 19109:2015 kapittel 8.7.4.3 Geometric complexes.

/req/spatial/complex	Samlinger av sammenhengende geometriobjekter skal benytte GM_Complex dersom de kun er sammenhengende i sine avgrensninger. Subklasser av GM_Complex kan modelleres dersom spesielle restriksjoner behøves. Objekter som deler elementer av sin geometri skal presenteres som GM_Complex som er subkompleks innen et større GM_Complex.
/req/spatial/composite	En GM_Composite skal benyttes til å representere komplekse objekter som har geometri fra flere geometriprimitiver.
/krav/geometriskeKomplekser	Geometriske komplekser angis i form av GM_CompositePoint, GM_CompositeCurve, GM_CompositeSurface og GM_CompositeSolid.

11.4.3.3.3 **Geometriske aggregater**

Geometriske aggregater utgjør en tilfeldig samling av GM_Objects som ikke krever noen spesiell geometrisk struktur, og som igjen kan inneholde andre aggregater.



Figur 11.7 Geometriske aggregater (forenklet)

Figur 11.7 forklarer sammenhengen mellom GM_Aggregate, GM_MultiPrimitive, GM_MultiPoint, GM_MultiCurve, GM_MultiSurface og GM_MultiSolid.

For nærmere angivelse av krav til bruken av GM_Aggregate henvises til ISO 19109:2015 kapittel 8.7.4.2 Geometric aggregates.

/req/spatial/aggregate	GM_MultiPoint, GM_MultiCurve, GM_MultiSurface, eller GM_MultiSolid skal benyttes ved behov for samlinger av geometriklasser av samme primitiv type. Brukerspesifikke geometriklasser kan modelleres spesielt i egne klasser som er subtyper av GM_MultiPrimitive.

11.4.3.3.4 **Segmenttyper**

Segmenttyper er knyttet til geometriske primitiver, og gir en nærmere beskrivelse av oppbyggingen av geometrien.

Det er ingen krav om at segmenttype må angis, men dersom den ikke er angitt må en mottaker ta høyde for at alle segmenttyper kan forekomme.

Segmenttyper kan ikke angis direkte i en modell, men kan angis som en restriksjon til de geometritypene som brukes i UML-applikasjonsskjema.

Tabell 11.12 viser segmenttyper for kurver.

Tabell 11.12 Segmenttyper for kurver

UML-type (ISO 19107)	**Nærmere forklaring**
GM_ArcString	
GM_Arc	Spesialisering av GM_ArcString
GM_Circle	Spesialisering av GM_Arc
GM_ArcsStringByBulge	Ingen restriksjon med denne – ingen direkte mapping til GML
GM_ArcByBulge	Subtype av GM_ArcStringByBulge
GM_SplineCurve	Ingen restriksjon med denne – ingen direkte mapping til GML
GM_PolynomialSpline	Spesialisering av GM_SplineCurve
GM_CubicSpline	Spesialisering av GM_PolynomialSpline
GM_BsplineCurve	Spesialisering av GM_SplineCurve
GM_Bezier	Spesialisering av GM_BSplineCurve
GM_Clothoide	
GM_GeodesicString	
GM_Geodesic	Spesialisering av GM_GeodesicString
GM_LineString	
GM_LineSegment	Spesialisering av GM_LineString
GM_Conic	
GM_OffsetCurve	

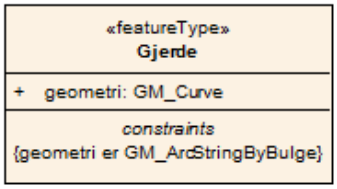
Tabell 11.13 viser segmenttyper for flater.

Tabell 11.13 Segmenttyper for flater

UML-type (ISO 19107)	Nærmere forklaring
GM_ParametricCurveSurface	Spesialisering av GM_SurfacePatch
GM_GriddedSurface	Spesialisering av GM_ParametricCurveSurface
GM_Cone	Spesialisering av GM_GriddedSurface
GM_BicubicGrid	Spesialisering av GM_GriddedSurface
GM_Cylinder	Spesialisering av GM_GriddedSurface
GM_BilinearGrid	Spesialisering av GM_GriddedSurface
GM_Sphere	Spesialisering av GM_GriddedSurface
GM_BSplineSurface	Spesialisering av GM_GriddedSurface
GM_Polygon	Spesialisering av GM_SurfacePatch
GM_Triangle	Spesialisering av GM_Polygon
GM_ParametricCurveSurface	
GM_GriddedSurface	Spesialisering av GM_ParametricCurveSurface

Disse UML-typene kan ikke angis direkte i en modell, men kan angis som en restriksjon til geometritypen. Et eksempel på dette er gitt i Tabell 11.14.

Tabell 11.14 Eksempel på restriksjon for geometritype.

Segmenttype		Geometrien for Gjerde er av typen GM_Curve med en restriksjon om at denne skal implementeres som en GM_ArcStringByBulge. Tilsvarende beskrives dette som et OCL uttrykk, f.eks: inv:self.geometri.segment.oclIsTypeOf (GM_ArcStringByBulge)

Dersom segmenttype ikke er angitt som restriksjon på klasser må en regne med at alle segmenttyper kan forekomme i et datasett.

anbefaling/segmenttype	Dersom en ønsker å spesifisere segmenttype gjøres dette som en restriksjon til klassen med knytning mot geometriegenskapen og vises i modellen.

11.4.3.3.5 Modellering av objekttyper som har geometrier som kan deles

Med deling av geometri menes når to instanser av en objekttype begge bruker samme instans av et GM_Object som verdi for den geometriske egenskapen.

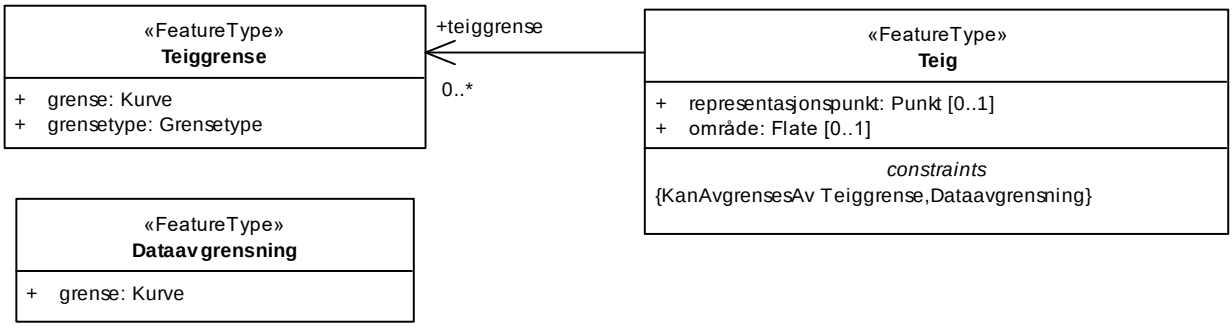
I tidligere versjoner av SOSI har en historisk hatt fokus på delt geometri. SOSI-format geometri spesifiserte delt geometri, hovedbudskapet var at geometrien var delt, men at dette ikke var påkrevd, dvs. at det kunne forekomme instanser i en SOSI-fil med heleid geometri.

Denne standarden beskriver ikke fordeler/ulemper med henholdsvis delt eller heleid geometri, men stiller krav til hvordan dette skal modelleres.

Vi har derimot ingen historikk knyttet til geometrisk beskrivelse av legemer (solid), og anbefaler ikke bruk av delt geometri for disse.

Delt geometri i et UML-applikasjonsskjema angis i form av en restriksjon på objekttypene. Navnet på restriksjonen skal starte med KanAvgrensesAv og fortsette med en liste med navnene på de objekttypene som har geometri som kan avgrense denne objekttypen.

Figur 11.8 viser hvordan man i en restriksjon angir at geometrien for en teig skal følge geometrien i en objekttype fra en angitt liste med objekttyper. Det er behov for en normal assosiasjon kun dersom teigen har behov for å referere egenskapsverdier (grensetype) i avgrensningsobjektet. Avgrensingsobjekttyper uten dette behovet skal ikke assosieres.



Figur 11.8 Objekttyper som har geometrier som kan deles

Dette er i tråd med hvordan denne type assosiasjoner er modellert i INSPIRE (eksempel AdministrativeUnit).

/req/spatial/shared	Et applikasjonsskjema kan i en restriksjon kreve at to eller flere instanser deler sin geometri presist. Geografiske assosiasjoner som ikke eksisterer i de underliggende geometriske eller topologiske primitivene skal modelleres som assosiasjoner i applikasjonsskjemaet.
/krav/deltGeometri	Delt geometri angis ved eksplisitt restriksjon som angir avgrensningskurven i objekttypen den avgrenser samt med restriksjoner på klassene jfr. Figur 11.8.

Modeller etter denne standarden skal ikke være synlig preget av begrensninger i en enkelt plattform. Men konfigurasjonssinformasjon for enkeltplattformer kan legges i restriksjoner eller tagged values. Dette beskrives mer utførlig i den aktuelle plattformrealiseringsstandarden (eks: SOSI generell del – Realisering i SOSI-format 5.0).

/krav/plattformuavhengig	Plattformspesifikke ting skal ikke vises i standardiserte fagområdemodeller eller produktspesifikasjonsmodeller.

/anbefaling/plattformtillegg	Plattformspesifikke ting kan legges i egne restriksjoner med fast syntaks eller i tagged values inne i alle standardiserte fagområdemodeller og produktspesifikasjoner. Restriksjoner ved navn KanAvgrensesAv kan etterfølges av en liste med aktuelle objekttypenavn fra gjeldende applikasjonsskjema.

11.4.3.3.6 **Geometri - for realisering i GML og SOSI-syntaks**

GML og SOSI-syntaks sine geometrimodeller inneholder færre geometriegenskaper enn EN-ISO 19107. Denne tabellen beskriver de deler av NS-EN ISO 19107 som kan benyttes for de tilfeller en ønsker å realisere modellen i GML eller SOSI-syntaks.

Tabellen under viser sammenhengen mellom UML-geometri, SOSI-format geometri og GML-geometri.

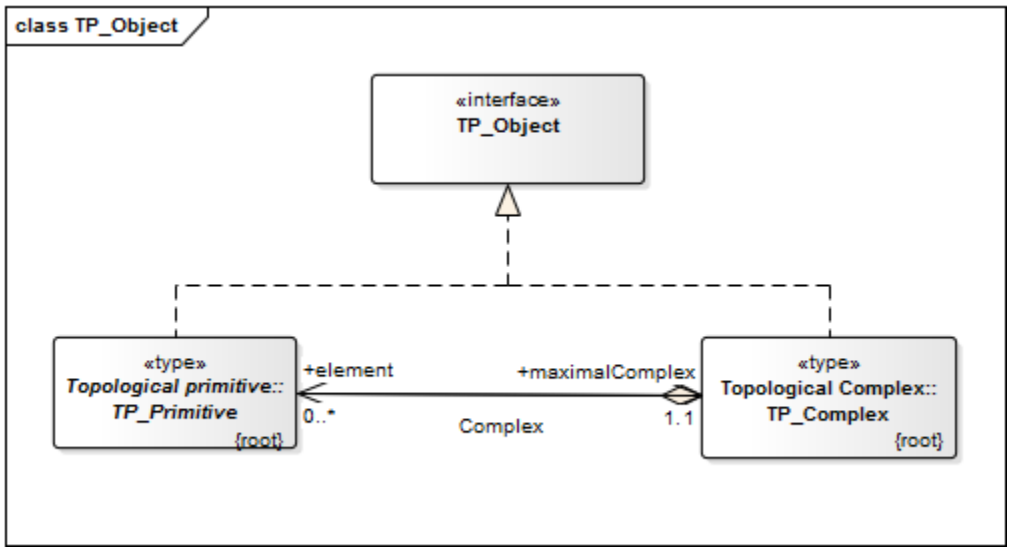
Tabell 11.15 Sammenheng mellom UML-geometri og geometri i GML og SOSI-format.

UML-class (ISO 19107)	SOSI-syntaks geometri	GML object element
GM_Point	.PUNKT	gml:Point
GM_Curve	.KURVE	gml:Curve
GM_Surface	.FLATE	gml:Surface
GM_MultiPoint	.SVERM	gml:MultiPoint
Segmenttyper (modelleres ikke direkte, men som en restriksjon til geometrien)		
GM_Arc	.BUEP	gml:Curve med gml:Arc
GM_Bezier	.BEZIER	gml:Curve med gml:Bezier
GM_Circle	.SIRKELP	gml:Curve med gml:Circle
GM_Clothoide	.KLOTOIDE	gml:Curve med gml:Clothoid

11.4.3.4 **Topologi**

Topologiske komplekser inneholder eksplisitte beskrivelser av sammenhengen mellom de topologiske primitiver som de er satt sammen av. Bruken av topologiske komplekser er i hovedsak å muliggjøre metoder fra "computational topology" til geometriske komplekser. En annen bruk er å muliggjøre beskrivelsen av sammenhengen mellom objekttyper uavhengig av deres geometriske konfigurering (jfr. ..REF i SOSI-format).

Figur 11.9 viser ulike realiseringer av TP_Object, i form av TP_Primitive og TP_Complex. De er alle abstrakte. Med abstrakt menes at disse ikke vil finnes i en produktspesifikasjons UML-modell, her vil en bruke instansierbare topologityper, eventuelt supplert med angivelse av segmenttyper.



Figur 11.9 Topologiske primitiver og topologiske komplekser (forenklet).

Topologiske primitiver og komplekser som kan brukes ved modellering er angitt i tabellen under:

Tabell 11.16 Topologiske primitiver og komplekser.

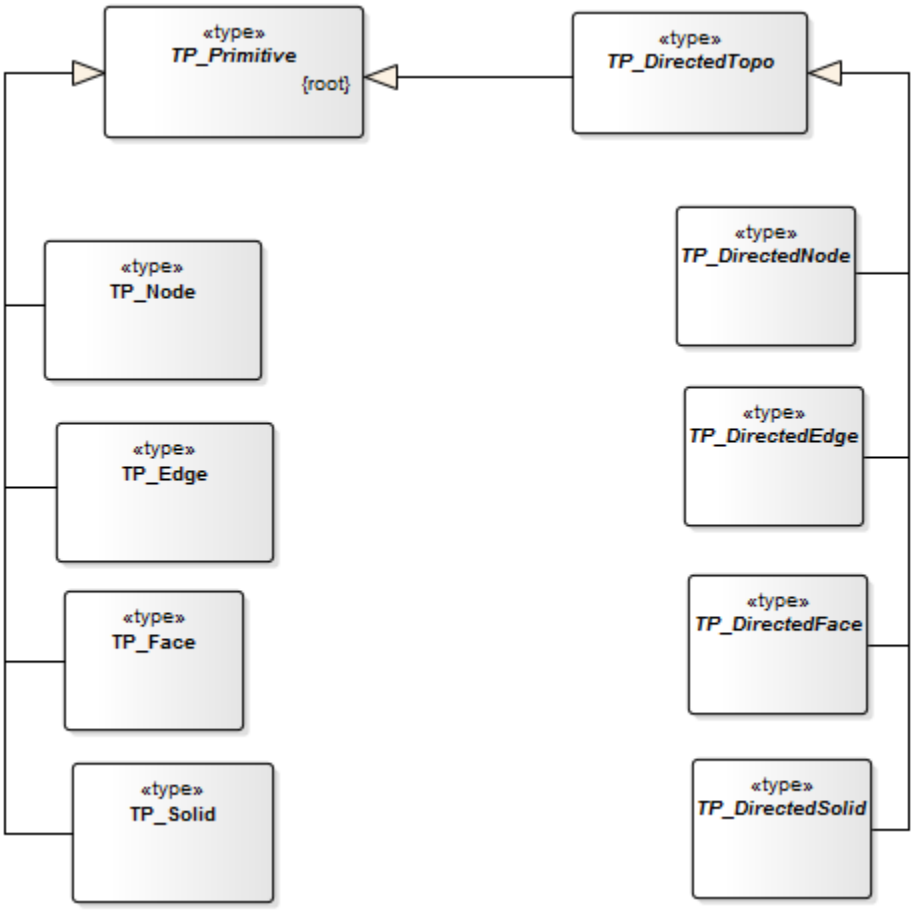
Topologiske primitiver	Topologiske komplekser
TP_Node TP_Edge TP_Face TP_Solid TP_DirectedNode TP_Directed Edge TP_DirectedFace TP_DirectedSolid	TP_Complex
Note: Tabellen lister bare opp høyest nivå, subtyper kan også brukes i et UML-applikasjonsskjema	

/krav/GMLtopologi	Modellering av topologi i UML-applikasjonsskjema for produktspesifikasjoner skal benytte geometrityper som angitt i tabell 11.16 (over) enten angitt direkte eller som restriksjon til abstrakt topologi.

/anbefaling/topologibruk	Topologi bør ikke brukes i et applikasjonsskjema med mindre det er absolutt påkrevet.

11.4.3.4.1 **Topologiske primitiver**

Som det fremkommer av Figur 11.10 er TP_Primitive supertypen til TP_Node, TP_Edge, TP_Face og TP_Solid, samt TP_DirectedNode, TP_DirectedEdge, TP_DirectedFace og TP_DirectedSolid, som alle er instansierbare.



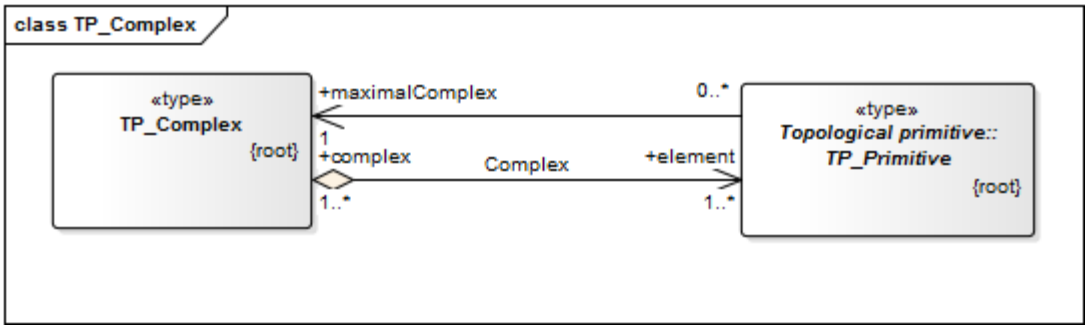
Figur 11.10 Topologiske primitiver (forenklet).

For nærmere angivelse av krav til bruken av topologi henvises til ISO 19109:2015 kapittel 8.7.4.6 Topological complexes.

11.4.3.4.2 Kompleks topologi

Kompleks topologi beskriver hvordan topologiske primitiver er sammensatt. Det er i hovedsak to anvendelsesområder:

1. "Computational topology" - beskrivelse av sammenhengen til komplekse geometrier
2. beskrivelse av sammenhengen mellom objekttyper uavhengig av deres geometri



Figur 11.11 Sammenhengen mellom TP_Complex og TP_Primitive (forenklet).

/req/spatial/topo-complex	TP_Complex skal brukes som en romlig egenskap til en objekttype der applikasjonen krever en eksplisitt beskrivelse av sammenhengen mellom de geometriske primitivene som representerer objekttypen.

For nærmere angivelse av krav til bruken av topologi henvises til ISO 19109:2015 kapittel 8.7.4.6 Topological complexes.

11.4.4 Modellering av raster og bildedata (Coverage)

Kapittel 11.4.3 Modellering av geometri og topologi fokuserer på objekter i den virkelige verden hvor egenskapene har enkelt verdier og er statiske. For noen anvendelsesområder er det mer hensiktsmessig å bruke en modell som fokuserer på variasjoner av egenskapsverdier i rom og tid, formalisert som raster og/eller bildedata, her omtalt som et Coverage.

Coverage er en objekttype som oppfører seg som en funksjon for å returnere verdier for enhver posisjon innenfor sin romlige og/eller temporale område. Coverage funksjoner brukes for å beskrive karakteristikken til et fenomen i den virkelige verden som varierer over rom og/eller tid, slik som temperatur, høyde, fordampning, bilde, etc. Eksempler på grupper av slike funksjoner er raster, "triangulated irregular network" (TIN), punktsamling eller flerdimensjonale grid.

En egenskap hvor verdiene varierer som en funksjon av tid kalles et temporalt Coverage (tids-serier). Et kontinuerlig Coverage (continuous Coverage) assosieres med en metode for å interpolere verdier mellom posisjoner i et Coverage.

Selv om et Coverage er en objekttype er det naturlig å skille i begrepsbruken mellom objekt- og Coverage typer. Et UML-applikasjonsskjema kan inneholde både objekt- og coveragetyper.

Skillet mellom objekttyper og coveragetyper er i stor grad sammenfallende med det som ble kalt "vektor" og "raster" data i tradisjonelle GIS implementasjoner. Tabell 11.17 viser mulige coveragetyper.

Tabell 11.17 Typer av coverage.

Abstrakte coveragetyper	**Diskre coveragetyper**	**Kontinuerlige coveragetyper**
CV_Coverage	CV_DiscretePointCoverage	CV_ThiessenPolygonCoverage
CV_DiscreteCoverage	CV_DiscreteGridPointCoverage	CV_ContinousQuadrilateralGridCoverageCoverage
CV_ContinousCoverage	CV_DiscreteCurveCoverage	CV_HexagonalGridCoverage
	CV_DiscreteSurfaceCoverage	CV_TINCoverage
	CV_DiscreteSolidCoverage	CV_SegmentedCurveCoverage

/req/coverage/schema	1. Enhver spesifikasjon av en coverage funksjon i et UML-applikasjonsskjema skal være i overensstemmelse med NS-EN ISO 19123. 2. Et UML-applikasjonsskjema som bruker coverage funksjoner skal importere coverage skjema slik dette er spesifisert i NS-EN ISO 19123:2007.* 3. En coverage funksjon skal defineres som en egenskap til en objekttype (FeatureType) hvor egenskapverdien er en realisering av en av typene i tabellen over. Coverage-skjema vil ligge som en modell i SOSI-modellregister.

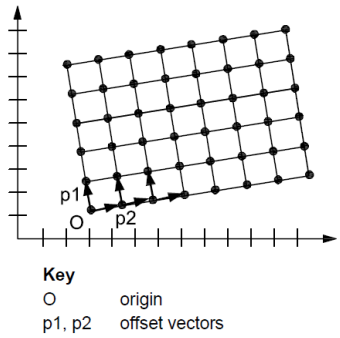
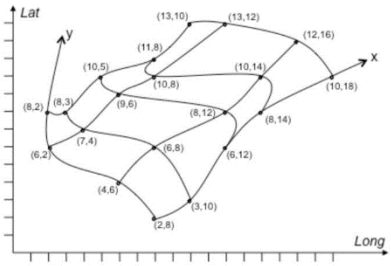
For å sikre implementasjon foreslås at coveragetypene i Tabell 11.18 benyttes i et UML-applikasjonsskjema.

Tabell 11.18 Foreslåtte coveragetyper i UML-applikasjonsskjema.

Abstract coverage types	**Coverages i form av "domain/range"**	**Coverages i form av "geometry/value pairs"**
n/a	RectifiedGridCoverage ReferenceableGridCoverage Timeseries (from WaterML 2.0)	TimeSeriesTVP (from WaterML 2.0)

Anbefaling: CoverageTyper	Coverage-typer brukt i et applikasjonsskjema bør være enten RectifiedGridCoverage, ReferencableGridCoverage eller TimeSeriesTVP fra WaterML 2.0, eller en subtype.

Tabell 11.19 Gridtyper

RectifiedGrid	 (Source: ISO 19136:2007)	Et grid (rutenett) hvor det er en affin transformasjon mellom grid-koordinatene og koordinatene i et koordinatreferansesystem. NB: Tilsvarer SOSI .RASTER.
ReferencableGrid	 (Source: GML 3.3.0)	Et grid (rutenett) assosiert med en transformasjon som kan benyttes til å konvertere grid koordinatverdier til koordinatverdier referert til et koordinatreferansesystem.

For nærmere informasjon om modellering av coverage henvises til NS-EN ISO 19123:2007 Geografisk informasjon - Modell for overdekkende tematisk (ISO 19123:2005). Ytterligere informasjon finnes også i INSPIRE D2.5_v3.4 kapittel 10.5

11.4.5 Modellering av nettverk og lineære referanser

Stedfesting med nettverk og/eller lineære referanser benyttes for å posisjonere objekter, egenskaper eller hendelser i et nettverk.

/krav/NettverkOgLineæreReferanser	Modellering av nettverk og lineære referanser i UML-applikasjonsskjema skal baseres på krav i SOSI generell del - Nettverk og lineære referanser 5.0.

11.4.6 Modellering av tidsaspekt

11.4.6.1 **Introduksjon**

NS-EN ISO 19108 gir en basis for å modellere temporale egenskaper, operasjoner og assosiasjoner.

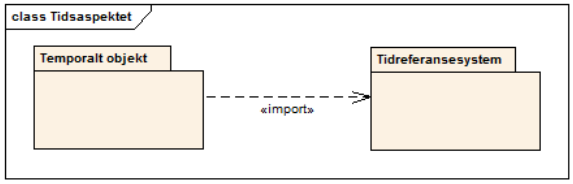
Tid kan modelleres som sammensatte temporale objekter med tilhørende tidsreferansesystem, eller som enkel angivelse av tid uten at tidsreferanser/soner er inkludert.

/req/temporal/succession	Dersom objekthistorikk kreves skal dette modelleres i applikasjonsskjemaet som en selvassosiasjon. Navning og multiplisitet skal være passende for objekthistorikk (succession).

11.4.6.2 **Tid som temporalt objekt**

Dette kapitlet omhandler modellering av tidsaspekt slik dette fremkommer i NS-EN ISO 19108 (2006) Modellering av tidsaspekt samt hvordan disse modellelementene er brukt i et UML-applikasjonsskjema jfr ISO 19109:2015 Rules for application schema.

Tid som temporalt objekt består av to pakker, temporale objekter og et temporalt referansesystem.



Figur 11.12 Tid som temporalt objekt

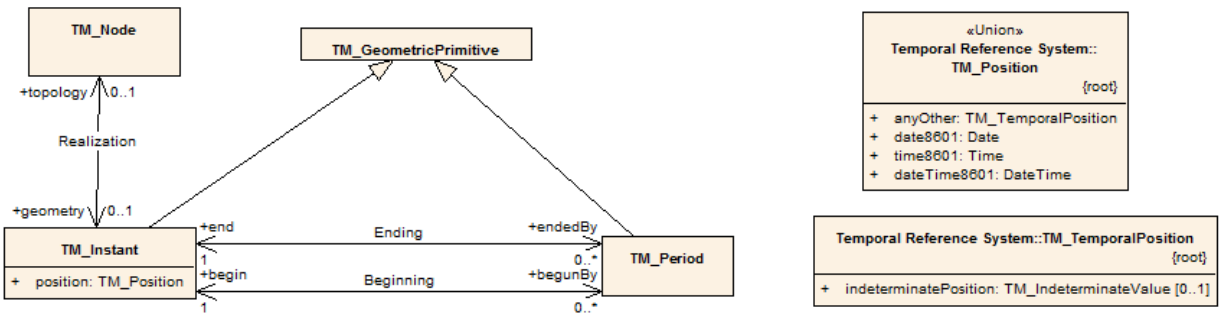
Selve modellen er nærmere beskrevet i NS-EN ISO 19108 Model for stedfesting. Dette er en relativt omfattende modell, som ikke er forklart i denne SOSI-standarden.

Tabell 11.20 viser de geometriske og topologiske primitiver samt topologisk kompleks som det er tillatt å benytte i et UML-applikasjonsskjema:

Tabell 11.20 Temporale primitiver og komplekser

Temporale geometriske primitiver	**Temporale topologiske primitiver**	**Temporale komplekser**
TM_Instant TM_Period	TM_Node TM_Edge	TM_TopologicalComplex

De mest vanlige primitivene er TM_Instant og TM_Periode.



Figur 11.13 Tid som temporalt objekt (2)

/req/temporal/schema	Enhver beskrivelse av tid som benyttes i geografiske data skal være i samsvar med spesifikasjoner angitt i ISO 19108:2002.

/anbefaling/temporaltObjekt	Temporale objektelementer (TM_xx) bør kun benyttes der det er viktig å trekke med seg tidsaspektets referansesystem i de enkelte dataobjektene, og er egnet til mer komplekse modeller som for eksempel innehar historikk.

For angivelse av en tidsperiode kan datatypen TM_Period benyttes. (ISO 19108)

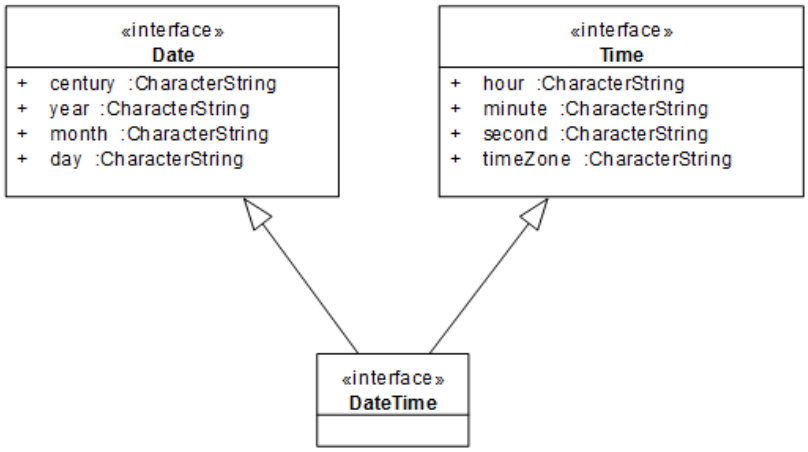
TM_Period er bygd opp av to TM_Instant som angir henholdsvis start og slutt tidspunkt for en periode. Se Figur 11.13. TM_Instant representerer et spesifikt tidspunkt i henhold til ISO 8601.

11.4.6.3 **Tid som tematisk objekt**

Det er også mulig å bruke Date, DateTime og Time direkte. Disse er å betrakte som tematiske egenskaper og ikke temporale, da det ikke er knyttet noe tidreferansesystem til disse.

/anbefaling/tid	Datatypene Date, Time og DateTime kan benyttes der datasettets metadata angir tidreferansesystem eller tidssone.

I SOSI-metoden brukes datatypene Date, Time og DateTime for å angi dato og tidspunkt. For en nærmere beskrivelse av hvordan dato og tid kan angis refereres det til ISO 8601.



Figur 11.14 Tid som tematisk objekt

Som vist i Figur 11.14 kan dato og tid angis som en kombinasjon av datatypene Date og Time ved å bruke datatypen DateTime.

11.4.7 Modellering av observasjoner

Observasjoner og målinger (O&M) er beskrevet i NS-EN ISO 19156:2013 Geografisk informasjon - Observasjoner og målinger. Observasjoner er en viktig instansiering av verditildelingsklassen i GFM.

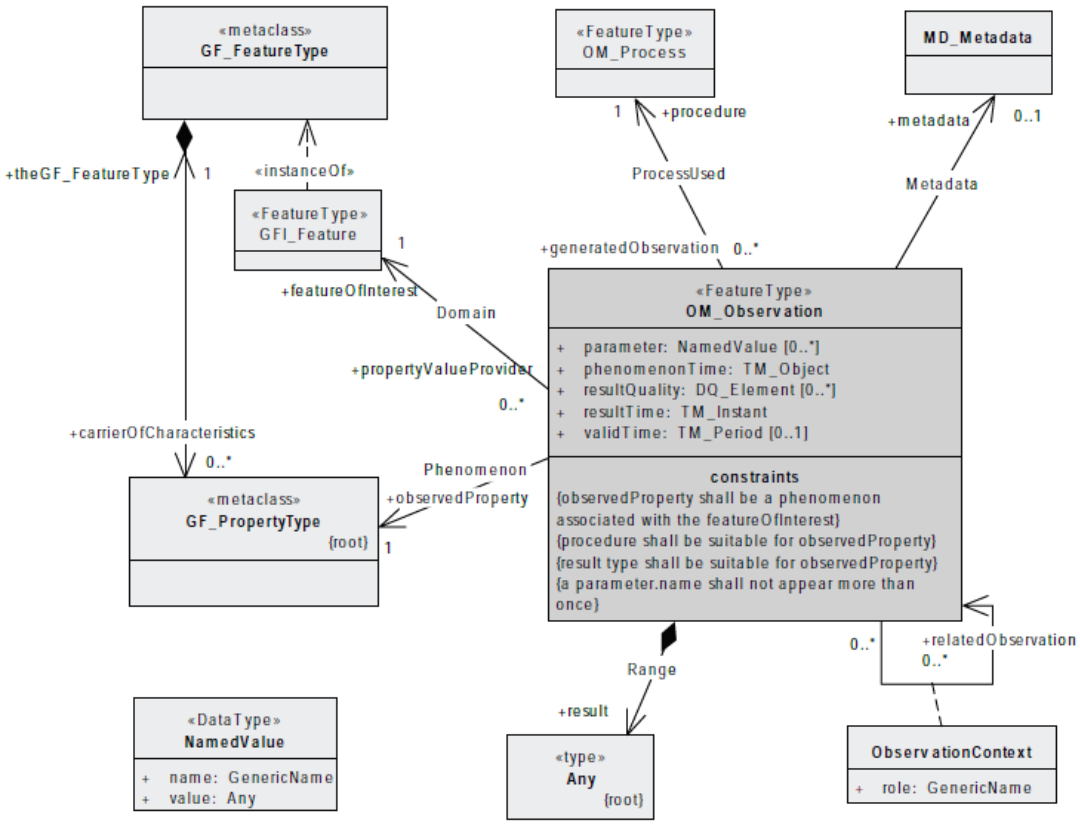
Det er to typer egenskaper knyttet til objekter:

- Verdi som er tildelt i en prosess av en autoritet (eksempel navn, type, etc). Disse verdiene er i utgangspunktet eksakte og modelleres som "vanlige" egenskaper.
- Verdi tildelt gjennom en observasjonsprosedyre. Disse er ofte estimater med en feilrate assosiert med verdien, og fremkommer ofte gjennom sensor, instrument, algoritme eller prosess. Dersom disse verdiene skal benyttes i en analyse er det ofte nødvendig å vite noe om denne feilraten og f.eks. tidspunktet for observasjonen.

Dette kapitlet beskriver modellering av verdier som er tildelt gjennom en observasjon.

En observasjon er en hendelse som skjer på et bestemt tidspunkt eller tidsintervall og resulterer i en verdi som knyttes til et fenomen. Selve observasjonen er i seg selv et objekt, siden den har egenskaper og identitet.

Figur 11.15 viser modellen for observasjoner, og hvordan en observasjon gjennom en GFI_Feature er knyttet opp til en objekttype i henhold til den generelle objektmodellen:



Figur 11.15 Modell for observasjoner og målinger

For nærmere forklaring til innholdet i modellen henvises til NS-EN ISO 19156:2013 Geografisk informasjon - Observasjoner og målinger (ISO 19156:2011).

/req/observation/model	Observasjoner i et applikasjonsskjema skal modelleres i henhold til skjema for observasjoner angitt i NS-EN ISO 19156 Geografisk informasjon - Observasjoner og målinger.
/req/observation/property	Verdien til en egenskap som er observert (målt) i data som er i henhold til applikasjonsskjema skal begrenses til å være en egenskapstype fra applikasjonsskjemaets objektkatalog, og skal være en karakteristisk egenskapstype for objekttypen.

11.4.8 Modellering av kvalitet

Kvalitet er et begrep som kan forstås på mange måter, og også på ulike nivåer. Ofte er kvalitet en del av metadata, men dessverre er det en utbredt oppfatning at metadata kun beskriver datasett (NS-EN ISO 19115-1 Geografisk informasjon - Metadata - Del 1: Grunnprinsipper).

I SOSI versjon 5 kan kvalitet modelleres på to måter:
- ved angivelse av datatypen Posisjonskvalitet som har vært med i SOSI i mange versjoner
- ved angivelse av DQ elementer som angitt i NS-EN ISO 19157:2013 Datakvalitet og i den norske standarden Geodatakvalitet.

11.4.8.1 Kvalitet i form av angivelse av DQ-elementer

ISO 19157:2013 Datakvalitet definerer en rekke kvalitetselementer for å beskrive kvalitet på geografiske data. Hver type kvalitetselement beskriver et bestemt aspekt ved datakvaliteten på geografiske data, inndelt i seks hovedkategorier:

- **stedfestingsnøyaktighet** – hvor godt stedfestingen samsvarer med virkeligheten eller fasit
- **kvalitet på tidfesting** - kvaliteten på egenskaper som definerer tid eller tidsavhenggheter
- **egenskapskvalitet** – nøyaktighet og riktighet av kvantitative egenskaper og assosiasjoner
- **fullstendighet** - beskrivelse av hvilke enheter som er med i datasettet i forhold til de som burde være med
- **logisk konsistens** – sammenheng mellom regler som gjelder for dataene og aktuelle data
- **egnethet** – beskrivelse som viser i hvilken grad dataene er egnet til formålet

Kvalitetselementene er spesifisert som egen datamodell i ISO 19157. Flere av kvalitetslementene karakteriserer hele datasett, og beskrives i datasettets metadata.

Imidlertid, kan noen av kvalitetselementene brukes når en ønsker å beskrive kvalitet på instanser av objekttyper
- kvalitet knyttet til objektet
- kvalitet knyttet til egenskapene for objektet

og modelleres i et applikasjonsskjema.

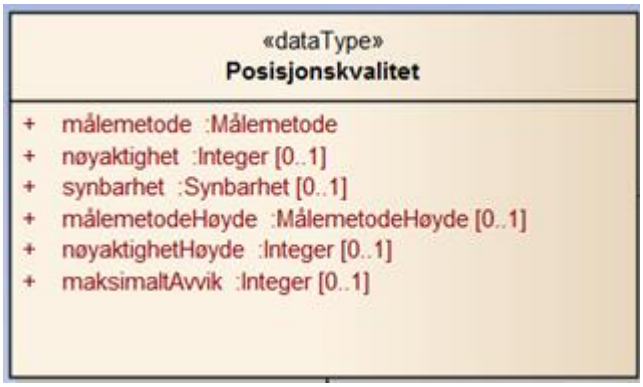
Tabell 11.21 viser subtyper av DQ_Element fra NS-EN ISO 19157 til bruk i et applikasjonsskjema.

Tabell 11.21 Subtyper av DQ_Element fra NS-EN ISO 19157.

Kategori	Kvalitetselement	Forklaring
Stedfestingsnøyaktighet		
Absolutt stedfestingsnøyaktighet	DQ_AbsoluteExternalPositionalAccurracy	Hvor nær stedfestet posisjon er den sanne posisjonen

Nabonøyaktighet	DQ_RelativeInternalPositionalAccuracy	Hvor bra stedfestet posisjon samsvarer med andre stedfestede posisjoner
Tidfestingskvalitet		
Tidsnøyaktighet	DQ_AccuracyOfATimeMeasurement	Hvor nær angitte tidsverdier er sanne verdier
Tidskonsistens	DQ_TemporalConsistency	Mål på om tidsangivelser er ordnet riktig
Tidsgyldighet	DQ_TemporalValidity	Mål på om tidsangivelser er gyldig
Egenskapskvalitet		
Klassifikasjonsriktighet	DQ_ThematicClassificationCorrectness	Sammenligning mellom anvendt klassifisering og virkelighet
Ikke-kvantitative egenskapsriktighet	DQ_NonQuantitativeAttributeCorrectness	Mål på om ikke-kvantitative verdier er riktige eller feil
Kvantitativ egenskapsnøyaktighet	DQ_QuanatitativeAttributeAccuracy	Hvor nær kvantitative verdier er sanne verdier

11.4.8.2 Kvalitet i form av datatypen Posisjonskvalitet

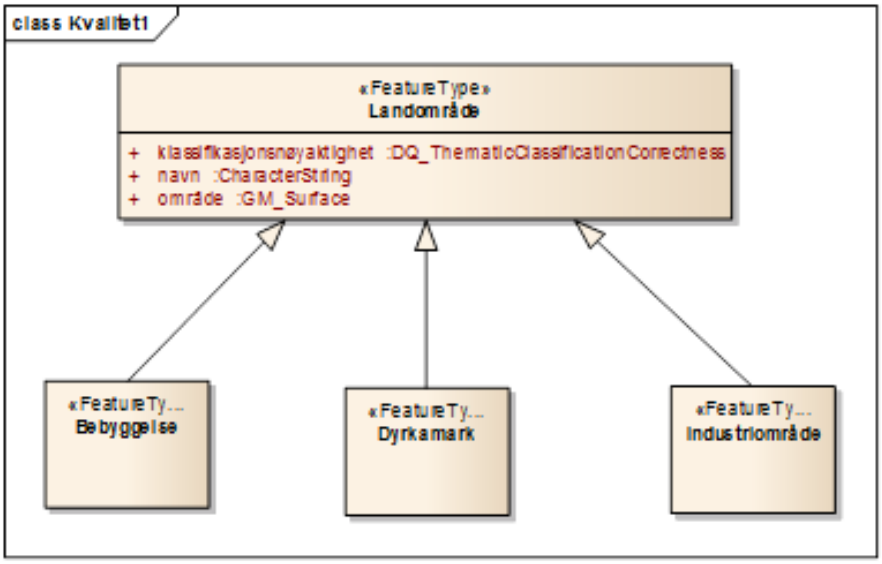


Figur 11.16 UML-modell for datatype Posisjonskvalitet

11.4.8.3 Krav til modellering av kvalitet

/req/quality/attribute	Informasjon om kvalitet knyttet til individuelle instanser av objekttyper eller egenskaper skal modelleres som egenskaper for de tilfeller der kvaliteten av en instans fraviker kvaliteten som er angitt for hele datasettet eller deler av et datasett.
/krav/kvalitet/egenskap	Kvalitetsinformasjon knyttet til selve objektet skal modelleres som egenskap i klassen som representerer objekttypen, enten ved bruk av datatypen Posisjonskvalitet eller ved å bruke en av subtypene til DQ_Element som angitt i tabellen over eller DQ_DataQuality.
/req/quality/additional-quality	Brukerspesifiserte kvalitetselementer skal defineres som subtyper av klassen DQ_Element eller en av dens subtyper, og følge reglene for profilering av standard skjema.
/req/quality/attribute-quality	Karakterisering av egenskapers kvalitet skal implementeres i henhold til /req/uml/attributeOfAttribute.

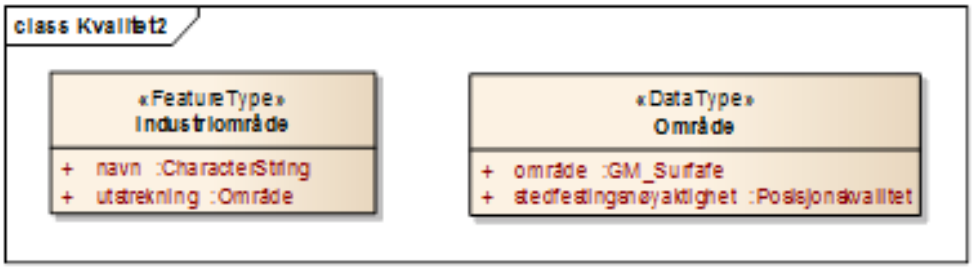
Eksempel 1: Figur 11.17 viser at egenskap klassifikasjonsnøyaktighet er bærer av informasjon om hvor nøyaktig klassifiseringen er.



Figur 11.17 UML-modell for bruk av DQ element

/req/quality/attribute	Kvalitetsinformasjon knyttet til egenskaper i objektet skal modelleres som en egenskap i en egen klasse med stereotype «dataType», sammen med aktuelle egenskap som skal beskrives med kvalitet (attributeOfAttribute)

Eksempel 2: Figur 11.18 viser at stedfestingsnøyaktighet er knyttet til egenskapen utstrekning, og bruk av Posisjonskvalitet.



Figur 11.18 UML-modell for bruk av Posisjonskvalitet

/anbefaling/kvalitet	En enkel form av posisjonskvalitet kan modelleres i henhold til datatypen Posisjonskvalitet. Dersom en ønsker en mer detaljert angivelse av kvalitet, eller ønsker at data skal være utgangspunkt også for leveranser internasjonalt anbefales det å bruke subtyper av DQ_element.

11.4.8.4 Andre egenskaper som direkte eller indirekte gir informasjon om kvalitet

SOSI_Objekt har en rekke egenskaper som direkte eller indirekte kan gi informasjon om kvalitet. De viktigste egenskapene er:

- digitaliseringsmålestokk
- prosesshistorie
- verifiseringsdato
- posisjonskvalitet

11.5 Forenklede regler for modellering av et fagområdestandards applikasjonsskjema

11.5.1 Introduksjon

SOSI del 2 Generell objektkatalog utgjør fagområdestandarder, dvs. modeller som representerer en tverrsektoriell enighet innen et fagområde. En fagområdestandards UML-applikasjonsskjema skal (i motsetning til UML-modellen for en produktspesifikasjon) ikke implementeres direkte men være utgangspunkt for produktspesifikasjoner. Dette fordi elementene i et fagområdeapplikasjonsskjema vil i de fleste tilfeller være for generiske og ufullstendige enn at de uten endringer kan implementeres. Multiplisiteter til egenskaper og assosiasjonsroller bør eventuelt innstrammes samtidig som det kan være aktuelt å ta i bruk elementer fra SOSI Fellesegenskaper, SOSI-Objekt eller andre fagområder.

/krav/applikasjonsskjema/ fagområde/implementasjon	Et fagområdeapplikasjonsskjema skal ikke implementeres direkte. Elementer fra et SOSI fagområdeapplikasjonskjema skal implementeres med utgangspunkt i en produktspesifikasjons applikasjonsskjema, som baserer seg på ett eller flere fagområdeapplikasjonskjemaer.

/anbefaling/fellesegenskaper	Et fagområdeapplikasjonsskjema bør ikke modellere inn fellesegenskaper jfr. kapittel 11.6.1 SOSI_Fellesegenskaper og SOSI_Objekt.

Der disse fellesegenskapene anses som absolutt nødvendige, kan det dokumenteres ved hjelp av realisering fra SOSI_Objekt (med et utvalg av aktuelle fellesegenskaper) og subtyping av realiseringsklassen slik at det er tydelig under arbeid med produktspesifikasjoner hvilke fellesegenskaper som allerede er med.

/anbefaling/delbarGeometri	Det anbefales å ikke spesifisere delbar geometri i en fagområdemodell.

For fagområdestandardenes UML-modell gjelder i utgangspunktet de samme regler som er definert for andre applikasjonsskjemaer under kapittel 11.4 med unntak av forenklingene i de følgende kapitlene.

11.5.2 Geometri

I utgangspunktet benyttes abstrakte geometrityper. Tabell 11.22 viser abstrakte geometrityper som benyttes for et fagområde.

Tabell 11.22 Geometiske objekter

Geometriske objekter	**Forklaring**
GM_Object	Det mest overordnede geometriske objektet(root). Alle geometriske primitiver, komplekser og aggregater er subtyper av denne. Dvs ingen begrensning i realisering.
GM_Primitive	Overordnet objekt (root) for alle geometriske primitiver, dvs GM_Curve, GM_Point, GM_Surface og GM_Solid (samt eventuelle subtyper)
GM_Complex	Overordnet objekt (root) for alle geometriske komplekser, dvs GM_CompositeCurve, GM_CompositePoint, GM_CompositeSurface og GM_CompositeSolid (samt eventuelle subtyper)
GM_Aggregate	Overordnet objekt (root) for alle geometriske aggregater, dvs GM_MultiCurve, GM_MultiPoint, GM_MultiSurface og GM_MultiSolid (samt eventuelle subtyper)

Tabell 11.23 Eksempel på bruk av geometri i fagområde.

GM_Object	geometri: GM_Object[1..*]	GM_Objekt er en abstrakt supertype for alle geometrityper. Dvs at det ikke er tatt noen stilling til om objektet skal realiseres som punkt, linje, flate eller legeme. Kan bare benyttes i fagormådestandarder.
GM_Point	geometri: GM_Point	Her presiseres det at geometrien er et punkt, og at det bare er en geometri.
Enklere angivelse av geometri i fagområdestandard	Tidligere var dette angitt som følger, samt med en constraint som sier at minst en geometri må være med: • posisjon: Punkt[0..1] • grense: Kurve [0..1] • område: Flate[0..1] I SOSI versjon 5 kan dette angis som • geometri: GM_Primitive	Dette blir en forenkling i forhold til SOSI versjon 4. [GM_Primitive dekker både punkt, linje, flate og legeme]. Der en ønsker å være mer presis på geometri kan GM_Point, GM_Curve, GM_Surface og GM_Solid benyttes.

Tabell 11.24 Nærmere angivelse av abstrakte geometrityper.

Geometritype	**Realisering**
geometri: GM_Object	Denne kan realiseres i form av alle geometriske primitiver, komplekser og aggregater, og er den mest abstrakte angivelsen av geometri.
geometri: GM_Primitive	Denne kan realiseres i form av alle geometriske primitiver, men utelater realisering i form av komplekser og aggregater.

geometri: GM_Complex	Denne kan realiseres i form av alle geometriske komplekser, men utelater realisering i form av primitiver og aggregater.
geometri: GM_Aggregate	Denne kan realiseres i form av alle geometriske aggregater, men utelater realisering i form av primitiver og komplekser.

For å være bakoverkompatibel med tidligere versjoner av SOSI tillates også følgende geometritypene vist i Tabell 11.25 brukt i en fagområdestandard.

Tabell 11.25 Bakoverkompatible geometrityper for SOSI-formatet.

Geometritype	**Tilsvarer**
Kurve	GM_Curve
Punkt	GM_Point
Flate	GM_Surface
Sverm	GM_MultiPoint

Eksempel fastmerke: +geometri: Punkt

/krav/fagområdeGeometri	Modellering av geometri i en fagormådestandard skal ikke være så spesifikk at en forhindrer nødvendig realisering i en produktspesifikasjon.

/anbefaling/abstraktGeometri	Dersom man ikke er sikker på hvilken type geometri som skal brukes kan man velge GM_Primitive eller GM_Object. Merknad: GM_Object og GM_Primitive er ikke implementerbare, i dette tilfelle må en implementerbar subtype til disse velges i en produktspesifikasjonsmodell.

11.5.3 Topologi

I utgangspunktet benyttes abstrakte topologityper. De abstrakte topologitypene gitt i Tabell 11.26 benyttes for et fagområde:

Tabell 11.26 Abstrakte topologityper

Topologiske objekter	**Forklaring**
TP_Object	Det mest overordnede topologiske objektet (root). Alle topologiske primitiver og komplekser er subtyper av denne. Dvs. ingen begrensning i realisering.
TP_Primitive	Overordnet objekt (root) for alle topologiske primitiver, dvs TP_Node, TP_Edge, TP_Face, TP_Solid, samt TP_DirectedNode, TP_DirectedEdge, TP_DirectedFace, TP_DiretedSolid, samt eventuelle subtyper). Tilsvarer GM_Primitve for geometriske objekter.
GM_Complex	Overordnet objekt (root) for alle geometriske komplekser, dvs GM_CompositeCurve, GM_CompositePoint, GM_CompositeSurface og

	GM_CompositeSolid (samt eventuelle subtyper). TP_Complex er en organisert struktur av TP_Primitives.

Eksempel eiendom: +topologi: TP_Object[0..1]

/krav/fagområdeTopologi	Modellering av topologi i en fagormådestandard skal ikke forhindre nødvendig realisering i en produktspesifikasjon.

En kan også modellere med både geometri og topologi i en fagområdestandard, slik som

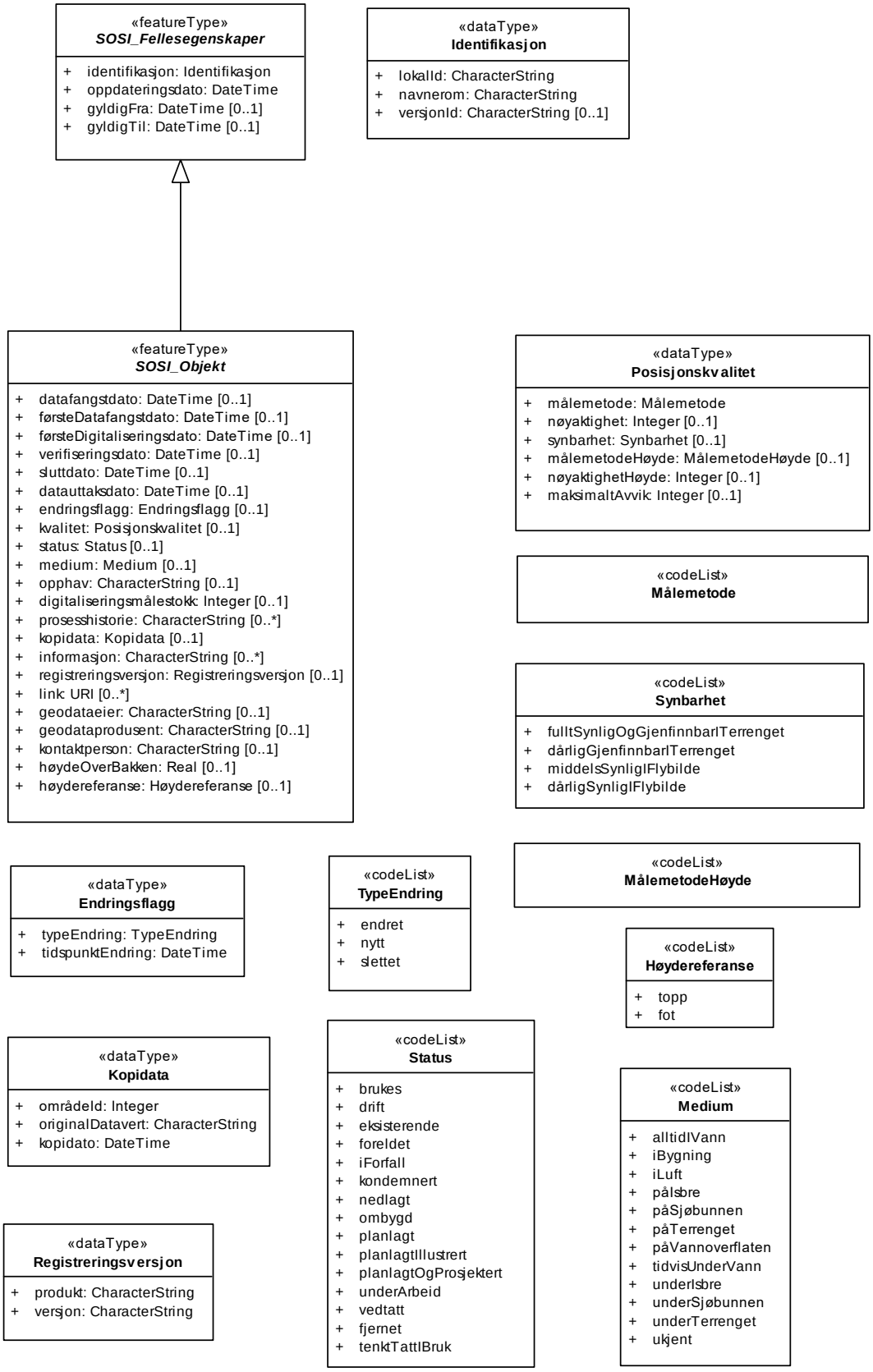
Eksempel Eiendom:
+ område: GM_Object[1..]*
+topologi: TP_Object[0..1]

11.6 Generelle typer

11.6.1 SOSI_Fellesegenskaper og SOSI_Objekt

Fagområder i SOSI inneholder ikke egenskaper som skal være felles i alle modeller. Det er laget egne abstrakte klasser som inneholder disse fellesegenskapene og som anbefales å være utgangspunkt for å lage implementerbare fellesegenskapsklasser i en produktspesifikasjon.

Det er to påkrevde og to anbefalte egenskaper i den nye klassen SOSI_Fellesegenskaper. I tillegg er det en det en ren videreføring av alle tidligere fellesegenskaper i klassen SOSI_Objekt som nå er en subtype av SOSI_Fellesegenskaper. SOSI_Objekt har i tillegg fått noen nye egenskaper som var modellert som standard datatyper i tidligere versjoner. Kodelistene Målemetode og MålemetodeHøyde er tomme og for validering må man hente verdiene fra et sentralt forvaltet register. (i dag gml:Dictionary)

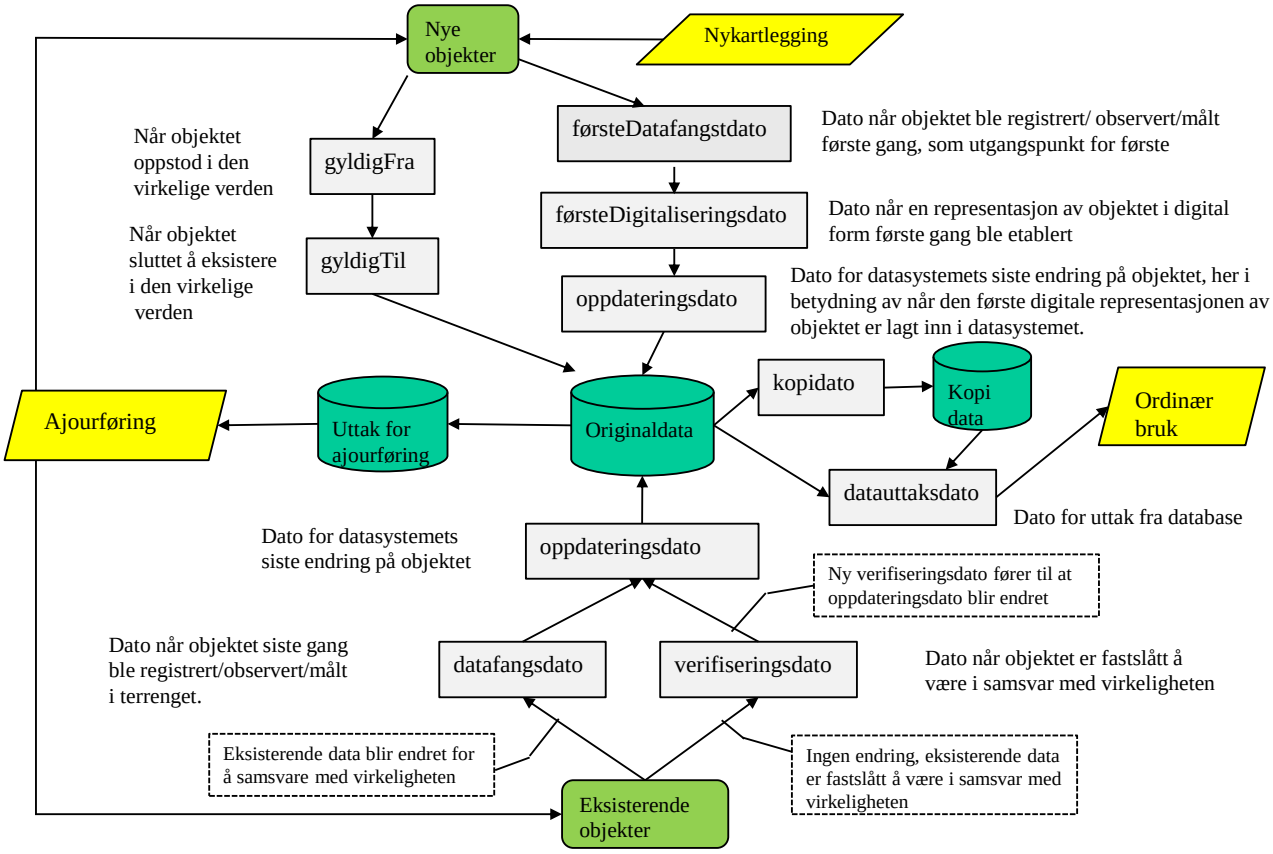


Figur 11.19 UML-modell over SOSI_Fellesegenskaper og SOSI_Objekt

Nærmere angivelse av modell for livsløpssyklus

Sammenhengen mellom de datoene som er angitt i dette kapittel er beskrevet i Figur 11.20.

Det som imidlertid ikke fremkommer er datoen gyldig til. Dette er datoen for når et objekt opphører å eksistere i den virkelige verden. En slik angivelse er best egnet for menneskeskapte objekter, som en administrativt vet når opphører å eksistere.



Figur 11.20 Eksempel på livsløpssyklus for geografiske objekter

Tekstlig beskrivelse av UML-modellen er angitt i Anneks E.1

11.7 Objekttyper med tydelige fellestrekk

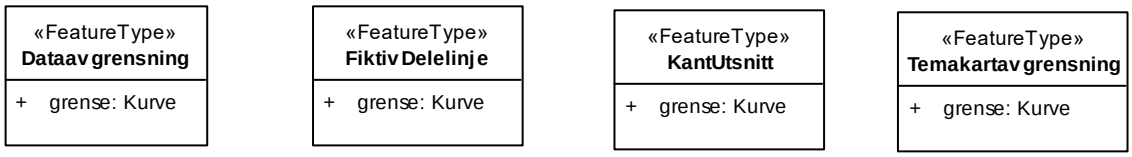
Dette kapitlet beskriver objekttyper og egenskaper som kan være av generell interesse, men som går ut over det som ligger i SOSI_Objekt. Disse ansees å være uavhengig av fagområdene, men av en slik karakter at de bør vurderes brukt i produktspesifikasjoner.

Dersom en produktspesifikasjon skal implementeres i SOSI-format er det et absolutt krav at alle objekttyper med flategeometri må hente sine avgrensninger fra objekttyper med kurvegeometri. Dette oppfylles enten ved å eksplisit modellere inn egne objekttyper for alle forhold, eller ved å benytte en spesiell mekanisme (som å legge inn instanser av en formatspesifikk objekttype Flateavgrensning).

Dette er ikke tilfelle for implementasjon i GML-format der flatene greit kan ha sin egen heleide geometri. Der bør man ikke modellere med slike egne avgrensingsobjekttyper dersom disse ikke har egne egenskaper (eks. egenskapen grensetype) eller de skal merkes spesielt. (Eks. Dataavgrensning – som betyr at objektet kan fortsette på andre siden.)

11.7.1 Avgrensningslinjer

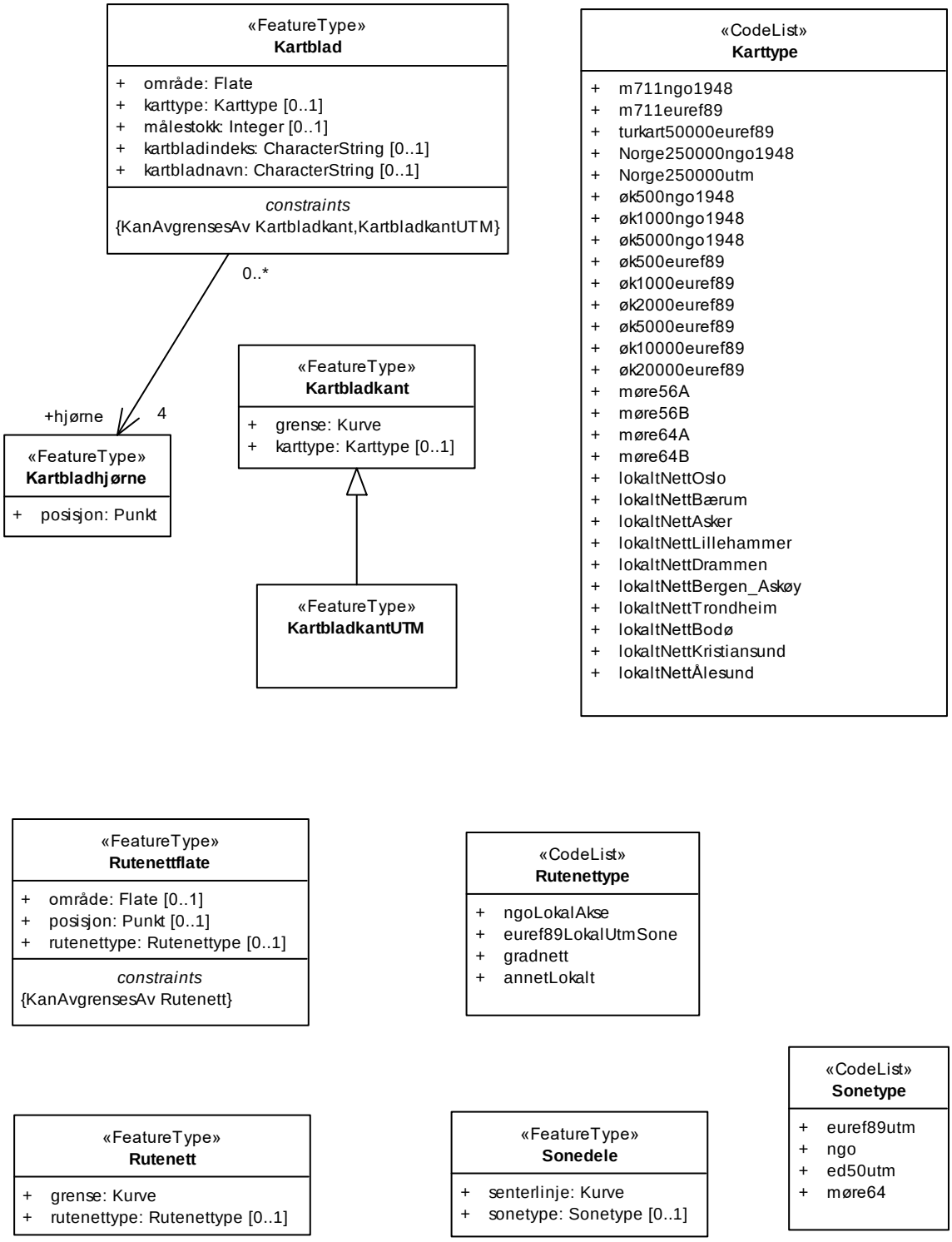
Mens et objekt i den virkelige verden kan avgrenses av objekttyper angitt i modellene for de respektive fagkapitler, vil en i et produkt ha muligheten til å innføre andre typer begrensninger. Eksempel på dette er kartbladkant og andre ulike typer avgrensninger. SOSI_Objekt hadde tidligere assosiasjoner til Kartbladkant, Dataavgrensning, FiktivDelelinje, Temakartavgrensning og KantUtsnitt. Objekttyper med flategeometri som ikke har modellert avgrensing via egne objekttyper med kurvegeometri må benytte egne mekanismer i formater der flateobjekttypene ikke kan bære sin egen geometri (Flateavgrensning i SOSI-format).



Figur 11.21 Avgrensningslinjer

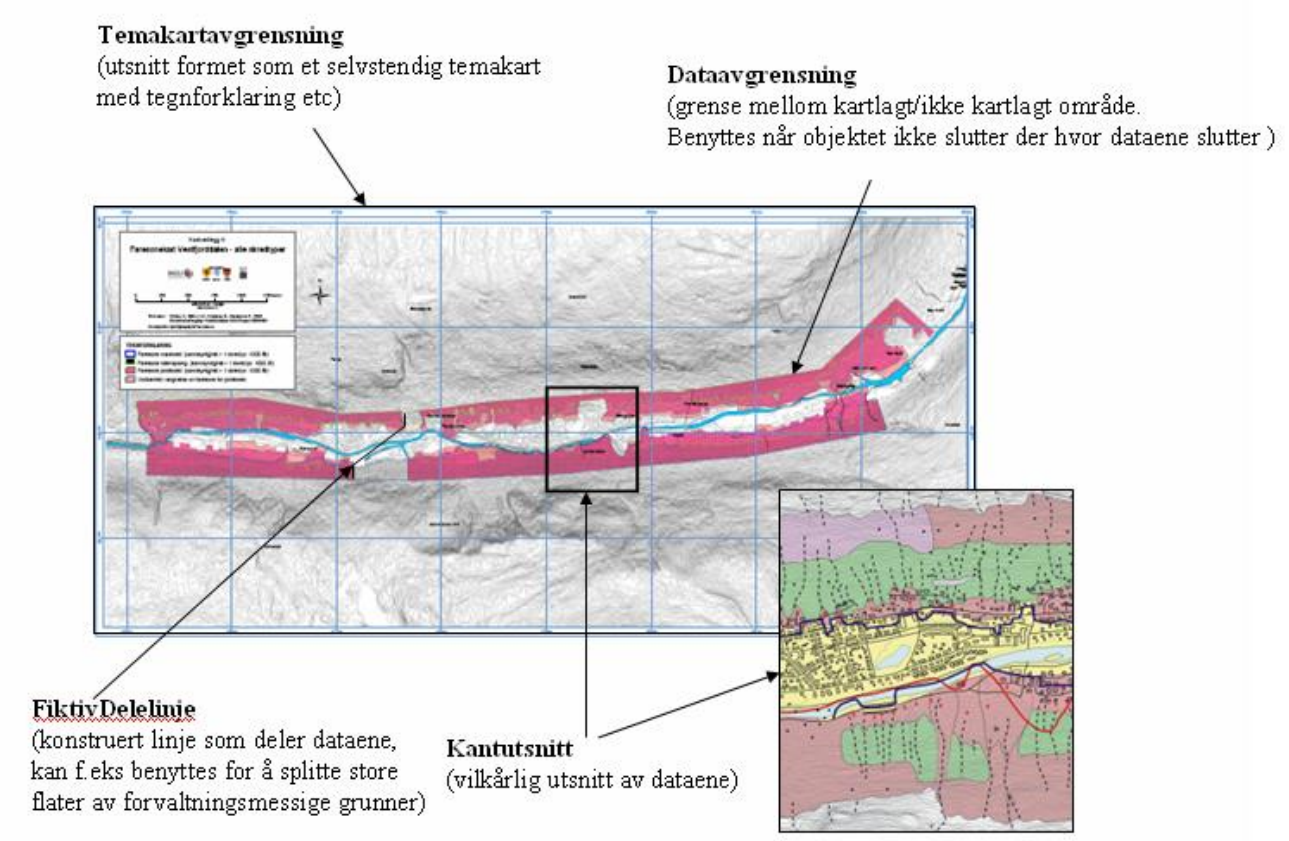
Tekstlig beskrivelse av UML-modellen er angitt i vedlegg E.3.

11.7.2 Kartbladkant/rutenett

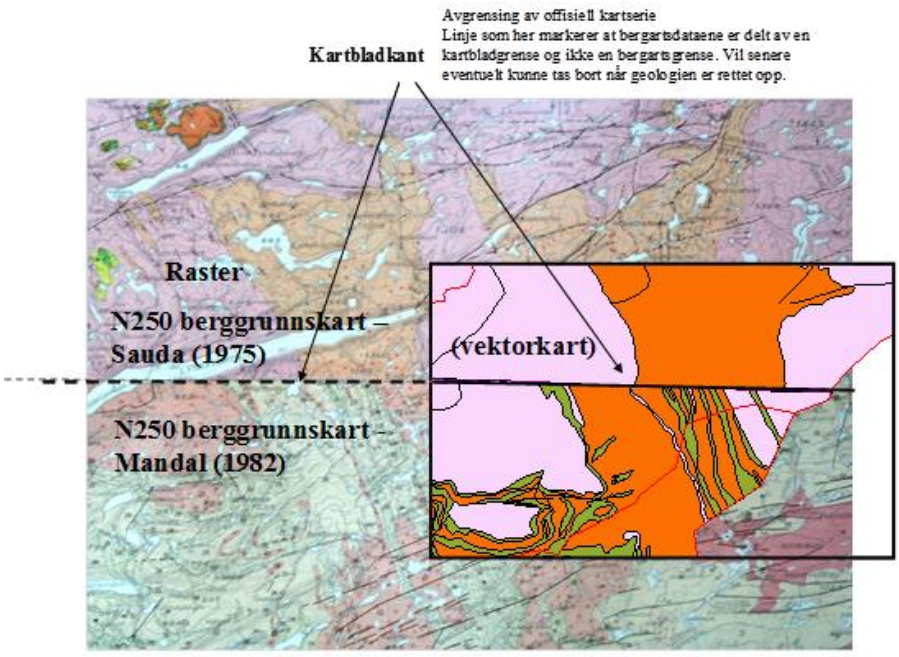


Figur 11.22 Kartblad og tilhørende objekttyper og egenskaper

Ved uttak av et tilfeldig utsnitt kan det være behov for å avgrense flateobjektene ved å bruke noen generelle avgrensningslinjer slik at man dokumenterer hvordan denne delen av flateavgrensingen er oppstått. Disse kan igjen være nødvendige for å bygge opp flater. Figur 11.23 viser noen eksempler på hvordan slike avgrensningslinjer kan benyttes.



Figur 11.23 Mulige avgrensningslinjer

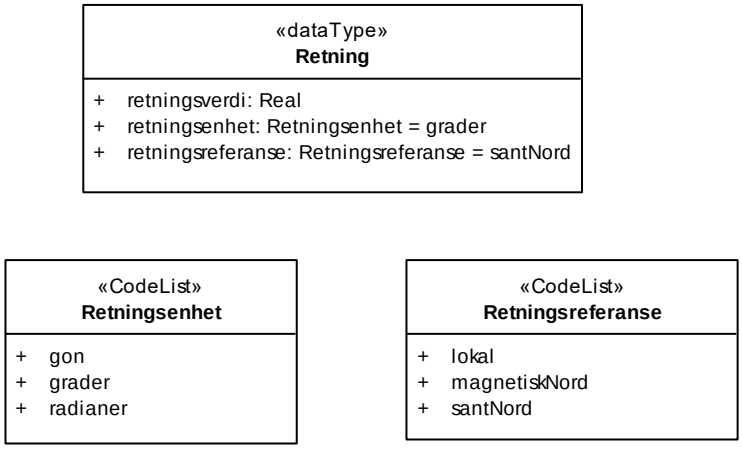


Figur 11.24 Kartbladkant som avgrensningslinje

/anbefaling/avgrensning	Denne standarden gir ingen krav til avgrensing av utsnitt ved klipping, men dersom dette ønskes anbefales det å bruke objekttypen KantUtsnitt som er angitt i dette kapittel i datautvekslingen, i henhold til E.3.1.

Tekstlig beskrivelse av UML-modellen er angitt i Vedlegg E.3.2

11.7.3 Retning

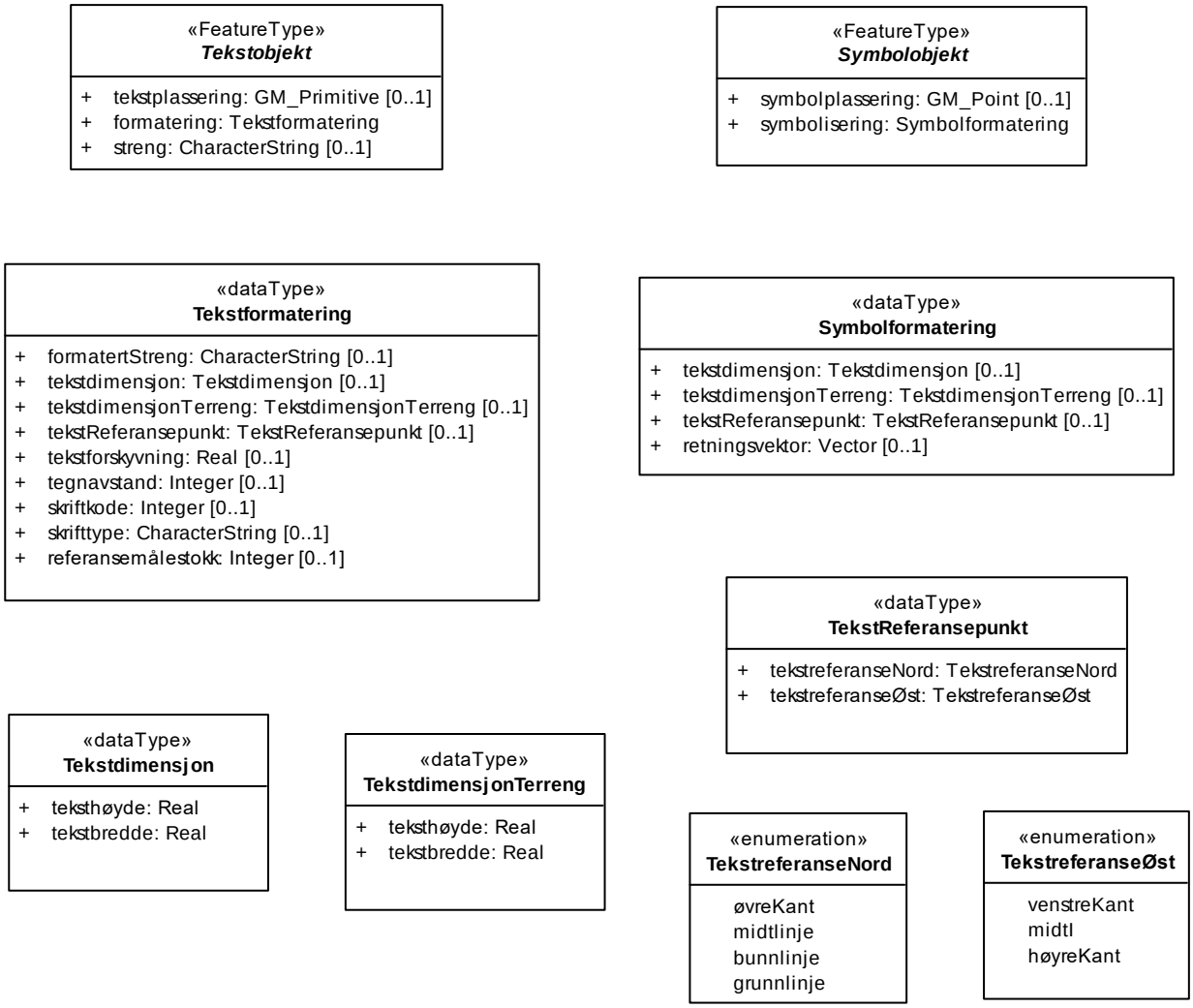


Figur 11.25 Retning

Tekstlig beskrivelse av UML-modellen er angitt i anneks E.3.3

11.7.4 Tekst, symbol og punkt med retning

For å kunne ta vare på data og muligheter fra eldre versjoner av SOSI-data er det laget to datatyper med egenskapene fra TEKST og SYMBOL og to abstrakte objekttyper for fast beskrivelse av hvordan geometriegenskapen skal brukes.

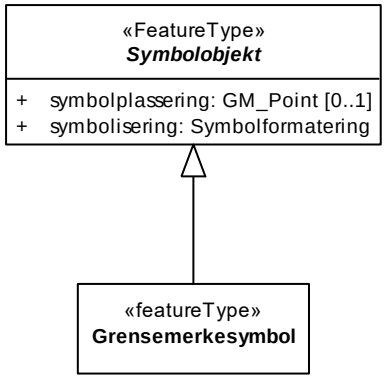
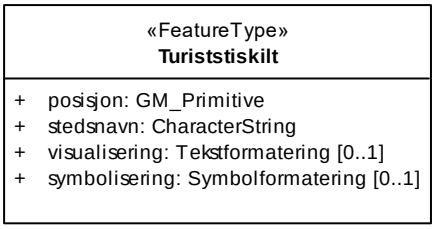


Figur 11.26 Tekst, symbol og punkt med retning

Tabell 11.27 er tre eksempel på bruk av datatypene.

Tabell 11.27 Eksempel på bruk av symbol og tekst.

subtyping av abstrakt tekstobjekt	«FeatureType» ***Tekstobjekt*** + tekstplassering: GM_Primitive [0..1] + formatering: Tekstformatering + streng: CharacterString [0..1] ↑ «FeatureType» **Høydetall**	Eksempel på arv av fast innhold for tekst og tekstplassering. Merk at den tidligere objektkoordinaten er utelatt.

subtyping av abstrakt symbolobjekt		Eksempel på arv av fast innhold for symbol og symbolplassering. Merk at den tidligere objektkoordinaten er utelatt.
egenmodellert symbol og tekst		Eksempel på friere bruk av datatypene Tekstformatering og Symbolformatering. Merk at objekttypens geometri er en vanlig standardtype, og at verdiene i datatypene må forholder seg til denne geometrien

11.8 Diagramregler

Dette kapittel gir en oversikt over ulike diagramtyper som er relevante til å bruke for visualisering av elementene i et applikasjonsskjema, relasjonene mellom disse elementene eller relasjoner til eventuelle elementer utenfor applikasjonsskjemaet.

/krav/visualisering	Alle klasser, egenskaper, assosiasjoner, assosiasjonsroller, operasjoner og restriksjoner i et applikasjonsskjema skal vises i minst ett av de følgende diagrammene: - hoveddiagram - oversiktsdiagram Avhengigheter mellom pakker skal vises i minst ett: - pakkeavhengighetsdiagram Dersom en realiserer pakker eller klasser i andre standarder skal dette vises i form av: - pakkerealiseringsdiagram - realiseringsdiagram

Formålet med visualiseringen er å vise informasjon i modellen på en strukturert og oversiktlig måte, men med så få diagrammer som mulig uten at sentrale modellelementer som er nevnt i kravet ovenfor utelates.

Se også ISO/TC 211 Best practise https://github.com/ISO-TC211/UML-Best-Practices. Her er det nyttig informasjon om god diagramdesign.

11.8.1 Pakkeavhengighetsdiagram

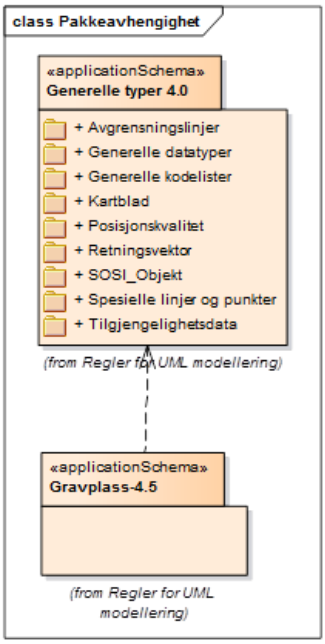
Pakkeavhengighetsdiagram viser avhengigheten mellom ulike applikasjonsskjemapakker som inngår i et UML-applikasjonsskjema. Dette er viktig med tanke på å vise hva de ulike konseptene er avhengige av. Se Kapittel 10.2.7, Krav 17.

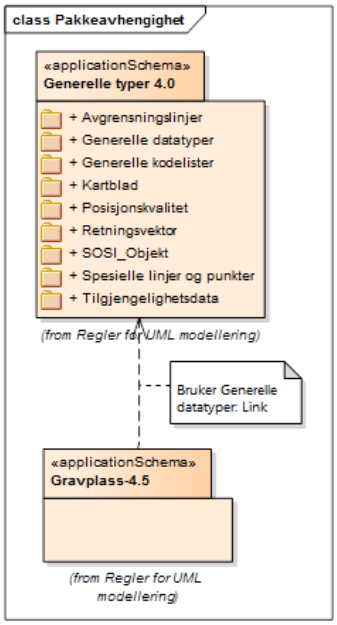
Det er ikke et krav å vise transitive avhengigheter, men det kan være hensiktsmessig i noen tilfeller.

/anbefaling/pakkeavhengighet/ fullStiTilPakke	Fullstendige referanser til hvor de ulike pakkene kommer fra bør vises i pakkeavhengighetsdiagrammet dersom dette er lesbart i figuren. Hvis figuren er så kompleks at det ikke er mulig å vise dette bør en vurdere å lage flere pakkeavhengighetsdiagram.
/anbefaling/pakkeavhengighet/navning	Det anbefales å navne pakkeavhengighetsdiagram etter følgende mønster: Pakkeavhengighet <navnet på applikasjonsskjema>.

/krav/pakkeavhengighet/ produktspesifikasjoner	Produktspesifikasjonsmodeller skal ikke ha eksterne avhengigheter og dermed ikke bruke pakkeavhengighetsdiagrammer.

Tabell 11.28 Eksempler på pakkeavhengighet

Eksempel 1: Pakkeavhengighet		Figuren viser et tenkt eksempel på et pakkeavhengighets-diagram med fullstendige pakkestier for SOSI Gravplass 4.5 som gjenbruker elementer fra Generelle typer 4.0

Eksempel 2: Pakkeavhengighet med tillegsinformasjon		Figuren viser et tenkt eksempel på et pakkeavhengighets-diagram med fullstendige pakkestier for SOSI Gravplass 4.5 som gjenbruker elementer fra Generelle typer 4.0, men med en ytterligere beskrivelse av hva som brukes. Denne tilleggsinformasjonen kan angis dersom den ikke overfyller diagrammet og ikke inkonsistent med det som faktisk brukes.

11.8.2 Hoveddiagram

Alle applikasjonsskjemapakker bør i utgangspunktet ha et hoveddiagram som oversiktlig og logisk viser alt innhold i alle objekttypene i pakka.
Assosiasjoner, assosiasjonsroller, kodelister og datatyper kan også vises i et hoveddiagram dersom det ikke blir uoversiktlig.

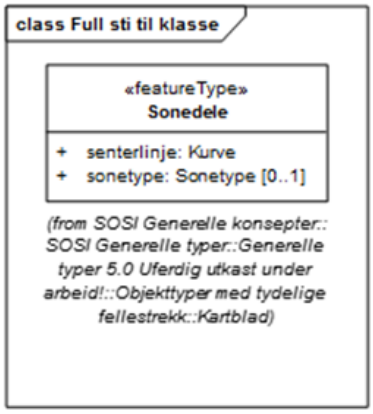
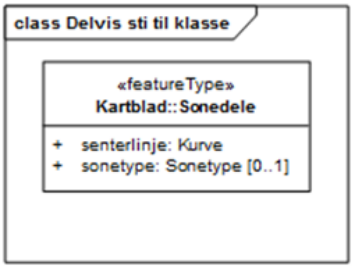
For en liten modell uten avhengigheter til eksterne pakker og uten realiseringer kan det være tilstrekkelig å visualisere all informasjon i ett enkelt hoveddiagram.

/krav/hoveddiagram	Alle objekttyper med alt innhold skal vises i ett eller flere hoveddiagrammer.
/krav/hoveddiagram/navning	Hoveddiagram skal navnes etter følgende mønster: Hoveddiagram <applikasjonsskjemanavn> Dette gjelder applikasjonsskjemaer der det bare finnes et hoveddiagram. Se krav /krav/hoveddiagram/detaljering/navning for navning av diagrammene der flere hoveddiagrammer er brukt.

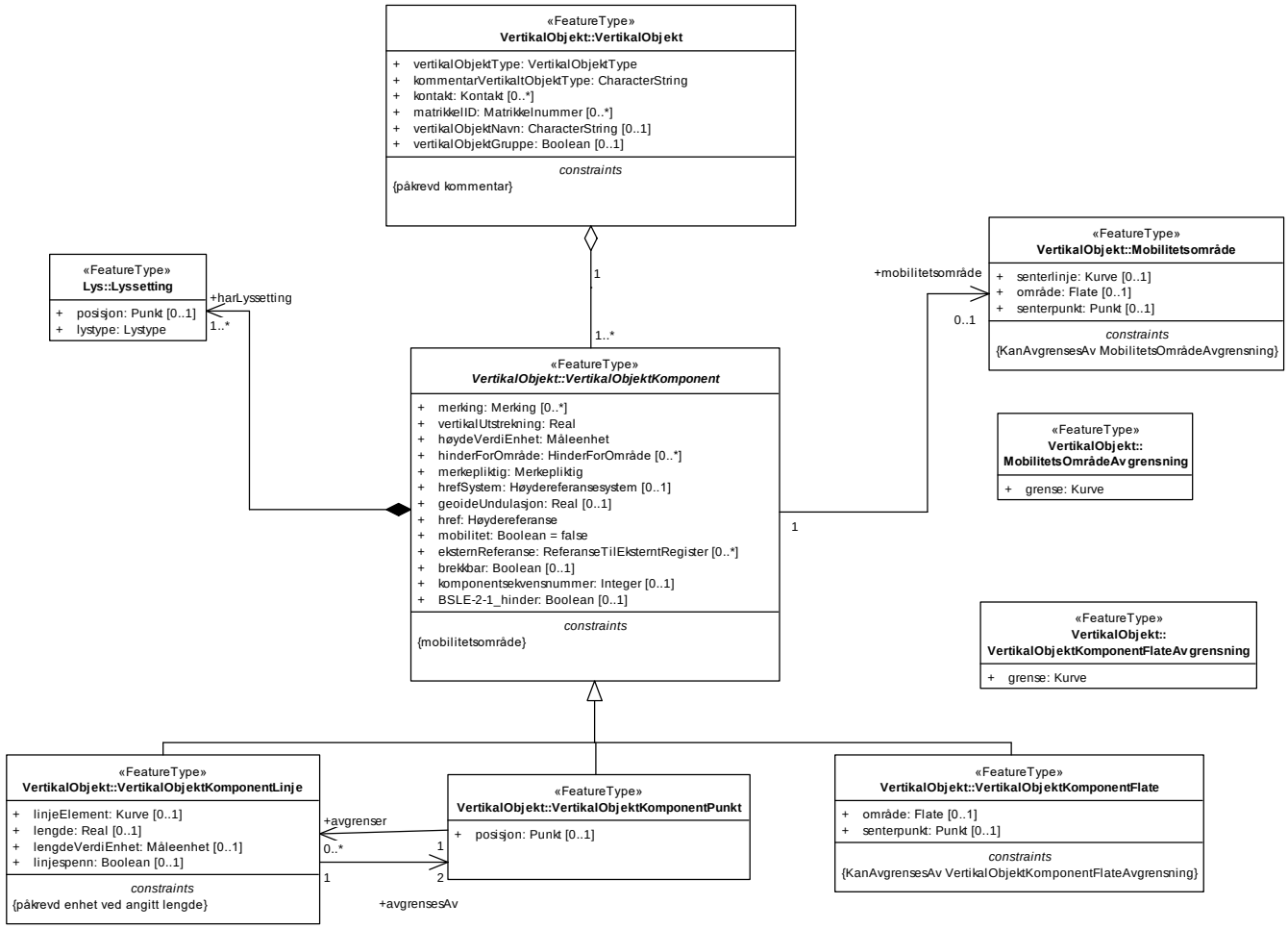
/anbefaling/hoveddiagram/ fullStiTilKlasse	Fullstendige referanser til hvor de ulike elementene kommer fra bør vises dersom dette er lesbart i figuren. Dette gjelder bare de modellelementer som er definert i andre modeller og gjenbrukes i UML-applikasjonsskjema.

Eksempel på full sti til klasse som finnes i en ekstern pakke er vist i Tabell 11.29.

Tabell 11.29 Eksempler på stier til klasse som finnes i eksterne pakke.

Full sti		Full sti til ekstern pakke (fully qualified tag)
Delvis sti		Eksempel på delvis sti til klasse som finnes i en annen (under-)pakke:

Figur 11.27 viser hoveddiagrammet for Luftfartshinder 4.5.1 som samler alle objekttyper i applikasjonsskjemaet og viser deres egenskaper, restriksjoner og assosiasjonene mellom dem.



Figur 11.27 Hoveddiagram Luftfartshinder-4.5.1

I noen tilfeller er modellene så kompliserte at det er vanskelig å vise alle detaljer i modellen i et enkelt hoveddiagram.

Modeller av applikasjonsskjema der objekttypene deles logisk i separate grupper (en enkelt eller et fåtall objekttyper) kan ha diagramstruktur som samsvarer med denne grupperingen.

Summen av alle slike diagrammer viser den fullstendige modellen.

Alle modellelementer (egenskaper og restriksjoner samt alle assosiasjoner som er navigerbare fra objekttypen(e)) skal framkomme i minst ett slikt diagram.

Der det finnes flere hoveddiagrammer, er de likestilte. Det ligger ingen hierarki i det.

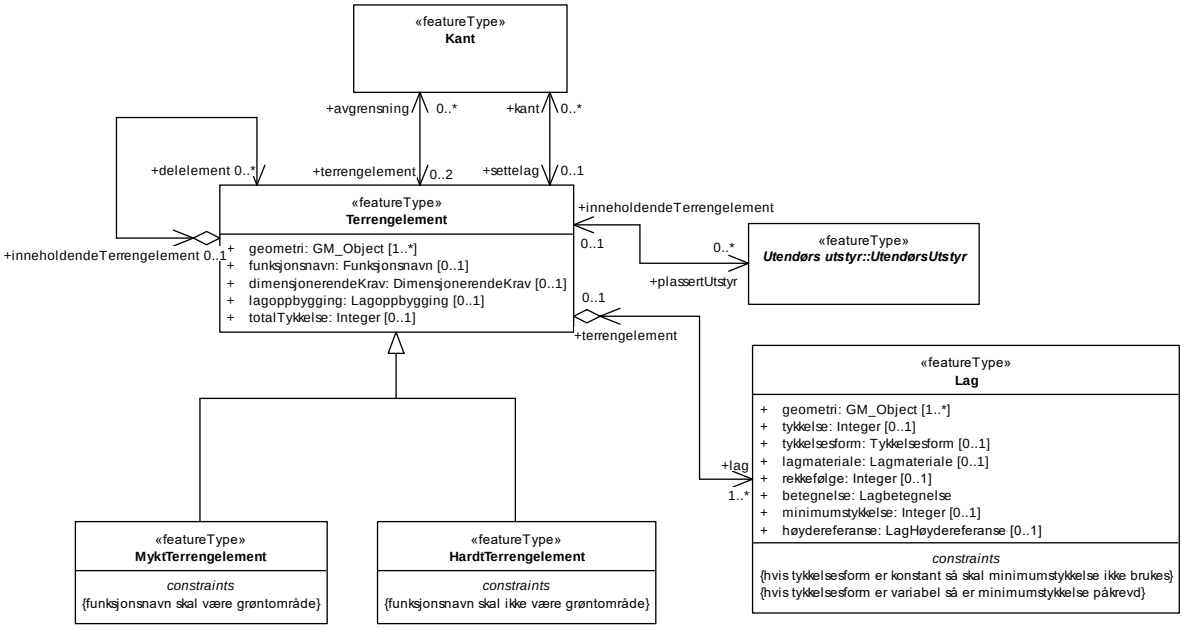
/krav/hoveddiagram/detaljering	Bruk ytterligere detaljering av hoveddiagram dersom det ikke er mulig å vise all informasjon i et enkelt hoveddiagram. Dersom hoveddiagrammet viser all informasjon trenger man ikke flere diagrammer utover dette.
/krav/hoveddiagram/detaljering/ navning	Der flere hoveddiagrammer er brukt for et applikasjonsskjema, navnes disse etter følgende mønster: Hoveddiagram <logisk gruppe> Som navn kan en bruke navnet til den sentrale objekttypen i diagrammet. Dersom diagrammet inneholder flere objekttyper kan en navngi etter hva disse omfatter.

Eksempel: Hoveddiagram Bygning og tak

Dersom en ønsker å samle kodelister og/eller datatyper i et eget diagram kan de navnes som f.eks:

- Kodelister og datatyper
- Kodelister
- Datatyper
- Kodelister Bygninger og Tak

Figur 11.28 viser hoveddiagrammet for objekttypen Terrengelement og Lag. Diagrammet er et av flere hoveddiagrammer i applikasjonsskjemaet Landskapsarkitektur og viser de sentrale objekttypenes (Terrengelement, Lag) egenskaper og alle assosiasjoner til andre objekttyper. De andre objekttypenes egenskaper, eller assosiasjoner mellom dem vises ikke i diagrammet.



Figur 11.28 Hoveddiagram Terrengelement og Lag

11.8.3 Oversiktsdiagram

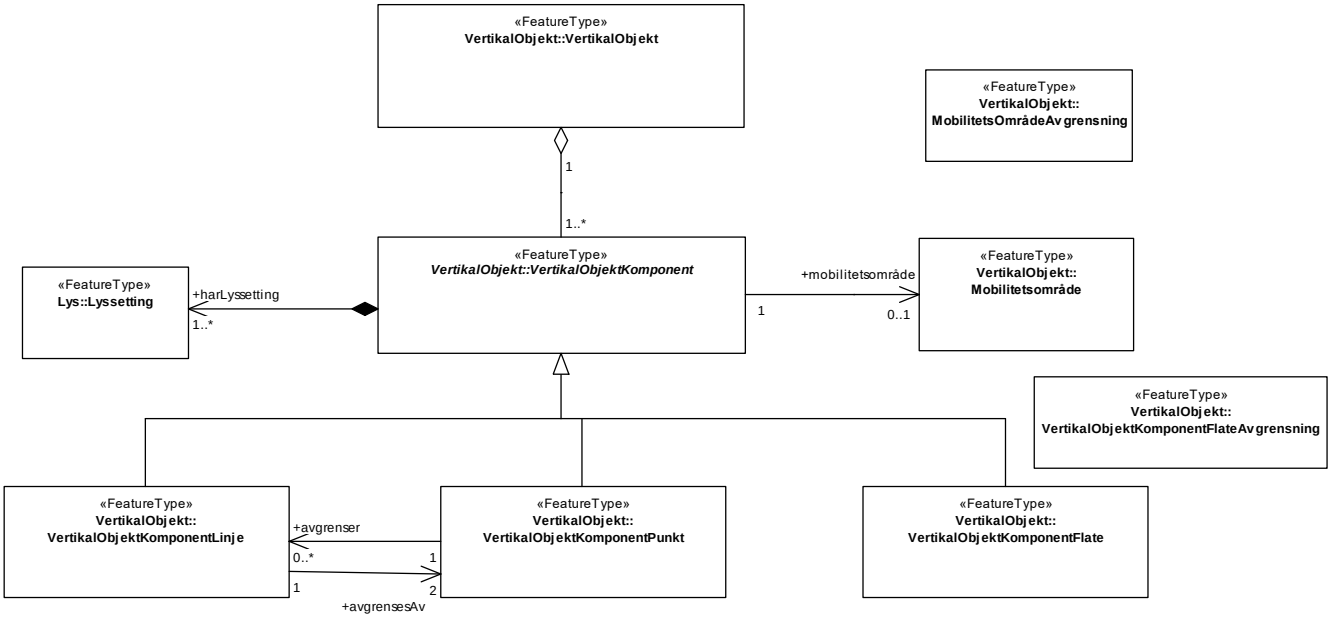
En UML-modell kan være kompleks med mange klasser, datatyper og kodelister samt assosiasjoner mellom klassene. I noen tilfeller er det ikke ideelt å vise alt innhold i modellen i en enkelt figur.

For å ha total oversikt over alle modellelementer en pakke inneholder, benyttes et oversiktsdiagram per pakke.

/krav/oversiktsdiagram	I de tilfeller der ikke alt i en pakke kan vises i et enkelt diagram skal det være et klassediagram for hele pakken, inkludert alle objekttypene (eventuelt med arv) og assosiasjoner mot andre klasser, med rollenavn. Visning av egenskaper, restriksjoner og selvassosiasjoner er ikke nødvendig i dette diagrammet. Dersom hoveddiagrammet viser all informasjon trenger man ikke et oversiktsdiagram.
/krav/oversiktsdiagram/navning	Oversiktsdiagram navnes etter følgende mønster: Oversiktsdiagram <navnet på pakke>

/anbefaling/oversiktsdiagram	Er modellelementene i en pakke fordelt over flere hoveddiagrammer, anbefales det å bruke et oversiktsdiagram i tillegg.

Figur 11.29 viser et eksempel på et oversiktsdiagram for Luftfartshinder 4.5.1 der alle objekttyper og assosiasjoner mellom dem vises mens egenskapene og restriksjoner ikke er en del av diagrammet.



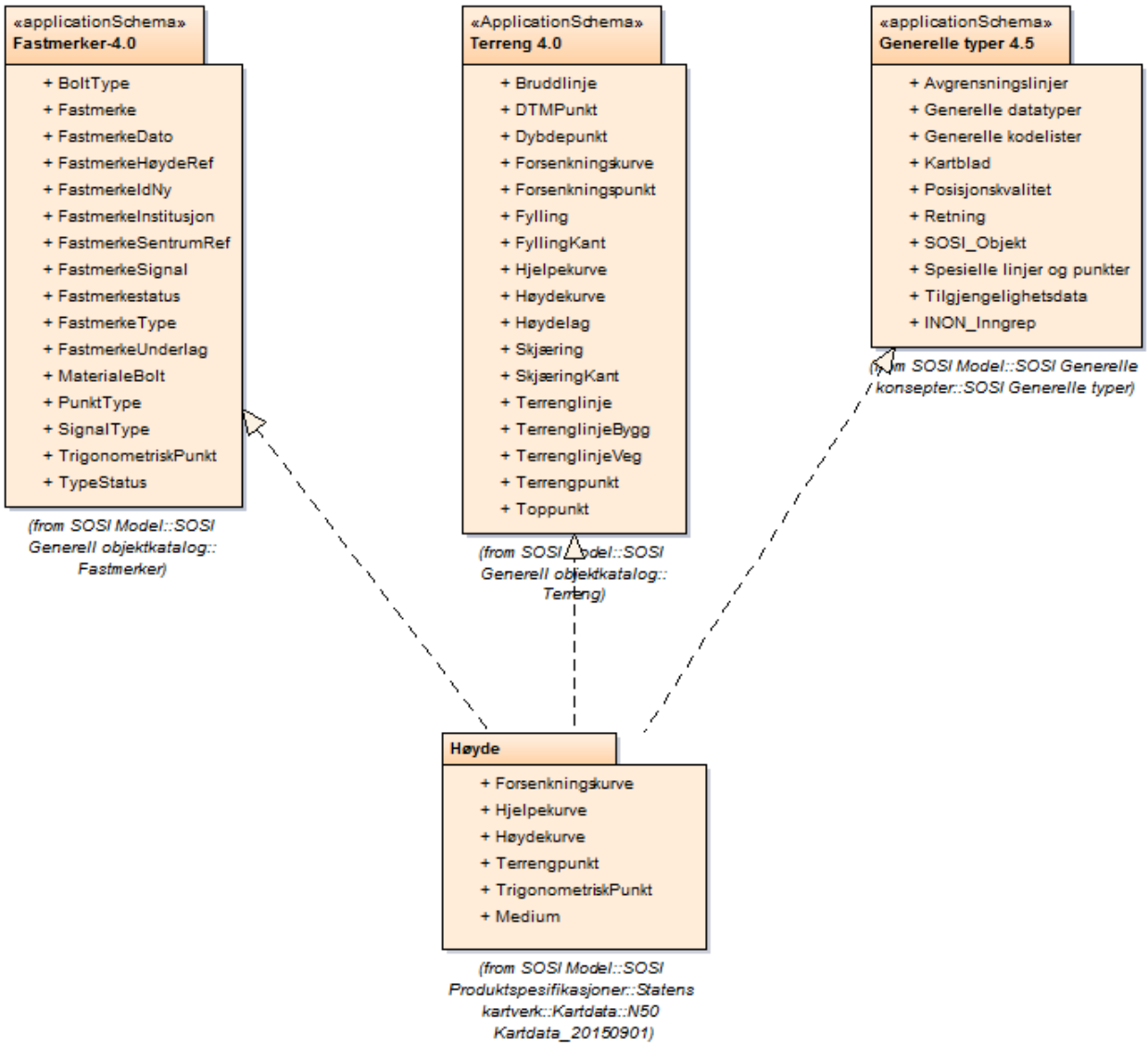
Figur 11.29 Oversiktsdiagram Luftfartshinder 4.5.1

11.8.4 Pakkerealiseringsdiagram

Et pakkerealiseringsdiagram viser pakker der en eller flere klasser har blitt realisert fra. Bruksområdet for slike diagrammer er hovedsakelig applikasjonsskjemaer for produktspesifikasjoner, men det kan også være aktuelt å bruke denne diagramtypen for fagområdestandardapplikasjonsskjemaer avhengig av hvilken variant (se kapittel D.1) man har valgt for å modellere med utgangspunkt i INSPIRE anneks I-III modellene.

/krav/pakkerealiseringsdiagram	Alle pakker som ligger i SOSI-modellregister og som inneholder klasser som er realiseringer av klasser i andre pakker i SOSI-modellregister skal vise dette i et pakkerealiseringsdiagram.
/krav/pakkerealisering/navning	Pakkerealiseringsdiagrammer skal navnes etter følgende mønster: Pakkerealisering <pakkenavn>

Figur 11.30 viser pakkerealiseringsdiagrammet for underpakke Høyde i N50 produktspesifikasjonsmodellen. Her ser man at klasser i pakka Høyde er realiseringer av klasser fra de tre applikasjonsskjemaer Fastmerker, Terreng og Generelle typer.



Figur 11.30 Pakkerealisering Høyde

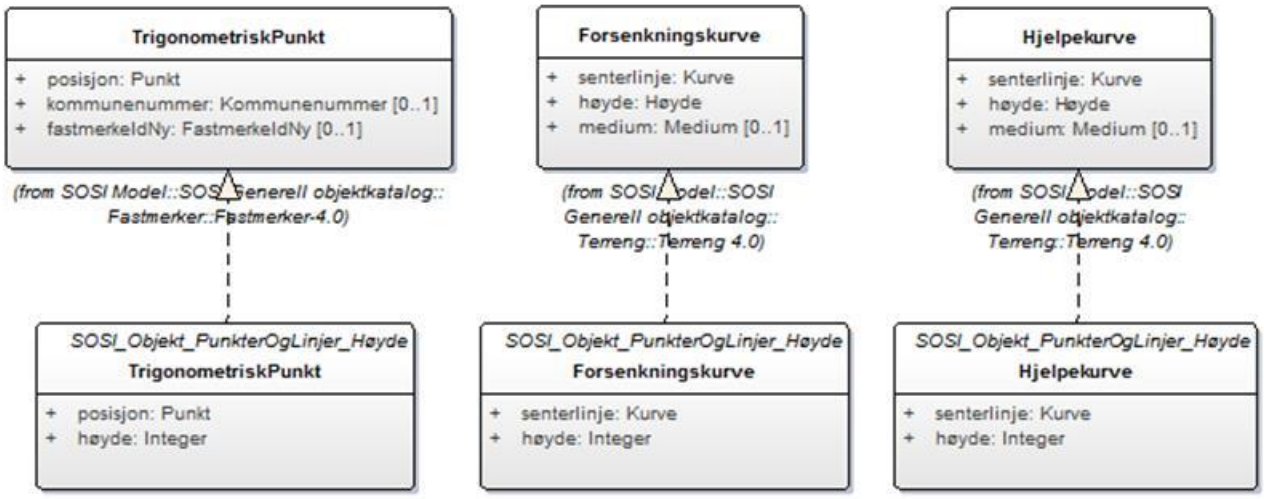
11.8.5 Realiseringsdiagram

Et realiseringsdiagram viser sammenheng mellom originalklasser fra andre applikasjonsskjemapakker og realiseringer slik at det er lett å se hvilke modifikasjoner man eventuelt har foretatt i den realiserte klassen sammenlignet med originalen. På samme måte som for pakkerealiseringsdiagrammer (se 11.8.4), gjelder for realiseringsdiagrammer at hovedbruksområdet er applikasjonsskjemaer for produktspesifikasjoner. Likevel kan det oppstå behov for denne diagramtype under arbeid med fagområdestandardapplikasjonsskjemaer dersom man realiserer elementer fra andre modeller, for eksempel fra INSPIRE anneks I-III modellene.

/krav/realiseringsdiagram	Alle klasser i et UML-applikasjonsskjema i SOSI-modellregister som er realiseringer av andre klasser skal vises i et eller flere realiseringsdiagram sammen med originalene for tydelig å se hva som er likt og hva som er endret. Det skal brukes realiseringspil fra realiseringsklassen til den originale klassen.
/krav/realiseringsdiagram/ navning	Realiseringsdiagram skal navngis etter følgende mønster: Realisering fra <navnet på kildeapplikasjonsskjema>

Vi fortsetter med samme eksempel fra 11.8.4 der pakkerealiseringsdiagrammet viser at Høyde-

pakka fra N50-produktspesifikasjonsmodellen inneholder realiserte klasser fra blant annet Fastmerker og Terreng. Figur 11.31 viser nå hvilke konkrete klasser dette gjelder. Det er tydelig at det ble gjort noen endringer på noen av egenskapene.



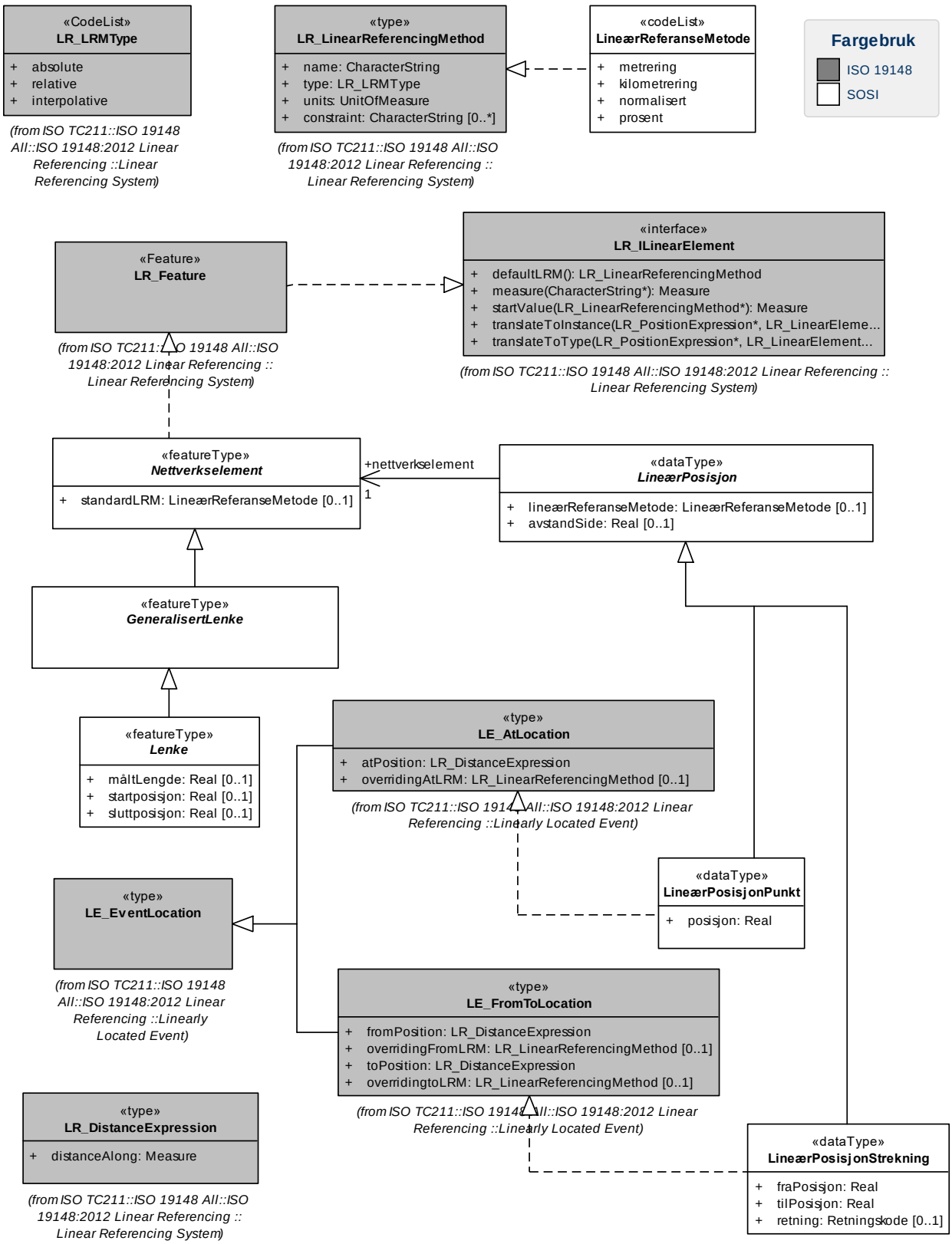
Figur 11.31 Utsnitt av realiseringsdiagrammet "Realisering fra Fastmerker og Terreng"

11.8.1 Bruk av farger i UML-modeller

UMLs metamodell legger ikke semantikk i farger. Denne standarden gjør heller ikke et forsøk på å standardisere hvilken betydning som evt. er tilknyttet hvilke konkrete farger i et diagram. Ulike brukergrupper kan derimot ha behov for fargebruk som forutsetter en felles tolkning innenfor denne gruppen. Tolkningen av farger kan være lik innenfor et fagdomene, en etat eller innenfor et prosjekt, men kan variere stort på tvers av disse. I slike tilfeller kan farger bidra til å øke den menneskelige forståelsen av diagrammene. Farger erstatter ikke fully qualified tag.

Krav/diagramfargebruk	Farger skal ikke inneholde standardisert semantikk som er avgjørende for modellen. Det er kun den grafiske UML-notasjonen som skal inneholde alt av modellens semantikk. Alle modeller kan benytte farger til blikkfang for spesielle elementer og som bakgrunnsfarger. Hvis en bruker farger skal en forklare fargebruken i diagrammet.

Eksempel: Figur 11.32 viser et eksempel på hvordan farger kan brukes i et diagram. Her brukes farger for å kunne skille lettere mellom elementene i SOSI-modellen for lineære referanser og klasser som er fra ISO 19148 modellen. En tegnforklaring gir oversikt hva de ulike fargene betyr.



Figur 11.32 Eksempel på bruk av farger i et diagram

11.9 Modellering av UML-applikasjonsskjema med utgangspunkt i Geodatalovens anneks I-III (INSPIRE)

Dette kapittel har til hensikt å beskrive hvordan de Europeisk harmoniserte modellene som Geodataloven peker på (INSPIRE Anneks I - III) kan være en kjerne i våre nasjonale spesifikasjoner. Dette kan gjøres på to ulike måter:

1. Ved å realisere INSPIRE objekter i våre nasjonale modeller. Dvs. en løselig kopling som kan suppleres med en mapping tabell.
2. Ved å subtype INSPIRE objekter samt legge til nasjonale modellelementer. Dette er en sterkere kopling som sikrer at nasjonale modeller utvikles i henhold til Geodatalovens intensjoner.

Ett av hovedmålene for SOSI er faglig fullstendighet – å dekke alle typer geografisk informasjon som er definert som viktig i infrastrukturen. Konsekvensen er at det både er og bør være overlapp mellom SOSI - og andre objektkataloger, ikke bare Geodatalovens anneks I til III. Men ulikheter i modeller er utfordrende og kompliserende for innsamling, forvaltning og utveksling av data, da informasjonen i begrenset grad kan gjenbrukes mellom objektkatalogene. Det er derfor ønskelig å ha mest mulig harmoniserte modeller og et mest mulig komplett felles modellregister. De reglene som beskrives her gjelder også for modellering av applikasjonsskjema med utgangspunkt i andre objektkataloger generelt.

/krav/objektkataloger	INSPIRE anneks I til III legges inn i SOSI-modellregister, enten som en kopi eller som lesetilgang til original. Andre objektkataloger kan legges inn ved behov og etter ønsker fra de som forvalter objektkatalogen. Disse modellelementene kan inngå i SOSI fagområdestandarder og SOSI produktspesifikasjoner, men kan ikke endres* da forvaltningsansvaret ligger hos andre.
/krav/INSPIRE_Tilnærming	I det videre arbeidet med SOSI fagområdestandarder skal forholdet til INSPIRE UML-modeller beskrives modelleringsmessig ved å bruke en av følgende metoder, der en slik tilnærming er hensiktsmessig: 1. realisere INSPIRE modell elementer ved bruk av <<realize>> samt tilhørende mappingtabell 2. subtype INSPIRE modeller med nasjonale utvidelser. Dersom det av faglige grunner er vurdert slik at INSPIRE ikke er relevant for et fagområde, skal dette angis i introduksjonen.

* Med endring menes en konseptuell endring i modellen. Innføring av alias og eventuelle tagged values på INSPIRE klassene er ikke å oppfatte som en konseptuell endring, men vil måtte legges inn på ny dersom INSPIRE modellene endres i regi av arbeidet med direktivet.

Eksempler på hvordan vi kan bruke INSPIRE modellelementer i våre modeller er vist i Vedlegg D.

12 Registre

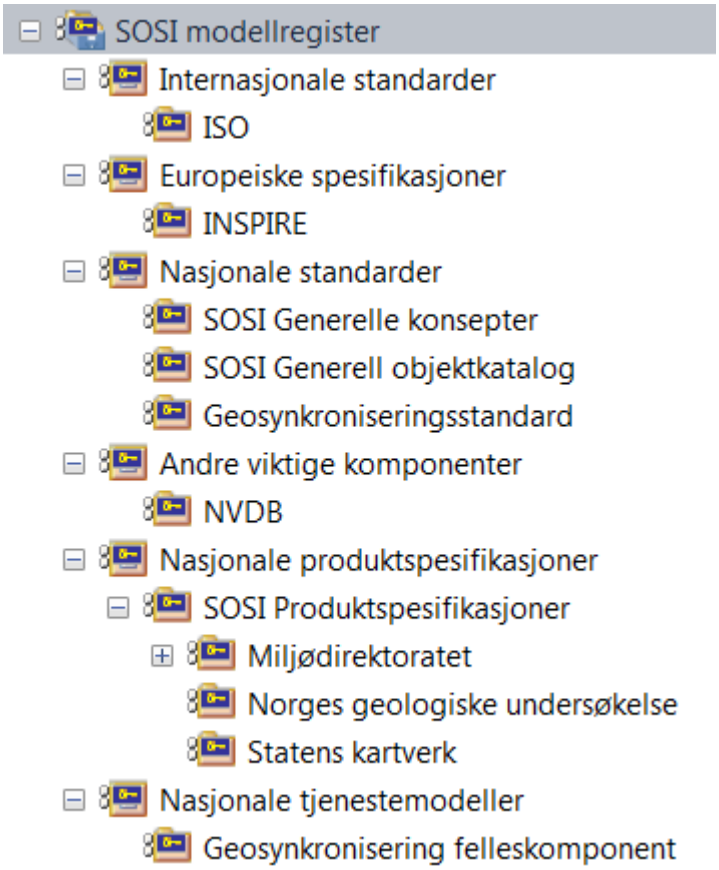
12.1 SOSI-modellregister

SOSI-modellregister er et SVN-basert register som er en sentral infrastrukturkomponent for SOSI-metoden. Det inneholder UML- og BPMN-modeller som er viktige for arbeidet med standarder og spesifikasjoner som er relatert til stedfestet informasjon. Alt innhold i SOSI-modellregister er åpent tilgjengelig og kan gjenbrukes uten innskrenkninger. Det oppfordres imidlertid til at opprinnelsen av modellelementer dokumenteres tilstrekkelig dersom originalene, eller varianter av dem gjenbrukes.

12.1.1 Innhold og struktur

Dette kapittelet gir en oversikt over innhold og pakkestrukturen i SOSI-modellregister.

Innholdet vil være en samling av UML og BPMN modeller fra relevante internasjonale standarder og spesifikasjoner samt nasjonale standarder, produktspesifikasjoner og tjenestemodeller.



Figur 12.1 Hovedpakkestruktur

Figur 12.1 er en skjermdump fra Enterprise Architect som viser et utdrag av hovedpakkestrukturen i SOSI-modellregister. Denne strukturen kan utvides etterhvert som nye behov oppstår (IHO, EMODnet).

For noen av de nevnte pakkene i Figur 12.1, som for eksempel SOSI Generelle konsepter, SOSI Generell objektkatalog og SOSI Produktspesifikasjoner, er SOSI-modellregister det primære lagringsstedet.

Disse pakkene vil brukerne av SOSI-modellregister kunne foreta endringer på forutsatt at pakken ikke er endelig vedtatt (og dermed låst), og at brukeren har skriverettigheter på den. Innhold i pakkene for INSPIRE og ISO er døde kopier av pakker forvaltet i egne registre til JRC.
Vedlegg D i denne standarden viser forskjellige måter å bygge på INSPIRE-modeller.

UML-applikasjonsskjemaer kan ha forskjellige statuser i de ulike fasene av livsløpet. Det begynner i utkastfasen og ender med at de blir erstattet av andre applikasjonsskjemaer, at de tilbaketrekkes eller blir erklært ugyldig. Statusene kobles til applikasjonsskjemapakkene i SOSI-modellregister ved hjelp av en tagged value SOSI_modellstatus. Tabell 12.1 viser hvilke statuser som er definert samt tilsvarende statuser i ISO 19135-1.

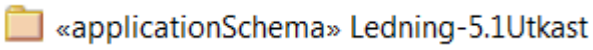
Tabell 12.1 Definisjoner av SOSI-modellstatusverdier og mapping til tilsvarende koder i ISO 19135-1

SOSI_modellstatus	Definisjon	ISO 19135-1
utkast	Modellen er under arbeid.	-
utkastOgSkjult	Modellen er under arbeid og innholdet skal ikke vises andre steder enn i SOSI-modellregister.	-
foreslått	Arbeid med modellen anses som avsluttet og den har blitt sent til godkjenning, men er foreløpig ikke godkjent.	submitted
gyldig	Modellen er godkjent. Den har ikke blitt erstattet av en annen modell og er heller ikke tilbaketrukket.	valid
fagligGodkjent	Modellens faglige innhold er godkjent av faginstans, men har ikke gjennomgått formell godkjenning.	-
standardisert	Modellen er godkjent etter en åpen høringsprosess.	-
erstattet	Modellen har blitt erstattet av en annen modell og er ikke lenger anbefalt å bruke.	superseded
tilbaketrukket	Modellen er ikke lenger anbefalt å bruke, men har ikke blitt erstattet av en annen modell.	retired
ugyldig	Modellen var gyldig tidligere, men inneholder en substansiell feil og vil vanligvis være erstattet av en korrigert modell.	invalid

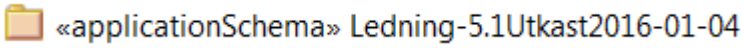
Modellene som har SOSI-modellregister som primært lagringssted må oppfylle noen krav når det gjelder pakkenavn og innhold i pakkene.
Nedenfor er vinkelparenteser (<>) brukt som plassholder, og vinkelparentesene med plassholder skal byttes ut med noe meningsfult. Hakeparenteser ([]) indikerer valgfrie elementer.

/krav/SOSI-modellregister/applikasjonsskjema/standard/pakkenavn/utkast	Modeller som er under arbeid skal ha "Utkast" som et midlertidig tillegg til pakkenavnet etter følgende mønster: <pakkenavnet>Utkast Elementer i pakkenavnet er definert lenger nede i dette kapittelet. NB: I noen tilfeller kan det være nyttig å legge til en dato. Dersom dette er ønskelig skal det skje etter følgende mønster: <pakkenavnet>Utkast<dato> Dato er sammensatt av ti tegn: <åååå>-<mm>-<dd> Her er det to sifre for dag (dd), to sifre for måned (mm) og fire sifre for år (åååå). Datoformatet er i henhold til ISO 8601:2004 kapittel 4.1.2.2 Complete representations.

Eksempel: Figur 12.2 og Figur 12.3 viser navn til pakker som er under arbeid. Navnene følger reglene som er definert i kravet ovenfor.



Figur 12.2 Eksempel på navnet til en pakke som er under arbeid



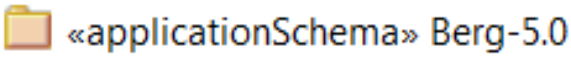
Figur 12.3 Eksempel på navnet til en pakke som er under arbeid med dato

/krav/SOSI-modellregister/applikasjonsskjema/status	Pakker laget etter SOSI-regler og med stereotype ApplicationSchema skal ha en tagged value SOSI_modellstatus. Verdien i denne tagged value kan være: • utkast • utkastOgSkjult • foreslått • gyldig • fagligGodkjent • standardisert • erstattet • tilbaketrukket • ugyldig Se Tabell 12.1 for definisjoner av de ulike statusene.
/krav/SOSI-modellregister/applikasjonsskjema/pakkenavn/vedtatt	Pakker med stereotype ApplicationSchema fra vedtatte SOSI-fagområdestandarder og SOSI-produktspesifikasjoner skal bruke følgende format for pakkenavnet: <fagområde>[<leveranse>][<målgruppe>]-<versjonsnummer> Pakker fra vedtatte SOSI-fagområdestandarder skal ikke ha leveranse eller målgruppe i pakkenavnet.

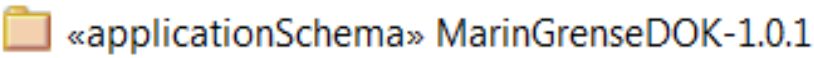
Eksempel: Figur 12.4 Eksempel på navnet til en pakke som er under arbeid med dato viser hvordan pakkenavnet til en vedtatt SOSI-fagområdestandard kan se ut. Berg er her fagområdet og 5.0 er versjonsnummeret.

Figur 12.5 viser et pakkenavn for en vedtatt SOSI-produktspesifikasjon med elementene fagområde (MarinGrense), leveranse (DOK = det offentlige kartgrunnlaget) og

versjonsnummer (1.0.1). NB: Figur 12.4 Eksempel på navnet til en pakke som er under arbeid med dato
kan også tolkes som en pakke til en SOSI-produktspesifikasjon siden leveranse eller målgruppe ikke er en påkrevd del av pakkenavnet.

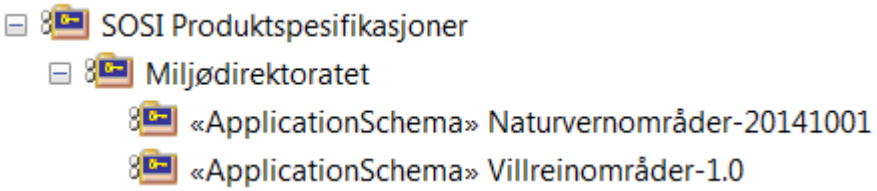


Figur 12.4 Eksempel på navnet til en pakke som er under arbeid med dato



Figur 12.5 Eksempel på pakkenavn til en vedtatt SOSI-produktspesifikasjon

/krav/SOSI-modellregister/ applikasjonsskjema/navneregler	Pakkenavn til applikasjonsskjema skal følge NCname-regler som er beskrevet i kapittel 10.2.2. Filnavn uten suffiks skal være identiske med pakkenavnet. Tagged value SOSI_kortnavn skal være identisk med pakkenavnet uten versjonsnummer. Hvis tittelen til det offisielle dokumentet er noe annet enn SOSI_kortnavn, skal tittelen lagres i tagged value SOSI_langnavn.
/krav/SOSI-modellregister/ applikasjonsskjema/versjonsnummer	Navnet til en pakke med stereotype ApplicationSchema og SOSI_modellstatus=gyldig i SOSI-Modellregister skal ha applikasjonsskjemaets versjonsnummer som siste element. Format: <navn>-<versjonsnummer> Et versjonsnummer kan være et løpenummer eller en dato. Se eksempel i Figur 12.6 nedenfor.



Figur 12.6 Eksempel på pakkenavn med versjonsnummer

/krav/SOSI-modellregister/applikasjonsskjema/navnerom	Pakker med stereotype ApplicationSchema fra vedtatte SOSI fagområdestandarder og SOSI produktspesifikasjoner skal bruke følgende format for tagged value targetNamespace: <URL>/<fagområde>[<leveranse>][<målgruppe>]/<versjonsnummer> Pakker fra vedtatte SOSI fagområdestandarder skal ikke ha leveranse eller målgruppe i pakkenavnet. NB: Dersom versjonsnummeret består av tre deler, der siste delen er build-nummeret, så skal build-nummeret ikke være en del av navnerommet. Eksempel 1: En pakke med versjonsnummer 3.0.1 skal ha navnerom med 3.0 som siste del. Eksempel 2: En pakke med versjonsnummer 2.1 skal ha 2.1 som siste del av navnerom.
/krav/SOSI-modellregister/konformitet/applikasjonsskjema	Alle applikasjonsskjemaene som har SOSI-modellregister som primær lagringsplass må oppfylle relevante krav fra kapittel 10 og 11.
/krav/SOSI-modellregister/konformitet/tjenestemodeller	Alle tjenestemodellene som har SOSI-modellregister som primær lagringsplass må oppfylle relevante krav fra kapittel 9, 10 og 11.

Selv om modellstatusen kan endre seg over tid, skal XMI-filnavnet være konstant gjennom hele livsløpssyklus til enhver pakke i SVN. Utgangspunktet for valg av filnavnet er applikasjonsskjemaets navn samt versjonsnummeret.

Det er en intensjon om at SOSI-modellregisteret skal bli det foretrukne modellregisteret for geografisk informasjon i Norge. Dette vil være en forutsetning for å kunne effektivt harmonisere elementer med fellestrekk i forskjellige modeller.

/anbefaling/SOSI-modellregister/innhold	Norske UML og BPMN-modeller for geografisk informasjon bør gjøres tilgjengelige i SOSI-modellregister.

SOSI-modellregister skal være et frivillig tilbud for lagring av UML og BPMN-modeller for geografisk informasjon. Det finnes likevel en gruppe modeller som har det som et krav at de tilgjengeliggjøres i SOSI-modellregisteret.

/krav/SOSI-modellregister/innhold	SOSI-modellregister skal være primært lagringssted for følgende applikasjonsskjemaer: - applikasjonsskjemaer for SOSI Generell del - applikasjonsskjemaer for SOSI Generell objektkatalog - applikasjonsskjemaer for SOSI Produktspesifikasjoner - applikasjonsskjemaer for eldre produktspesifikasjoner til datasett som er en del av DOK datasettene

12.1.2 Tilgang til innhold og struktur i SOSI-modellregister

Innholdet i SOSI-modellregister består av xmi-filer (én xmi-fil per versjonskontrollert pakke) og er tilgjengelig ved hjelp av diverse programvarer. En mulig konfigurasjon beskrives i

veilederdokumentet "Installasjon av nødvendig programvare for arbeid med SOSI-produktspesifikasjoner".

Det er også mulig å visualisere innhold i modellregisteret ved hjelp av andre innsynsløsninger. Et eksempel på det er webinnsyn til objektkatalogen via portalen Geonorge.

12.1.3 Nedlasting av modeller fra SOSI-modellregister

Ved nedlasting av et applikasjonsskjema fra modellregisteret er det viktig å også laste ned de pakkene som det er en avhengighet til. Dette fremkommer i applikasjonsskjemaets pakkeavhengighetsdiagram. Nærmere beskrivelse er angitt i veiledningsmateriell på https://kartverket.no/geodataarbeid/Standarder/SOSI/Retningslinjer-og-veiledere-SOSI-produktspesifikasjoner/

12.2 Register over kodelister

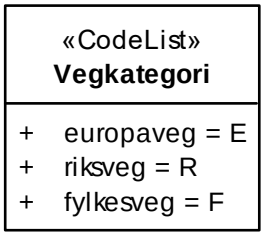
12.2.1 Introduksjon

En **kode** er et element i en kodeliste og kan bestå av et navn (**kodens navn**), en definisjon, opsjonelt en initialverdi (benyttes som utvekslingsalias), tagged values og restriksjoner.

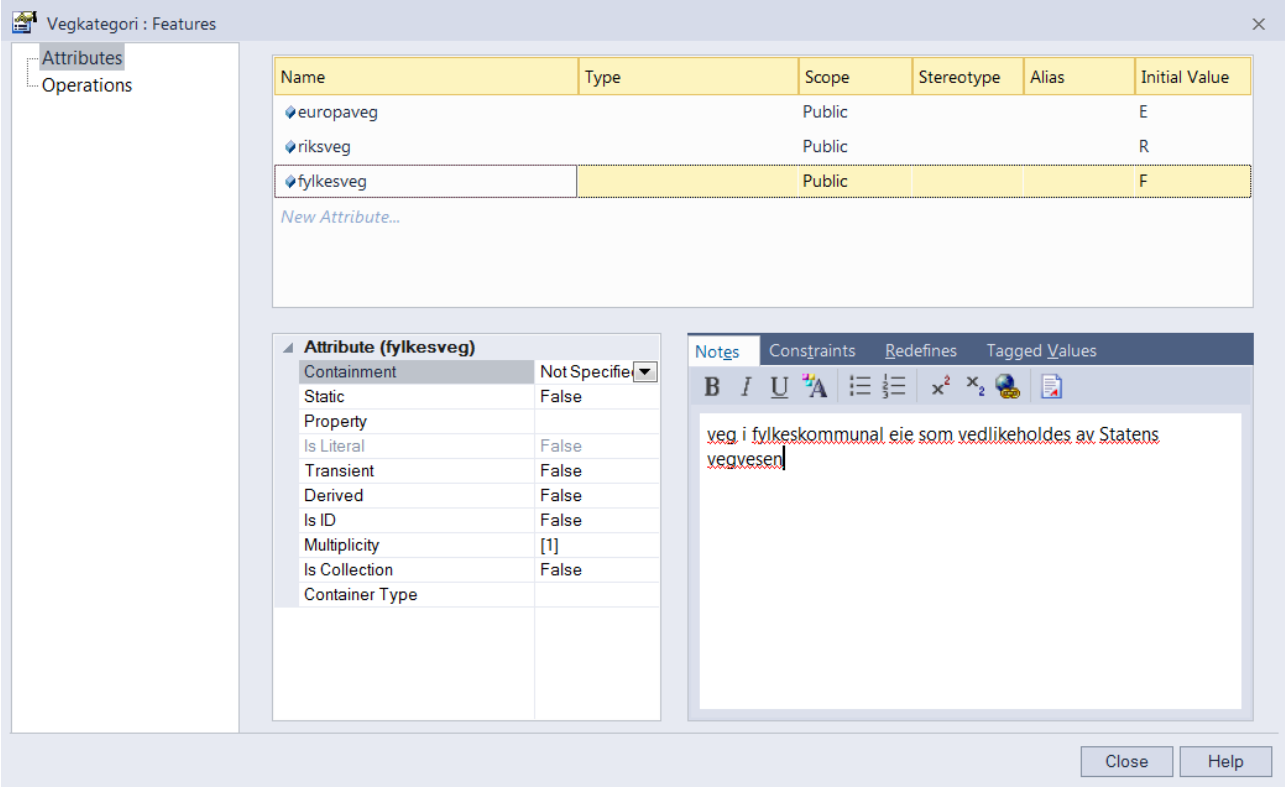
ISO 19100-serien har ikke definert initialverdier som en del av koder eller kodelister i plattformuavhengige modeller på et konseptuelt nivå. I UML-verktøy er kodelister klasser og klasser kan ha egenskaper med initialverdier. En initialverdi til en egenskap gir føringer for hvordan den første verdien til egenskapen skal være når klassen som egenskapen tilhører instansieres dersom det ikke er spesifisert på en annen måte. Denne verdien kan endres under objektets livsløp i motsetning til hvordan det er tenkt å bruke initialverdien til en kode. En kodes initialverdi er statisk og permanent koblet til koden. I kontekst av en kodeliste er initialverdi en misvisende betegnelse, den brukes her som et alternativ til kodens navn.

Korrekt bruk av initialverdier vises i kapittel 11.7.3 der for eksempel egenskapen retningsenhet har initialverdi "grader". Denne verdien er tatt fra en kodeliste som har flere koder som alle er lovlige verdier for egenskapen retningsenhet og som kan erstatte initialverdi "grader" på instansnivå.

> *Eksempel: Figur 12.7 viser en kodeliste med vegkategorier der hver vegkategori har en kode. Koden inkluderer alt som står i en linje, altså for eksempel "+ europaveg = E" der europaveg er kodens navn og E er kodens initialverdi. Figur 12.8 viser koden fylkesveg. Her ser vi at koden kan ha flere elementer enn de som vises i Figur 12.7. Den er sammensatt av elementene navn "fylkesvei" (**kodens navn**) i Name-feltet, definisjon "veg i fylkeskommunal eie som vedlikeholdes av Statens vegvesen" i Notes-feltet og en alternativ verdi "F" (**initialverdi**) i Initial Value-feltet. Koden kan ha flere elementer som for eksempel restriksjoner i Constraints-fanen eller tagged values i Tagged Values-fanen. Innhold i disse vises ikke her, men fanene er plassert ved siden av Notes-fanen.*



Figur 12.7 Eksempel på en kodeliste med initialverdier



Figur 12.8 Eksempel på elementer en kode kan være sammensatt av (fra UML-verktøy Enterprise Architect 12.0)

En **kodeliste** er et sett med koder (termer) som utgjør lovlige verdier for en egenskap. Kodelister kan opptre som uavhengige sett med koder, men i mange tilfeller er gyldigheten for koder i en kodeliste avhengig av verdien koder i en annen kodeliste – dvs. kodelister kan ha hierarkiske avhengigheter eller ha flere ulike grupperinger. I tillegg kan koder ha synonymer.
Forvaltning av koder innebærer at over tid kan koder merkes som utgått og bli erstattet av en eller flere nye. Koder i datasett vil derfor forholde seg til det som var definert for koden akkurat på registreringstidspunktet, og ikke til hva som er beskrivelser før og etter dette tidspunktet.

Interne kodelister er kodelister som har noen koder definert i modellen men nye koder kan i utgangspunktet fritt legges til senere av brukeren selv. Disse nye kodene bør ikke ha overlappende betydning med de kodene som er i modellen. Men to ulike brukere som legger til nye koder vil ikke kunne være sikre på å unngå overlapp i betydning seg imellom.

Eksterne kodelister er kodelister som forvaltes i egne datasystemer (koderegistre) og kan kun endres av forvaltere som er autorisert for dette. Modellen kan inneholde de kodene som var gyldige på modelleringstidspunktet, men tillegg kan ha kommet til senere. Slike kodelister kan også være tomme i modellen (som INSPIRE). Et godt prinsipp i koderegistere er å ikke gjenbruke koder til andre betydninger, men kun merke koder som utgått eller utvide med koder med ny betydning.

Eksterne kodelister skal merkes med en tagged value asDictionary = true. De skal også ha en tagged value codeList = <http-URI som identifiserer hvor kodelisten forvaltes>.

Denne identifikatoren peker på ressurs som definerer kodene. Den samme ressursen kan støtte flere formater ved å bruke mediatyper.

> *Eksempel fra INSPIRE: Kodelista NamedPlaceTypeValue finnes som html på URL'en http://inspire.ec.europa.eu/codelist/NamedPlaceTypeValue/*
>
> *Hvis en ønsker kodelista som en annen medietype, for eksempel RDF, finnes den ved å legge til URN'en NamedPlaceTypeValue.en.rdf, med andre ord http://inspire.ec.europa.eu/codelist/NamedPlaceTypeValue/NamedPlaceTypeValue.en.rdf*

Koder fra eksterne kodelister bør i størst mulig grad vises i UML-applikasjonsskjema på samme måte som interne kodelister. Spesielt viktig er dette dersom elementer fra eksterne applikasjonsskjema (som f.eks. INSPIRE), er inkludert i det nasjonale skjema ved hjelp av subtyping. Den lokale kopi kan dermed oversettes til norsk og gjøres tilgjengelig for dataprodusenter og brukere av produktspesifikasjonene. Om det er mulig bør man slippe å forholde seg til/slå opp i eksterne registertjenester. Oppdatering av de eksterne kodeverdiene i det nasjonale skjema må da rutinemessig utføres av ansvarlig organisasjon for applikasjonsskjemaet/produktspesifikasjonen.

12.2.2 Krav og anbefalinger til kodelister

/krav/internKodeliste	Interne kodelister (uten tagged value asDictionary = true) skal ikke utvides med koder som har samme betydning som noen av de eksisterende.

/anbefaling/internKodelisteLåst	Interne kodelister kan låses mot utvidelse ved at man modellerer dem som klasser med stereotype enumeration.

/krav/eksternKodeliste	Eksterne kodelister (med tagged value asDictionary = true) skal ha en tagged value codeList med verdi som er en http-URI som identifiserer hvor man kan få tilgang til kodene.

> *Eksempel på http-URI til nettsted der koden er definert:*
> *http://inspire.ec.europa.eu/codelist/NamedPlaceTypeValue/landform*

/anbefaling/eksternKodelisteTjeneste	Eksterne kodelister (med tagged value asDictionary = true) som ligger i et eget register anbefales å identifisere en egen tjeneste som beskriver de aktuelle kodene.
/anbefaling/eksternKodelisteMediatype	Dersom eksterne kodelister (med tagged value asDictionary = true) som ikke ligger i et eget register anbefales å la brukere angi ønsket format ved å angi foretrukken mediatype.
/anbefaling/eksternKodelisteSKOS	Eksterne kodelister (med tagged value asDictionary = true) som ikke ligger i et eget register anbefales å identifisere en standard SKOS RDF/xml-fil som beskriver de aktuelle kodene.

Standarder innen semantisk web har modnet betraktelig de senere år. "Best practice" i dag er å bruke http-URI'er for å referere til vokabularene og RDF (SKOS) for å kode deres beskrivelse. Denne standarden gir ikke føringer for hvilke verktøy som brukes.

Eksempel på http-URI til koder i SKOS-fil:
http://inspire.ec.europa.eu/codelist/NamedPlaceTypeValue/landform/landform.en.rdf

Mediatype for SKOS/RDF/xml-kodelister er *application/rdf+xml,*

/anbefaling/eksternKodelisteGML	Dersom eksterne kodelister ikke kan benytte SKOS RDF/xml-fil så anbefales det å identifisere en gml:Dictionary xml-fil som beskrevet i GML-standarden (ISO 19136-2:2015 eller GML 3.3 fra OGC).

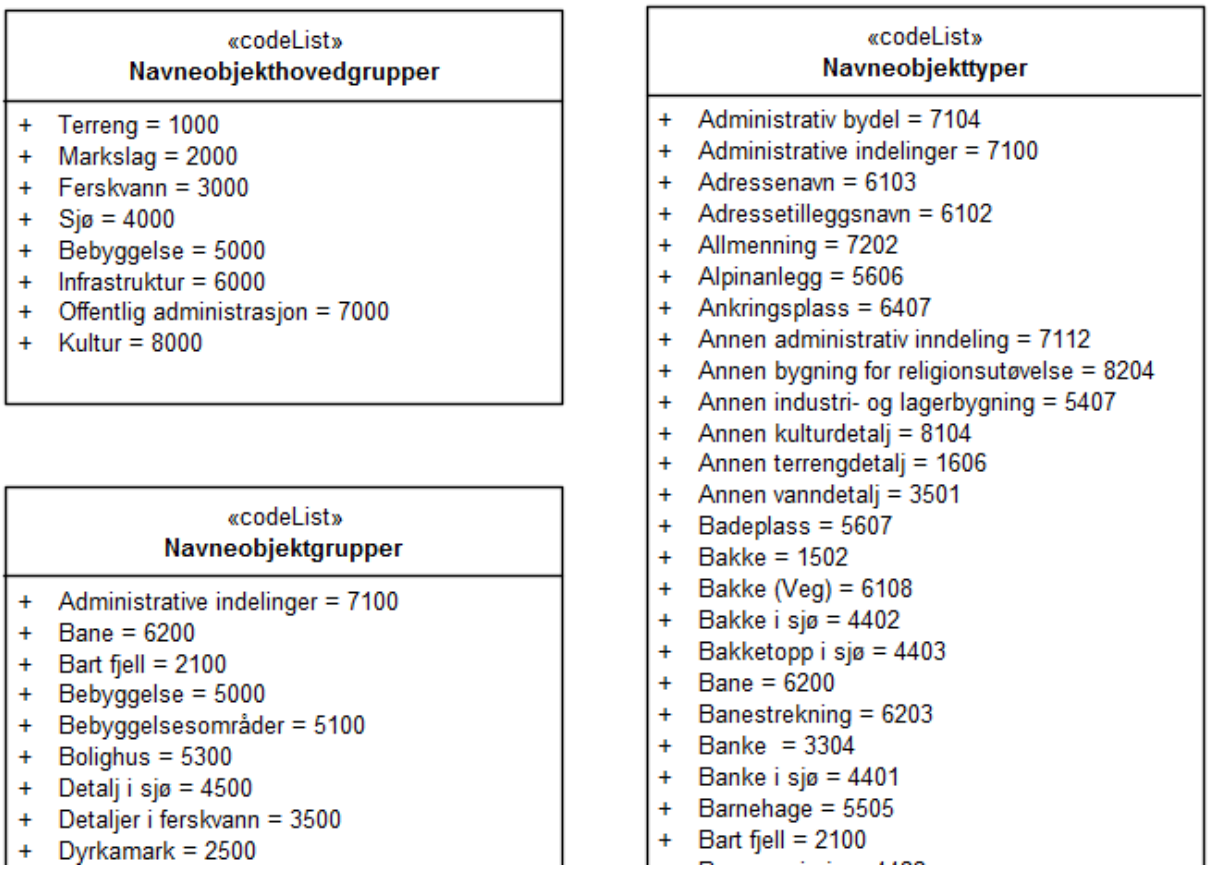
Eksempel på http-URI til (koder i) gml:Dictionary-fil:
http://skjema.geonorge.no/SOSI/produktspesifikasjon/Stedsnavn/4.5/Navnetypegruppe.xml

Mediatype for gml:Dictionary-xml-kodelister er *application/gml+xml*.

/anbefaling/gruppertKodeliste	Dersom kodelistekoder har innbyrdes forhold anbefales det å modellere grupperingskodene separat fra detaljkodene. Det innbyrdes forholdet mellom grupper og detaljkoder vil da måtte vedlikeholdes utenfor modellen eller som egendefinerte tagged values.

Eksempel:

Figur 12.9 viser kodelister med hierarkiforhold der en hovedgruppe inneholder et sett med grupper, og en gruppe inneholder et sett med detaljerte koder. I eksemplet indikeres at denne interne sammenhengen kommer fra et sett med eldre initialverdier. Denne gamle måten å angi sammenheng på vil det ikke være mulig å vedlikeholde over tid, så disse initialverdiene bør fjernes fra kodene.



Figur 12.9 Eksempel på hierarki av koder indikert via initialverdier på kodelister

13 SOSI-UML-profil

13.1 UML-applikasjonsskjema og tjenestemodeller

13.1.1 Introduksjon

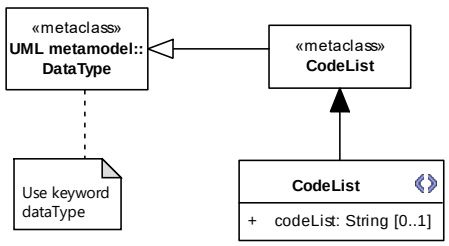
Dersom de vanlige modellelementene i standard UML ikke er tilstrekkelige for modelleringen kan man lage egne modellkonstruksjoner med egen spesifikk mening (semantikk). Dette gjøres ved å innføre UML-profiler med stereotyper og tilhørende tagged values. Dette kan hovedsaklig benyttes til modelldrevne plattformimplementasjoner. ISO 19103:2015 og ISO 19109:2015 definerer sine respektive plattformnøytrale stereotyper og tagged values, og ISO 19136:2007 har tilsvarende plattformspesifikk profilbeskrivelse for GML. I dette kapitlet beskrives en SOSI-formatprofil for forenkling av realisering i SOSI-format og en GML-formatprofil for realisering i GML-format. Disse UML-profilene har sammenhenger som er beskrevet i følgende UML-profildiagrammer.

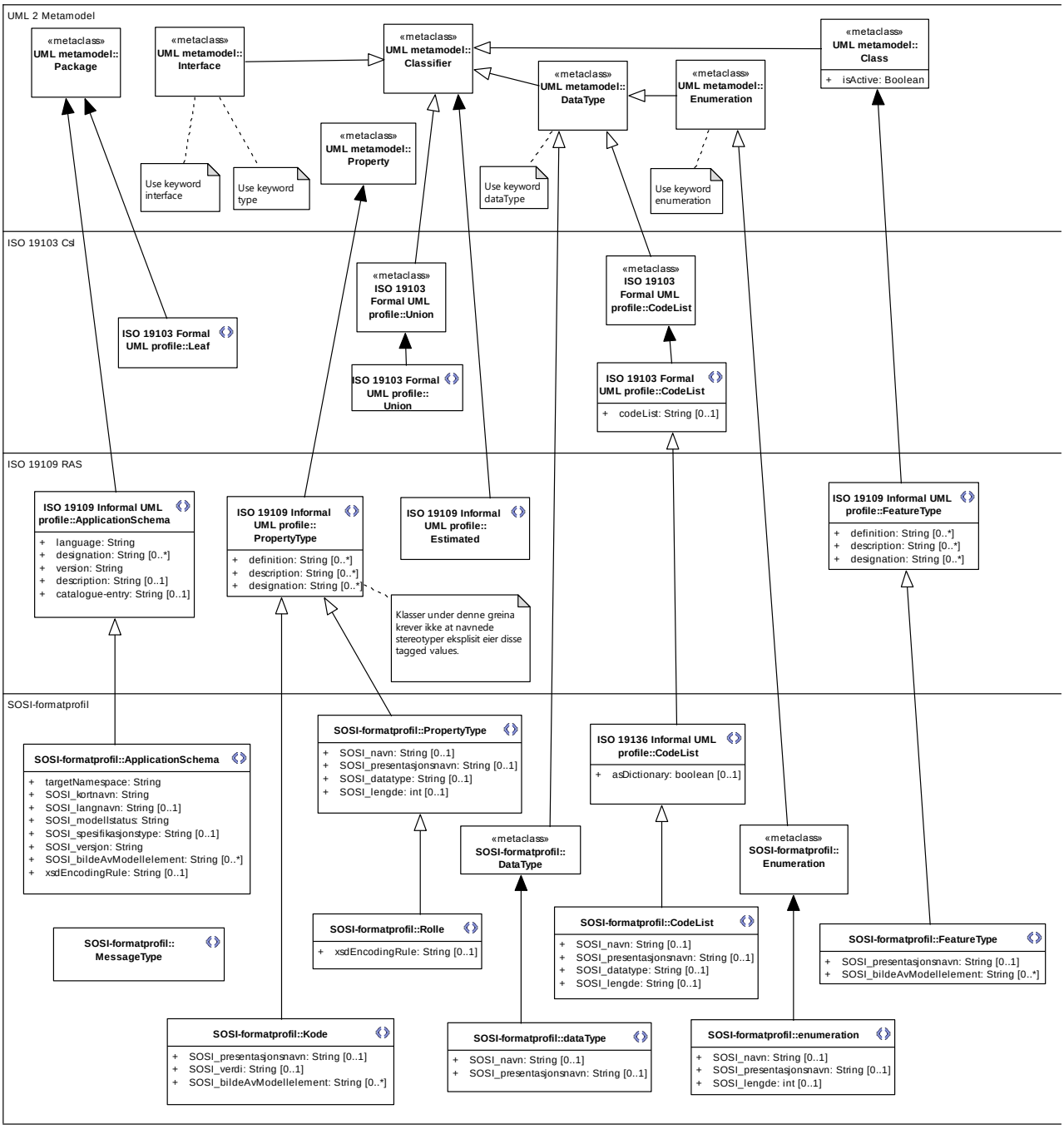
Alle tagged values bør på sikt forvaltes sammen i et register over alle modellelementer.

De gjøres også tilgjengelige som en implementasjon av SOSI-UML-profil for uvalgte modelleringsverktøy som forenkler realisering av innholdet i begge formater.

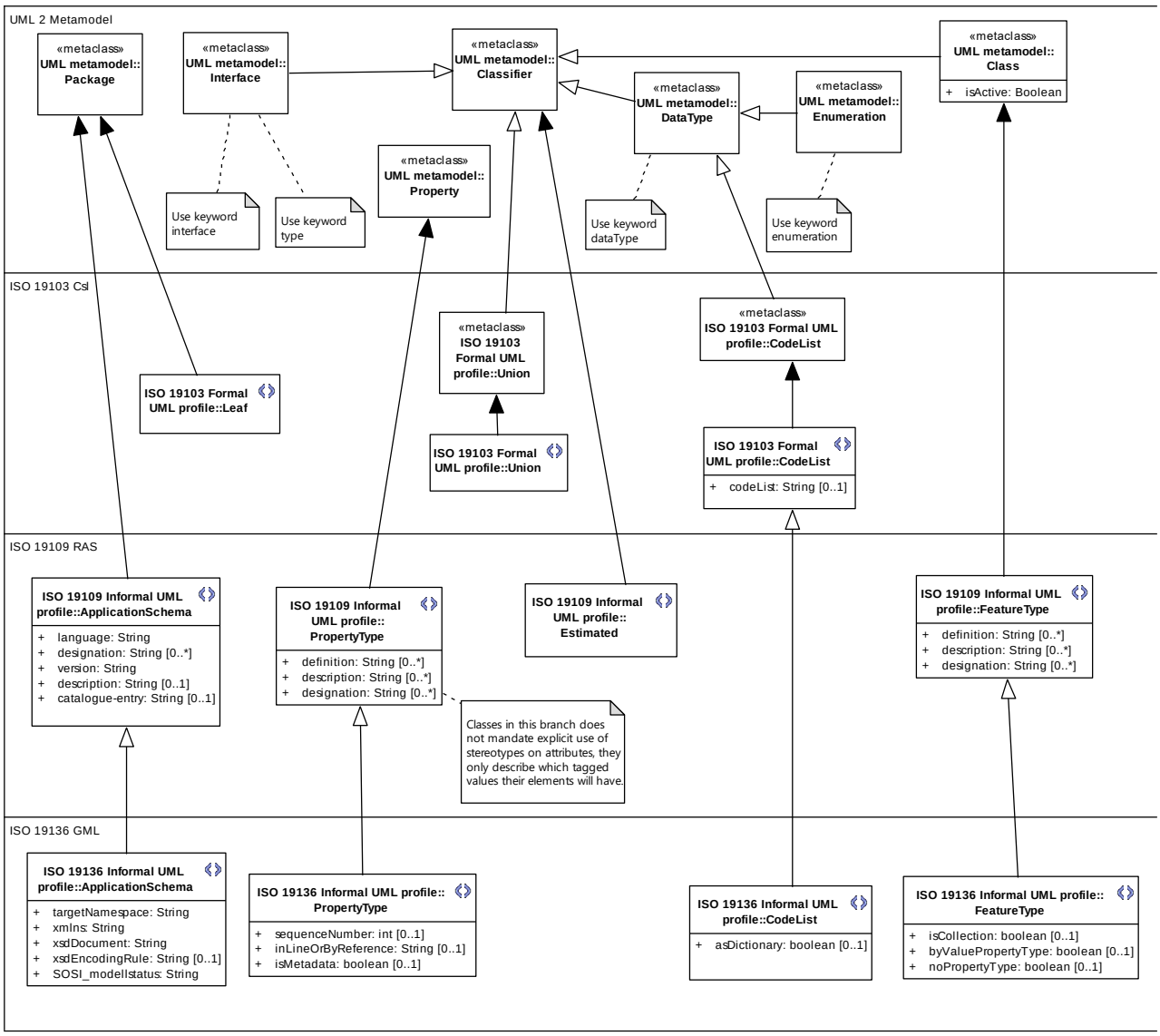
Url til installerbar xml-fil:
http://skjema.geonorge.no/SOSI/UML-modellering/5.1

Tabell 13.1 Profildiagram for å beskrive stereotyper

Stereotyper defineres i egne diagramtyper som kalles UML-profildiagram.		Stereotypens navn kommer fra metaklassenavnet i UML-profildiagrammet.



Figur 13.1 SOSI-formatprofil



Figur 13.2 GML-formatprofil

13.1.2 Krav til tagged values

/krav/ taggedValueSpråk	Alle applikasjonsskjemapakker skal gis språkkode.

/krav/ taggedValueGML	Et UML-applikasjonsskjema som skal være utgangspunkt for generering av GML-applikasjonsskjema skal fylle inn følgende tagged values som er obligatoriske for GML : language, version, targetNamespace, xmlns, xsdDocument, SOSI_modellstatus

/krav/ taggedValueSOSI	Et UML-applikasjonsskjema som skal være utgangspunkt for generering av SOSI filstruktur og parameterfiler for SOSI-kontroll skal fylle inn følgende tagged values som er obligatoriske for SOSI-format :

	language, version, targetNamespace, SOSI_kortnavn, SOSI_modellstatus, SOSI_versjon.

13.1.3 Beskrivelse av alle tagged values

Tabellen med beskrivelser og eksempler må ses i sammenheng med arkitekturfigurene. Figurene er UML profildiagram som angir hvilke modellelement som har hvilke tagged value, hvor i arkitekturen den er definert, og hvilke multiplisitet og type verdien har.

Tabell 13.2 Beskrivelse av alle tagged values.

tagged value:	**language**
beskrivelse:	språk som modellen hovedsaklig er modellert i
eksempel:	no
tagged value:	**designation**
beskrivelse:	navn i alternative språk, følger fast mal og kan gjentas i flere språk
eksempel:	"GeographicalNames"@en "NomsGéographiques"@fr
tagged value:	**version**
beskrivelse:	versjonsnummer for applikasjonsskjemaet
eksempel:	5.1.1
tagged value:	**description**
beskrivelse:	beskrivelse i alternativt språk, følger fast mal og kan gjentas i flere språk
eksempel:	"Stedsnavn etter Inspire-modell"@no "Geographical names according to the Inspire-model"@en
tagged value:	**catalogue-entry**
beskrivelse:	dersom elementet er hentet fra en ekstern objektkatalog kan en referanse tilbake til denne legges inn.
eksempel:	http://ssb.no/SamiskeValkrinsar
tagged value:	**targetNamespace**
beskrivelse:	Navnerom til applikasjonsskjemaet
eksempel:	http://skjema.geonorge.no/SOSI/produktspesifikasjon/Stedsnavn/4.5
forklaring:	Det siste elementet i navnerommet skal være sammensatt av kortnavnet og et versjonsnummer, og som også er identisk med siste del av pakkenavnet (Stedsnavn-4.5). Versjonsnummeret bør være lik første del av tagged value version. (4.5.1)
tagged value:	**xmlns**
beskrivelse:	kortform av navnerom til GML-applikasjonsskjema
eksempel:	Det anbefales å benytte verdien app i vanlige produktspesifikasjoner.
tagged value:	**xsdDocument**

beskrivelse:	filnavn for GML-applikasjonsskjema (uten skilletegn)
eksempel:	Stedsnavn.xsd
tagged value:	**SOSI_kortnavn**
beskrivelse:	kortnavn på modellen (uten skilletegn), bør være samme som brukes som pakkenavn og xmi-filnavn
eksempel:	KartlagtFriluftsliv-5.0
tagged value:	**SOSI_langnavn**
beskrivelse:	fullt navn på modellen
eksempel:	Kartlagte og verdsatte friluftslivsområder
tagged value:	**SOSI_modellstatus**
beskrivelse:	status for en modellpakke, obligatorisk ved realisering i GML og SOSI-format
eksempel:	godkjent, under arbeid, ...(se tabell 12.1 for fullstendig liste over verdier)
tagged value:	**SOSI_spesifikasjonstype**
beskrivelse:	type av modellpakke, angir om fellesegenskaper er innarbeidet
eksempel:	prodspesifikasjon eller fagområde
tagged value:	**SOSI_versjon**
beskrivelse:	versjonsnummer på den underliggende SOSI del-1
eksempel:	5.1
tagged value:	**SOSI_bildeAvModellelement**
beskrivelse:	angir url til illustrerende bilde av modellelementet
eksempel:	http://skjema.geonorge.no/SOSI/fagområdestandard/Gravplass/4.5/gravsted.png
merknad:	Format og bruksmønster for bildene må angis i produktspesifikasjon el.
tagged value:	**SOSI_presentasjonsnavn**
beskrivelse:	presentabel og lesbar variant av modellelementnavnet
eksempel:	Gang- og sykkelveg (fra modellelementet GangSykkelveg)
merknad:	Brukes kun om norske presentasjonsnavn, andre språk må bruke designation.
tagged value:	**xsdEncodingRule**
beskrivelse:	regelsett som skal brukes ved generering av GML-applikasjonsskjema
eksempel:	sosi50_2015 eller notEncoded
kilde:	ShapeChange
tagged value:	**definition**
beskrivelse:	definisjon i alternative språk, følger fast mal og kan gjentas i flere språk
eksempel:	"proper name of one geographic phenomenon"@en

	"le nom propre d'un phénomène géographique"@fr
tagged value:	**sequenceNumber**
beskrivelse:	heltall som angir fastsatt rekkefølge på egenskaper og roller, skal være unike for hver klasse. Anbefalt for å holde kontroll på rekkefølgen av egenskaper og roller i gml-applikasjonsskjema.
eksempel:	
tagged value:	**inLineOrByReference**
beskrivelse:	angir om verdien til egenskapen/rollen skal innkapsles eller refereres
eksempel:	inline
tagged value:	**isMetadata**
beskrivelse:	angir om egenskapen eller rollen er metadata til annet modellelement
eksempel:	false
tagged value:	**SOSI_navn**
beskrivelse:	navn som skal benyttes i filer på SOSI-format
eksempel:	SNTYSTAT
tagged value:	**SOSI_lengde**
beskrivelse:	maksimalt antall tegn for verdier av elementnavnet i SOSI-format
eksempel:	32
tagged value:	**SOSI_verdi**
beskrivelse:	SOSI-formatverdi for koden i ei kodeliste (ofte bokstav eller tallverdi)
eksempel:	T (brukes i SOSI-format istedenfor påTerrenget i kodelista Medium)
tagged value:	**codeList**
beskrivelse:	identifikator for ei kodeliste, helst en http-URI som kan benyttes som link til beskrivelsen
eksempel:	http://skjema.geonorge.no/SOSI/produktspesifikasjon/Stedsnavn/4.5/Navnetype.xml
tagged value:	**asDictionary**
beskrivelse:	angis dersom kodelista med sine koder skal ligge som en ekstern beskrivelse, og at den ikke skal ligge inne i GML-applikasjonsskjemaet
eksempel:	true
tagged value:	**SOSI_datatype**
beskrivelse:	angir hvilke SOSI-type verditypen skal mappes til (H,D,T,...)
eksempel:	D
kommentar	Nødvendig kun der automatisk mapping ikke er tilstrekkelig
tagged value:	**isCollection**

beskrivelse:	Angir at klassen beskriver en samling av objekter fra andre klasser
eksempel:	false
tagged value:	**byValuePropertyType**
beskrivelse:	angir om en objekttype som brukes som datatype til en egenskap skal kunne kodes inline
eksempel:	false
tagged value:	**noPropertyType**
beskrivelse:	true angir at klassen ikke skal generere en XSD ComplexType med fast navnemønster: <klasse>PropertyType.
eksempel:	true

Vedlegg A (normativt) Konformitetsklasser og tester

De regler for modellering som skal oppfylle de tekniske krav angitt i denne standarden må oppfylle kravene for de konformitetsklasser som modellene skal være konforme med. Disse konformitetsklassene er beskrevet i kapittel 4. Testene for disse konformitetsklassene er beskrevet i kapittel A1 til A.4.

Denne standarden dekker ulike typer modellering. Kravene til ulike typer modellering vil fremkomme i ulike konformitetsklasser.

A.1 Basisregler for UML

For å sikre ensartede og mest mulig like modeller er det viktig å benytte en rekke generelle regler og basiskomponenter som er beskrevet i internasjonale standarder. Denne konformitetsklassen tar utgangspunkt i ISO 19103 Conceptual schema language.

Konformitetsklassen for basisregler for modellering av UML klassediagrammer med realisering av internasjonale standarder er angitt i Tabell A.1A — Basisregler for UML.

Tabell A.1A — Basisregler for UML med realisering av internasjonale standarder

Hensikt med test	Verifisere at generelle konsepter i UML benyttes på en korrekt måte..
Testmetode	Inspisere modellen
Avhengighet	UML 2.0 ISO 19103 Conceptual schema language
Referanse	Alle krav i kapittel 10 med unntak av /krav/kodenavn og /krav/kodebruk
Type test	Basis

Konformitetsklassen for basisregler for modellering av UML klassediagrammer med nasjonale tilpassinger er angitt i Tabell A.1A — Basisregler for UML med nasjonale tilpassinger.

Tabell A.1B — Basisregler for UML med nasjonale tilpassinger

Hensikt med test	Verifisere at generelle konsepter i UML benyttes på en korrekt måte..
Testmetode	Inspisere modellen
Avhengighet	UML 2.0 ISO 19103 Conceptual schema language
Referanse	Alle krav i kapittel 10 med unntak av /krav/6
Type test	Basis

For krav identifisert med tall henvises til konformitetstester i ISO 19103.

A.2 UML-applikasjonsskjema

Et UML-applikasjonsskjema representerer en avbildning av et eller flere fenomener i den virkelige verden, fortrinnsvis geografiske. Denne avbildningen spesifiseres i form av et subset av UML, en generell objektmodell.

UML-applikasjonsskjema har flere konformitetsklasser, en generell klasse for kravene i kapittel 11 samt flere for valg av geometri og topologityper. Dette siste med fokus i å dele opp i ulike typer geometri/topologi ut fra brukerbehov.

Konformitetsklassene for modellering av UML-applikasjonsskjema er angitt i Tabell A.2 — UML-applikasjonsskjema til Tabell A.6 — Geometriske aggregater komplekser.

A.2.1 UML-applikasjonsskjema

Tabell A.2 — UML-applikasjonsskjema

Hensikt med test	Verifisere at et UML-applikasjonsskjema er korrekt modellert med unntak av det som har med geometri og topologi.
Testmetode	Inspisere modellen
Avhengighet	Konformitetsklasse: Basis regler for UML
Referanse	Alle kravene i kapittel 11 med unntak av krav under kapittel 11.4.3 Geometri, samt alle krav i kapittel 13
Type test	Basis

A.2.2 Enkle geometriske primitive

Tabell A.3 — Enkle geometriske primitiver

Hensikt med test	Verifisere at et UML-applikasjonsskjema som bruker enkle geometriske primitiver er korrekt modellert
Testmetode	Inspisere modellen
Avhengighet	Konformitetsklasse: UML-applikasjonsskjema
Referanse	krav/EnkleGeometriskePrimitiver
Type test	Basis

A.2.3 Andre geometriske primitive

Tabell A.4 — Andre geometriske primitive

Hensikt med test	Verifisere at et UML-applikasjonsskjema som bruker enkle geometriske primitiver er korrekt modellert
Testmetode	Inspisere modellen
Avhengighet	Konformitetsklasse: UML-applikasjonsskjema
Referanse	krav/AndreGeometriskePrimitiver
Type test	Basis

A.2.4 Geometriske komplekser

Tabell A.5 — Geometriske komplekser

Hensikt med test	Verifisere at et UML-applikasjonsskjema som bruker geometriske komplekser er korrekt modellert
Testmetode	Inspisere modellen
Avhengighet	Konformitetsklasse: UML-applikasjonsskjema

Referanse	krav/geometriskeKomplekser req/spatial/complex rec/spatial/composite
Type test	Basis

A.2.5 Geometriske aggregater

Tabell A.6 — Geometriske aggregater

Hensikt med test	Verifisere at et UML-applikasjonsskjema som bruker geometriske aggregater er korrekt modellert
Testmetode	Inspisere modellen
Avhengighet	Konformitetsklasse: UML-applikasjonsskjema
Referanse	/req/spatial/aggregate
Type test	Basis

A.2.6 Topologiske primitiver

Tabell A.7 — Topologiske primitiver

Hensikt med test	Verifisere at et UML-applikasjonsskjema som bruker toplogiske primitiver er korrekt modellert
Testmetode	Inspisere modellen
Avhengighet	Konformitetsklasse: UML-applikasjonsskjema
Referanse	/krav/GMLTopologi
Type test	Basis

A.2.7 Topologiske komplekser

Tabell A.8 — Topologiske komplekser

Hensikt med test	Verifisere at et UML-applikasjonsskjema som bruker topologiske komplekser er korrekt modellert
Testmetode	Inspisere modellen
Avhengighet	Konformitetsklasse: UML-applikasjonsskjema
Referanse	/krav/GMLTopologi /req/spatial/topo-complex
Type test	Basis

For krav identifisert med engelske navn henvises til konformitetstester i ISO 19109.

A.3 Tjenestemodellering

For å sikre ensartet og mest mulig lik modellering av tjenester er det viktig å benytte en rekke generelle regler og basiskomponenter som er beskrevet i internasjonale standarder. Denne konformitetsklassen tar utgangspunkt kapittel 9 som har en referanse til ISO 19119 Services.

Tabell A.9 — Tjenestemodellering

Hensikt med test	Verifisere at en tjenestemodell er korrekt modellert
Testmetode	Inspisere modellen
Avhengighet	Basis regler for UML
Referanse	Alle kravene i kapittel 9
Type test	Basis

A.4 Registre

SOSI-modellregister er et SVN-basert register som er en sentral infrastrukturkomponent for SOSI-metoden. Det inneholder UML- og BPMN-modeller som er viktige for arbeidet med standarder og spesifikasjoner som er relatert til stedfestet informasjon.

A.4.1 SOSI-modellregister

Tabell A.10 — SOSI-modellregister

Hensikt med test	Verifisere at de modellene som inngår i SOSI-modellregister er korrekt modellert.
Testmetode	Inspisere modellen
Avhengighet	Konformitetsklasser • BasisReglerForUML • UMLapplikasjonsskjema • Tjenestemodellering
Referanse	Alle krav i kapittel 12.1
Type test	Basis

A.4.2 Kodelister

Tabell A.11 — Kodelister

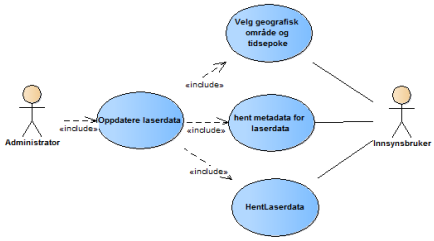
Hensikt med test	Verifisere at kodelister er korrekt konfigurert i modellen
Testmetode	Inspisere modellen
Avhengighet	Konformitetsklasser • BasisReglerForUML • UMLapplikasjonsskjema
Referanse	Alle krav i kapittel 12.2
Type test	Basis

Vedlegg B (informativt) «Use case» maler

B.1 I henhold til ISO 19119 Services

Tabell B.1 viser en tabell brukt for nærmere beskrivelse av et brukertilfelle («use case»).

Tabell B.1 — Brukstilfelle ISO 19119 Services

Brukstilfelle mal	Beskrivelse	Eksempel
Brukstilfelle navn	Navnet på brukstilfellet	*Oversikt over eksisterende og planlagte laserdata i Norge*
Brukstilfelle ID	Unik identifikator av brukstilfellet	*Laserdata_2014-12-01T13:44*
Versjon og referanse	Versjon = Versjonsnummer på brukstilfelle ID Referanse = URL til brukstilfellet	*V01,*
Brukstilfelle diagram	Beskrivelse av UML «use case» diagram for det aktuelle brukstilfellet. Diagrammet bør inkludere "extend" og relasjoner hvis de fines. UML-figuren kan legges til nederst i dokumentet ved å laste opp en bitmap generert i en UML-editor	
Status	Status for utvikling av brukstilfellet: En av de følgende: Planlagt Under utvikling	*Under utvikling*
Prioritet for gjennomføring (opsjon)	Beskrivelse av hvilken prioritet det har å få gjennomført brukstilfellet. En av de følgende: Må: Brukstilfellet må implementeres for å få godkjent løsningen /systemet Bør: Brukstilfellet bør implementeres for å få godkjent løsningen /systemet. Alternativ løsning kan aksepteres Kan: Brukstilfellet bør implementeres, men løsningen kan aksepteres uten at kravet er oppfylt.	*Må*
Mål	Kort beskrivelse av hvilket mål man oppnår ved å realisere brukstilfellet (maks 100 tegn).	*Kartløsning på nett som viser dekning av laserdata med tilhørende metadata.* *Løsningen skal vise dekningsoversikter med dato og punkttetthet*
Sammendrag	Utfyllende beskrivelse av brukstilfellet	*Løsningen skal vise brukeren hva som finnes av laserdata i Norge ved å hente inn ulike tjenester fra laser- forvaltningsløsningen og eksterne datakilder.* *Løsningen skal fungere i alle målestokksområder.*

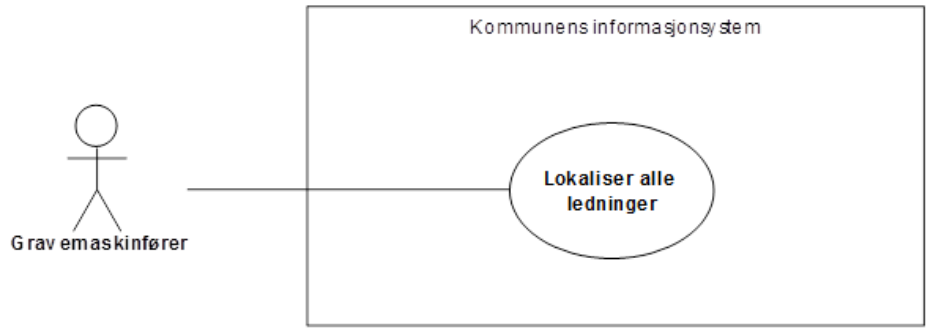
Kategori	Kategorisering av brukstilfellet i henhold til overordnet referansearkitektur.	*[avventer arbeidet med overordnet referansearkitektur]*
Aktør	Liste med brukere av brukstilfellet	• *Administrator/dataforvalter* • *Innsynsbruker* • *Online-bruker* • *Desktop bruker*
Primær aktør	Aktøren som initierer bruken av brukstilfellet.	*Alle*
Eier (opsjon)	Institusjon eller interessegruppe som har behov for å iverksette brukstilfellet.	
Nødvendige informasjonsressurser (opsjon)	Informasjon eller objekt som kreves for å kjøre brukstilfellet eller som blir generert gjennom brukstilfellet Nødvendig informasjonsressurs skal listes opp sammen med ressursens tilgangsmodus (les, skriv, oppdater eller slett) eller ”administrere” som omfatter alle modi.	• *Bakgrunnskart: topo2_graatone* • *Georef-laser, planlagte prosjekt/under arbeid (tilpasset løsningen) (WMS)* • *ProsjektDekning (WMS)* • *ProsjektDatoDekning (WMS) må lages* • *SkyggeTerreng (WMS)* • *SkyggeOverflate (WMS)* • *Punkttetthet (WMS)* • *PunkttetthetBakke (WMS)* • *Søketjeneste med autocomplete (adresse, stedsnavn, Gnr/Bnr, kommune, fylke, dekningsnummer, prosjektnavn)*
Forhåndsbetingelse	Beskrivelse av krav som stilles for å starte brukstilfellet Merk at brukstilfellene kan være koblet til hverandre via ”forhåndsbetingelser”. For eksempel kan en forhåndsbetingelse enten være en ekstern hendelse eller et annet brukstilfelle. Hvis det er et annet brukstilfelle bør brukstilfelle ID være oppgitt i dette feltet.	*Oppdaterte data med tilhørende metadata må være tilgjengelig i forvaltningsløsningen.*
Triggere (opsjon)	Ekstern hendelse som fører til at brukstilfellet blir utført. Merk at brukstilfellene kan være koblet til hverandre via ”triggere”. For eksempel kan en trigger for et brukstilfelle enten være en ekstern hendelse eller et annet brukstilfelle. Hvis det er et annet brukstilfelle bør brukstilfelle ID være oppgitt i dette feltet..	
Hovedsuksesscenario	Nummerert sekvens av aktiviteter (workflow) som må utføres for å realisere brukstilfellet.	*1. Bruker åpner løsning og får opp oversiktskart med laserdekning i egen farge.* *2. Bruker søker eller zoomer til aktuelt sted.* *3. SkyggeTerreng vises som default fra målestokk 1:500 000. Samtidig kommer det opp meny med valgmulighet for visning av SkyggeOverflate, Punkttetthet og PunkttetthetBakke.* *4. Løsningen skal vise metadata for alle prosjekt innenfor skjermbilde alternativt prosjekt under valgt punkt.*

Utvidelser	Utvidelse av en aktivitet under suksesscenario. Aktiviteten skal referere til hovedaktiviteten feks nr 1a, 1b osv.	1a.bruker definerer tidsområde. 1b. Brukeren definerer et feilaktig tidsområde. Et nytt dialogvinde åpnes og det kreves nytt tidsvindu..
Alternativ løype (opsjon)	Alternativ løype for å gjennomføre hovedsuksesscenariet, med referanse til en nummerert aktivitet.	4a. Bruker kan velge å vise rapport i ulike formater, tabularisk eller grafisk.
Krav til resultatet	Beskrivelse av resultatet etter vellykket gjennomføring av brukstilfellet	Rapport vises på skjerm.
Ikke-funksjonelle krav	Beskrivelse av ikke-funksjonelle krav med hensyn til ytelse, sikkerhet, kvaltet på tjenesten eller pålitelighet.	Visning av rapport på skjerm bør ikke ta lenger tid en 20 sekunder.
Suksesskriterier	Liste med kriterier som indikerer hvordan man validerer en vellykket realisering av brukstilfellet.	
Kommentar	Tilleggskommentarer eller notater (også for andre enn forfatteren).	
Dato og forfatter	Forfatter av brukstilfellet og dato for siste versjon	

B.2 I henhold til INSPIRE

B.2.1 Eksempel – brukstilfelle1 – Beliggenhet

En gravemaskinfører trenger å vite beliggenheten til alle ledninger i grunnen innen et område. "Use case" diagram er beskrevet i figur B.1.



Figur B.1 – "Use case" diagram – Brukstilfelle 1 – Beliggenhet

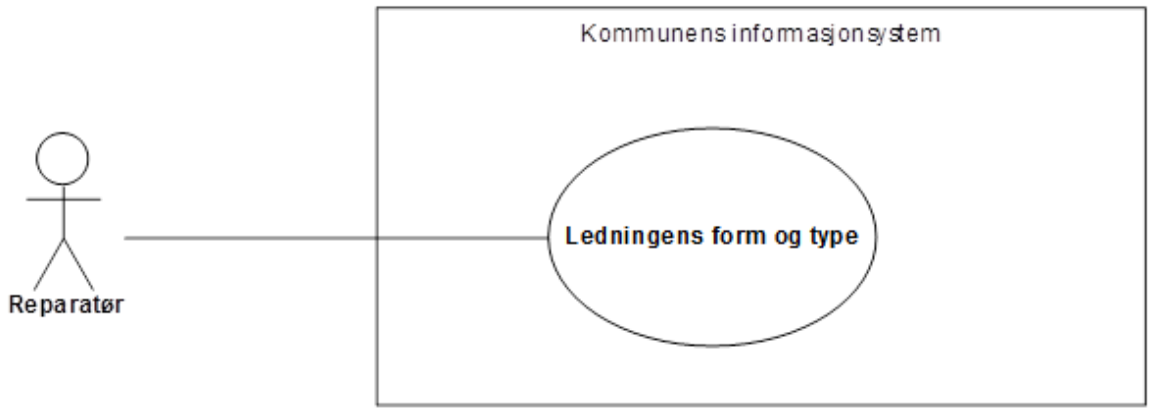
Tabell B.2 — Brukstilfelle 1

Brukstilfellebeskrivelse	
Brukstilfellets navn	**Beliggenhet**
Beskrivelse	**En gravemaskinfører må vite beliggenheten til alle ledninger i grunnen innen et område.**
Prioritet	**Lav ?**

Forhåndsbetingelse	
Sekvens av hendelser	
Steg 1	**Søk i kommunal database etter ledninger inne et område**
Steg 2	**Spør lokalkjente**
Steg 3	**Se etter spor i terrenget**
Resultat	**Satt ned stikker som angir hvor alle ledninger ligger**
Datakildebeskrivelse	**kart2.nois.no/hole**
Geografisk omfang	**Eiendom 202/27 og /61 i kommune 0612**

B.2.2 Eksempel – brukstilfelle 2 – Form

En reparatør trenger å vite type og form på en sammenkobling av avløpsledninger. "Use case" diagram er beskrevet i figur B.2.



Figur B.2 – "Use case" diagram – Brukstilfelle 2 - Form

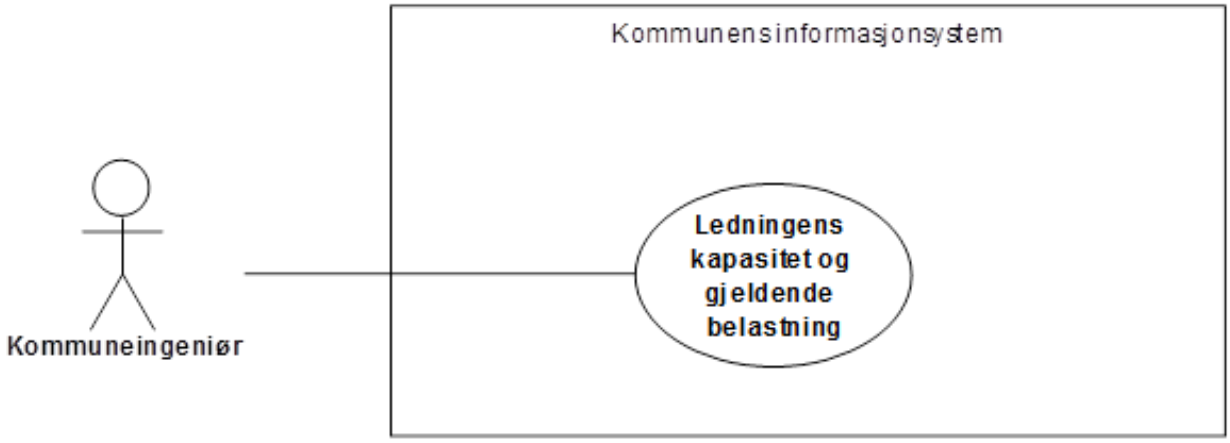
Tabell B.3 — Brukstilfelle 2

Brukstilfellebeskrivelse	
Brukstilfellets navn	**Form**
Beskrivelse	**En reparatør trenger å vite type og form på avløpsledninger og sammenkoblinger.**
Prioritet	**Lav ?**
Forhåndsbetingelse	
Sekvens av hendelser	

Steg 1	**Søk i kommunal database etter ledninger inne et område**
Steg 2	**Identifiser de aktuelle komponentene**
Steg 3	**Last ned og studer 3D-tegninger av komponentene**
Resultat	**Plan for graving og reparasjon**
Datakildebeskrivelse	**kart2.nois.no/hole**
Geografisk omfang	**Eiendom 202/27 og /61 i kommune 0612**

B.2.3 Eksempel – brukstilfelle 3 - Funksjon

En kommune må analysere om avløpsnettet klarer belastningsendringen fra et nytt byggefelt. "Use case" diagram er beskrevet i figur B.3.



Figur B.3 – "Use case" diagram – Brukstilfelle 3 - Funksjon

Tabell B.4 — Brukstilfelle 3

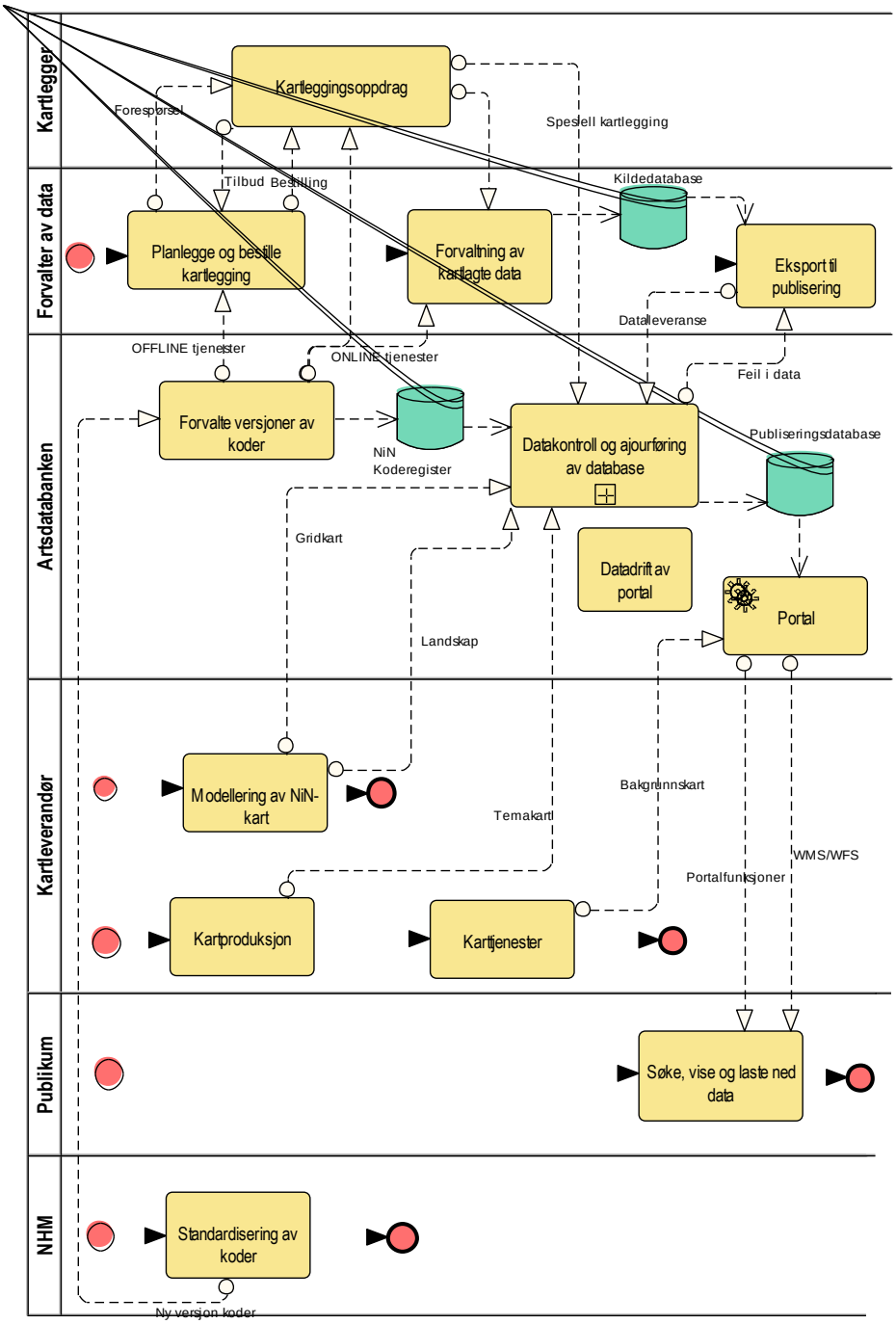
Brukstilfellebeskrivelse	
Brukstilfellets navn	**Funksjon**
Beskrivelse	**En kommune trenger å analysere om avløpsnettet klarer belastningsendringen fra et nytt byggefelt.**
Prioritet	**Lav ?**
Forhåndsbetingelse	
Sekvens av hendelser	
Steg 1	**Søk i kommunal database etter alle ledninger i nettverket**
Steg 2	**Identifiser de aktuelle komponentene**

Steg 3	**Last ned og analyser muligheten for ytterligere utnytting**
Resultat	**Anbefaling om byggetillatelse eller større nettoppgradering**
Datakildebeskrivelse	**kart2.nois.no/hole**
Geografisk omfang	**Soria Moria byggefelt i kommune 0612**

Vedlegg C (informativt) Eksempel på sammenhengen mellom ulike diagramteknikker

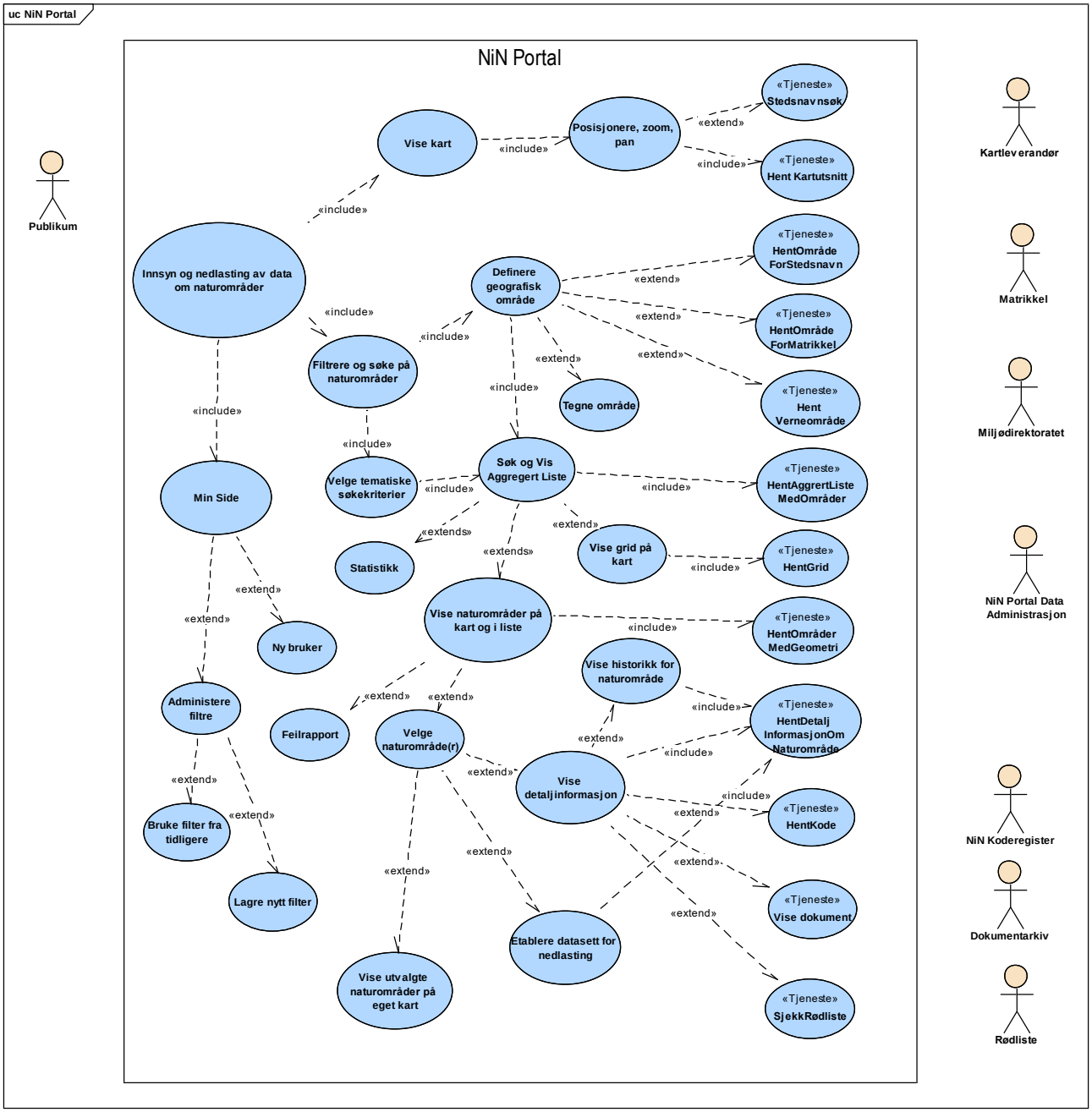
Dette vedlegget viser et gjennomgående eksempel fra artsdatabanken på sammenhengen mellom ulike typer diagrammer. (Dette eksemplet er fra tidlig i prosessen med arbeidet rundt artsdatabanken, og må kun benyttes som et eksempel på hvordan ulike diagramteknikker kan knyttes sammen).

Det første diagrammet beskriver forretningsprosessen i form av et BPMN diagram, se figur C.1.



Figur C.1 – Eksempel på prosessdiagram i BPMN 2.0

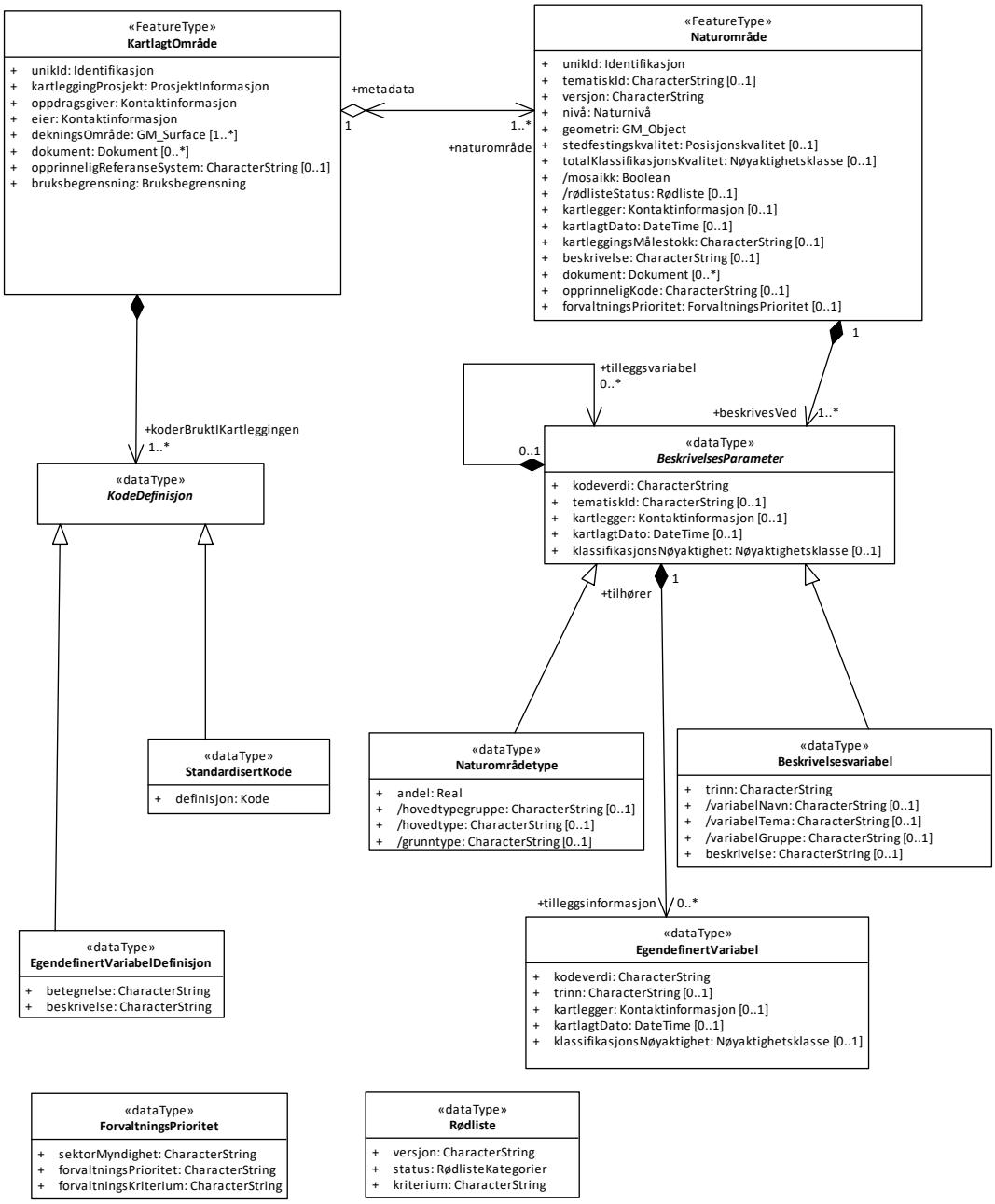
Hver av disse aktivitetene kan beskrives i form av et "use case" diagram. Figur C.2 viser brukstilfellene for aktiviteten “Søke, vise og laste ned data” i prosessdiagrammet.



Figur C.2 – Eksempel på UML "use case" diagram

I virksomhetsdiagrammet er det beskrevet en "publiseringsdatatbase", som vil inneholde dataene.

Disse dataene er ytterligere modellert i form av et UML-applikasjonsskjema. Figur C.3. inneholder deler av applikasjonsskjema for NiN publiseringsdatabasen.



Figur C.3 – Eksempel på UML klassediagram (applikasjonsskjema)

Vedlegg D (informativt) Modellering av nasjonale data med utgangspunkt i INSPIRE modeller

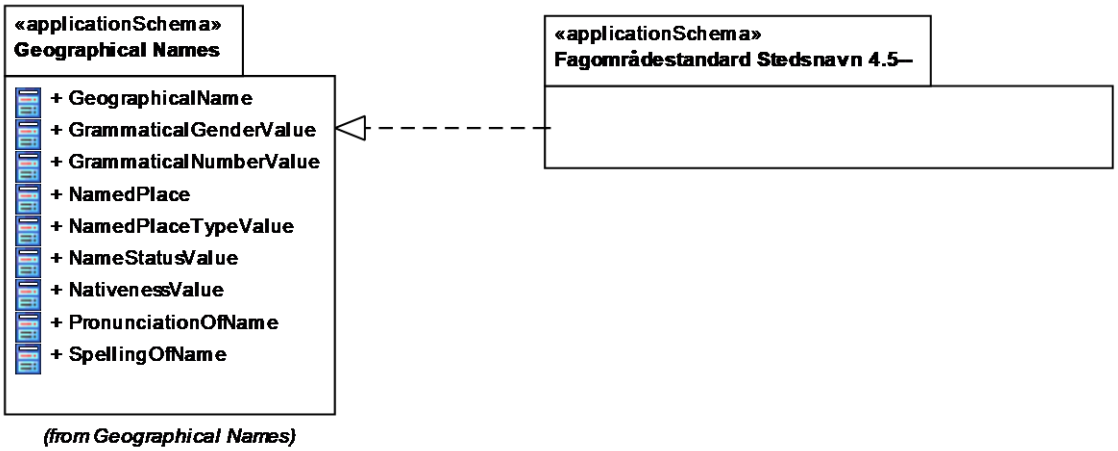
Dette vedlegget viser eksempler på hvordan en nasjonal modell kan bygge på INSPIRE modeller. Standarden har ingen krav på hvordan dette gjøres, men vil inneholde anbefalinger på ulike metoder.

D.1 Eksempel 1 - Påbygging på INSPIRE med bruk av <realize>

Dette vedlegg beskriver hvordan objekter, egenskaper og assosiasjoner i et fagområde kan knyttes opp til en INSPIRE (Geodatalov) spesifikasjon med bruk av modelleringsmetodikken <<realize>>. Det er en løselig kopling, hvor ingen INSPIRE modell elementer er en del av fagområdemodellen, men en forteller hvile INSPIRE modell element som er realisert (utgangspunkt for) den nasjonale spesifikasjonen. En gjenbruker ikke noen modellelementer fra INSPIRE. XML skjema (GML) som utvikles vil ikke gjenbruke INSPIRE GML skjema.

D.1.1 Pakketilknytning

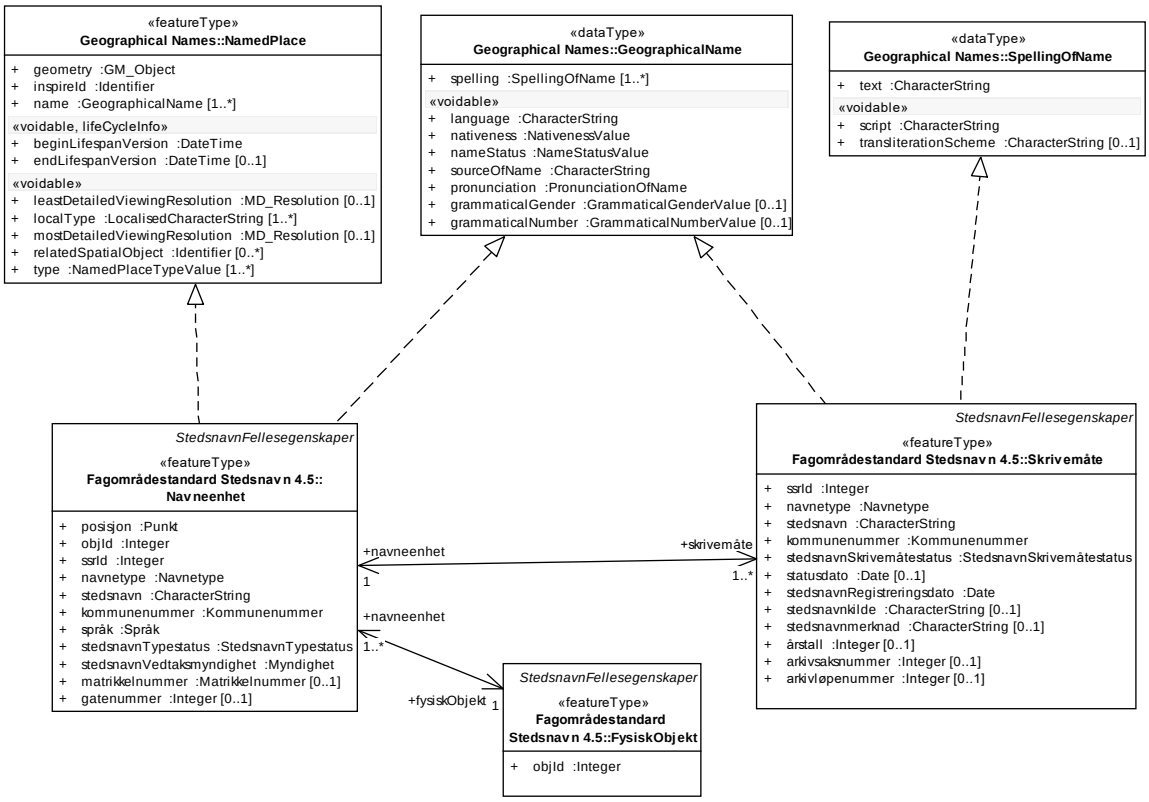
Figur D.1 viser et eksempel på forholdet mellom applikasjonsskjema SOSI Stedsnavn i SOSI og INSPIRE GeographicalName.



Figur D.1 – Pakketilknytning

D.1.2 Realisering av pakker

Figur D.2 viser et eksempel på realisering av flere pakker fra INSPIRE GeographicalName.



Figur D.2 – Eksempel på <<Realisering>> av pakker

D.1.3 Mappingtabell INSPIRE - SOSI

Tabeller som i detalj beskriver hvilke nasjonale elementer som realiserer hvilke INSPIRE elementer finnes i egen tabellfil, se figur D.3.

Application Schema 'Geographical Names' (version 3.0)							Application Schema <Stedsnavn 4.5>								
Type	Documentation	Attribute Association role Constr	Attribute / Association role / Constraint	Values / Enumerations	Multiplicity	Voidable / Non-Voidable	Type	Do cume	Attribute Association role Constr	Attrib ute	Values / Enumeratio ns	Multi plicity	Voida bl	Status	Remarks
GeographicalName	Proper noun applied to a real world entity.						Skrivemåte							Lett	Ikke avslåtte skrivemåter
		language	Language of the name, given as a three letters code, in accordance with either ISO 639-3 or ISO 639-5.NOTE 1More	CharacterString	1	voidable			fx(navneenhet.språk)		kodeliste			Lett	mappes til ISO 639
		nativeness	Information enabling to acknowledge if the name is the one that is/was used in the area where the spatial object is	NativenessValue* endonym* exonym	1	voidable					endonym			1:1	fast verdi innenfor Norges landområder
		nameStatus	Qualitative information enabling to discern which credit should be given to the name with respect to	NameStatusValue* official* standardised* historical* other	1	voidable			fx(stedsnavnTypestatus)		kodeliste			Lett	kun koder G+V+S; bør også ha med H+P+L . stedsnavnTypestatus
		sourceOfName	Original data source from which the geographical name is taken from and integrated in the data set	CharacterString	1	voidable			stedsnavnKilde		CharacterString			1:1	Kan dette være en forekomsinformasjon lik N50, N250, etc ??
		pronunciation	Proper, correct or standard (standard within	PronunciationOfName	1	voidable								ikke aktuell	
		spelling	A proper way of writing the geographical name.NOTE 1 Different spellings should only be used for names rendered in different scripts. . NOTE 2 While a	SpellingOfName	1..*				stedsnavn		CharacterString			1:1	Her er det en uløselig situasjon i henhold til "Lov om stadnamn" versus Note 2. Vi har forskjellige skrivemåter som ikke kan prefereres hvis en da ikke bytter nivå skrivemåte
		grammaticalGender	Class of nouns reflected in the behaviour of associated	GrammaticalGenderValue* masculine* feminine* neuter*	0..1	voidable								ikke aktuell	
		grammaticalNumber	Grammatical category of nouns that expresses count distinctions.NOTE the attribute has cardinality [0..1] and is voidable,	GrammaticalNumberValue* singular* plural* dual	0..1	voidable								ikke aktuell	Vi har navnetyper som representeres entall/(flertalls forhold. 215;265;261; 262:214. Totalt 1700 objekter med 2250 skrivemåter.

Figur D.3 – Eksempel på deler av mappingtabell INSPIRE – SOSI for stedsnavn

D.1.4 Oppsummering

Fordeler

- Enklere modelleringsmessig. Forholder seg ikke til INSPIRE klasser med unntak av å vise hvilke som er utgangspunkt for de nasjonale objekttyper/egenskaper.
- Norske navn på alle modellelementer. Ved bruk av alias mekanismen kan vi også ha engelske navn.
- Inneholder "mapping" regler fra norske data til INSPIRE (Geodatalov), som går ut over det som det er mulig å beskrive i en modell.

Ulemper:

- Ingen automatisk oppdatering med tanke på endringer i INSPIRE (Geodatalov) modeller og skjema.
- Kan ikke gjenbruke INSPIRE GML skjemaer, tyngre å implementere (erfaring fra ELF)
- Mappingreglene ligger ikke i modellen.

D.2 Eksempel 2 – Subtyping og utvidelser av en kopi av INSPIRE modell

D.2.1 Introduksjon

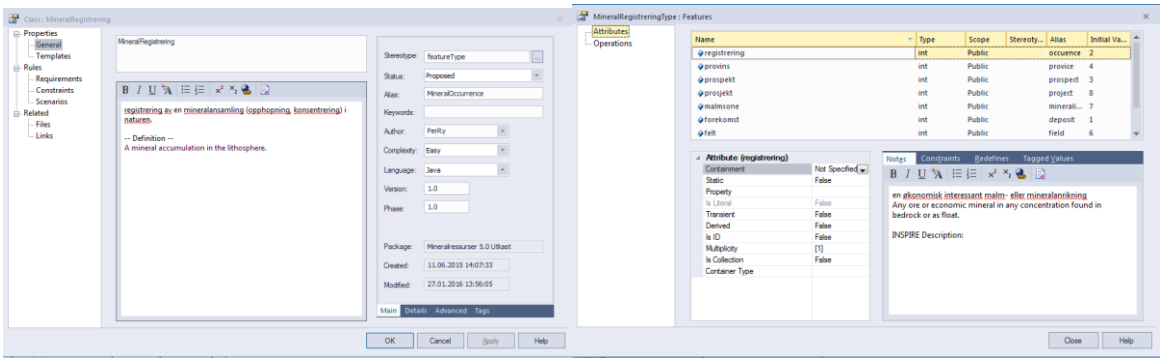
Dette er et eksempel fra SOSI mineralressurser, en ny objektkatalog under utarbeidelse av NGU. Den skal erstatte dagens SOSI-del 2, Råstoffutvinning, som er fra 2006, og hvor det er behov for flere oppdateringer. Samtidig med en slik omfattende revisjon av en SOSI-standard er det ønskelig å inkludere obligatoriske og viktige deler av UML-klassediagrammet fra INSPIRE dataspesifikasjonen MineralResources. INSPIRE- spesifikasjonen gir oss anledning til å beskrive og verdisette viktige mineralforekomster i Norge på en bedre måte, samtidig som det gir innholdet en engelsk språkform og et applikasjonsskjema som gjør det mulig å kommunisere innholdet med den øvrige del av verden.

Målet er å beholde så mye som mulig av SOSI-spesifikasjonen, men koble denne til deler av applikasjonsskjema fra INSPIRE-spesifikasjonen. Denne gir oss samtidig anledning til å koble videre mot klassediagrammer for Geologi, en annen av dataspesifikasjonene i INSPIRE.

D.2.2 Bruk av alias

Figur D.3 illustrerer at de engelske navnene i INSPIRE-spesifikasjonen legges inn som alias og klassenavnet erstattes av norsk navn. Det samme gjelder datatyper og kodelister. Det legges dermed opp til skrivetilgang til både norsk og engelsk kopi av INSPIRE-spesifikasjonen/SOSI-spesifikasjonen. Definisjonene skal også oversettes begge veier. Dette gjøres i modelleringsprogrammet Enterprise Architect (EA).

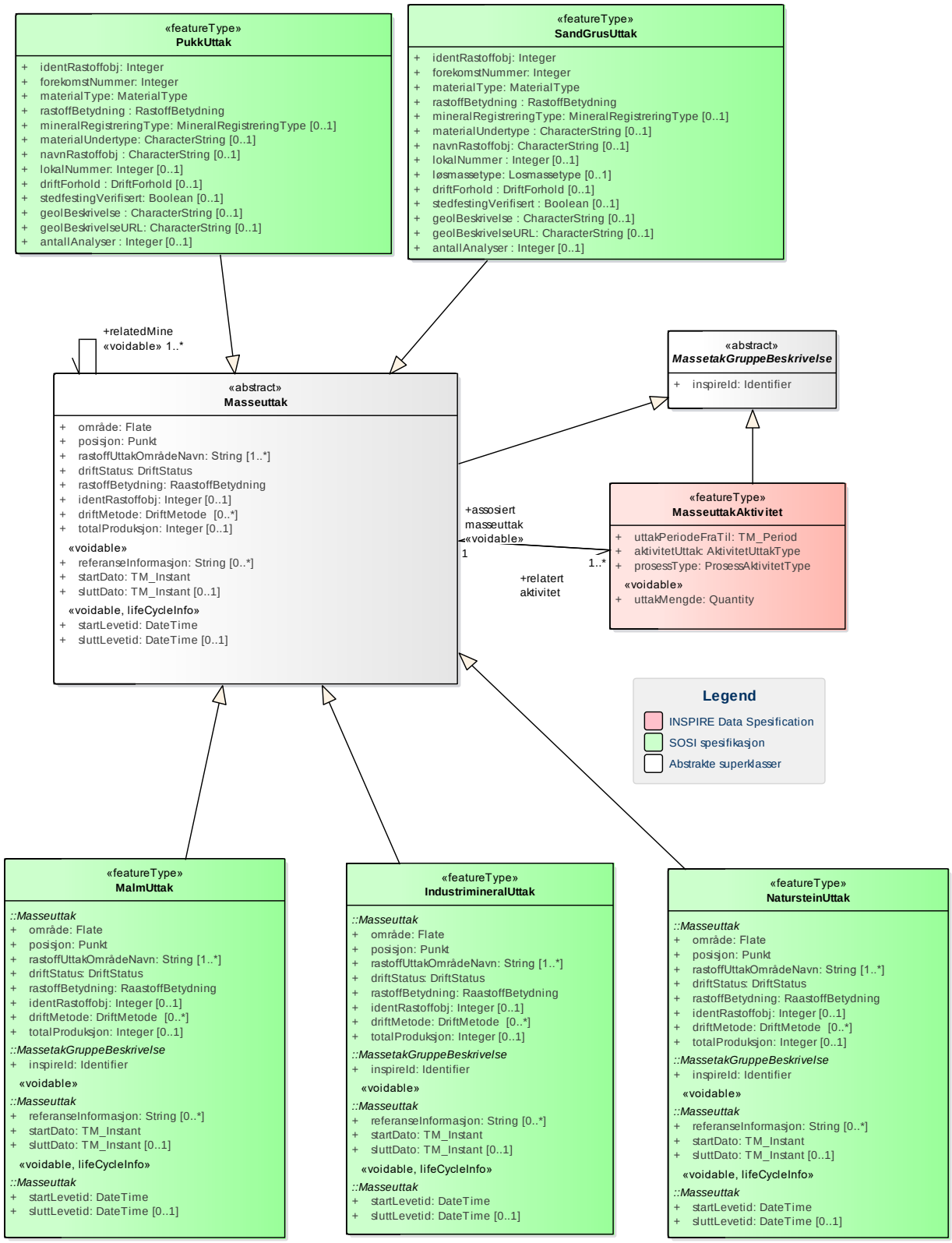
NB. Denne standarden har regler for språkhandtering, basert på predefinerte tagged values. Denne tillater også flere språk. Se 10.2.6 Bruken av alias er en foreløpig mekanisme i EA for å handtere to språk.



Figur D.3 – Bruk av "alias" for å ha engelsk og norsk i modellene

D.2.3 Subtyping av INSPIRE modeller

Figur D.5 viser hvordan objekter, egenskaper og assosiasjoner fra en kopi av en INSPIRE-modell kan kobles sammen med SOSI modellelementer. Eksemplet er hentet fra et utsnitt av datamodellen for Mineralressurser 5.0 som er under utarbeidelse og ligger i SOSI-modellregister. Deler av INSPIRE-spesifikasjonen for Mineralressurser kopieres og knyttes sammen med elementer fra eksisterende modell i SOSI. Kun deler av samordningen av modellene er vist i figuren. Den delen av klassediagrammet som stammer fra INSPIRE-spesifikasjonen er angitt i rød farge, og uten farge når klassen er abstrakt. Klasser fra SOSI-standarden har grønn farge. En subtyping gir oss mulighet til å beholde den mer detaljerte klasseinndelingen vi måtte ha i det nasjonale fagsystemet. I dette tilfellet at et generelt masseuttak (INSPIRE = mine) deles inn i ulike typer av uttaksområder (henholdsvis Pukk, SandGrus, Industrimineraler, naturstein og malm (metaller)). Det at det opereres med en kopi av INSPIRE-klassene gjør det samtidig mulig å oversette klassenavn, egenskaper og kodeverdier til norsk. Det engelske innholdet er bevart som alias i modellen.



Figur D.5 – Subtyping av INSPIRE pakker i SOSI

Figur D.5 viser hvordan SOSI mineralressurs-klasser er "subtypet" fra en kopi av INSPIRE-klasser i dataspesifikasjonen. Erfaringene så langt innenfor fagområdene geologi og mineralressurser har vist at klassene definert i INSPIRE ofte er en generalisering av mer detaljert klasseinndeling i SOSI. En kobling mellom INSPIRE-skjema og SOSI-skjema vil i slike tilfeller gjøres med en enkel ”Subtyping”. SOSI-klassene arver dermed egenskapene til INSPIRE-supertypen.

D.2.4 Oppsummering

Fordeler

- Enklere modelleringsmessig ved å jobbe på en kopi av INSPIRE.
- Endringer som skjer i INSPIRE-skjemaene vil resultere i et behov for endringer i eksisterende nasjonale databaser, men endringene kan gjennomføres på en kontrollert måte og ikke skje automatisk.
- Ved bruk av alias mekanismen kan vi introdusere både norske og engelske navn.
- Man kan utvide leveransen til nasjonale formål med elementer fra INSPIRE-spesifikasjonen uten å endre for mye i den opprinnelige nasjonale dataleveransen.
- Man kan foreta en nasjonal dataleveranse til Norge digitalt og en leveranse til INSPIRE uten for mye dobbeltarbeid.
- INSPIRE-modellen har i mange tilfeller en bedre faglig strukturering/oppbygging av klassene, og gir oss mulighet til sammenkobling med applikasjonsskjema fra andre beslektede temaområder fra INSPIRE.

Ulemper:

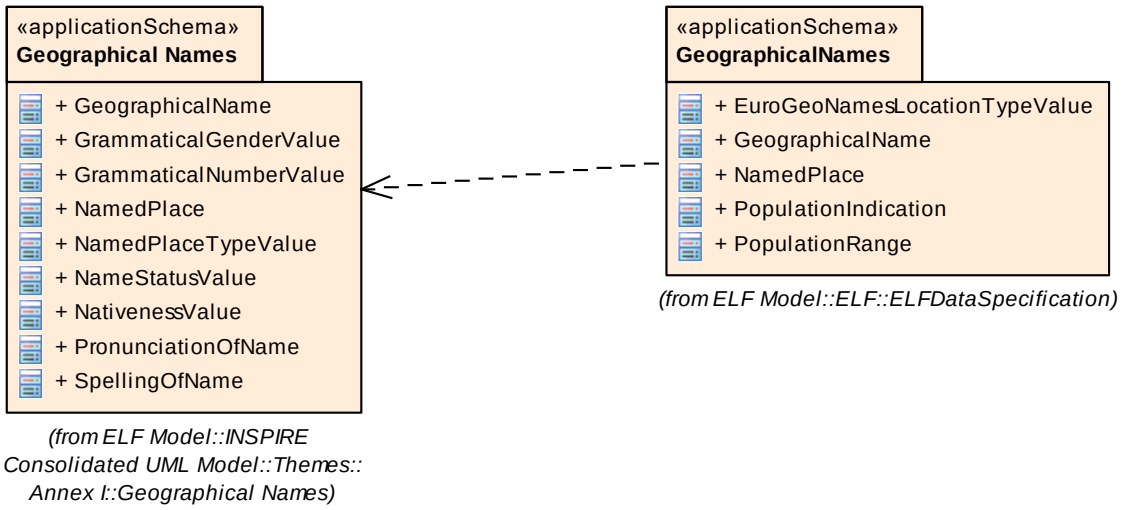
- Ingen automatisk oppdatering med tanke på endringer i INSPIRE-modeller og -skjema.
- Vil resultere i et behov for å gjøre endringer i eksisterende nasjonale databaser.
- Kan ikke gjenbruke INSPIRE GML-skjemaer. Er derfor tyngre å implementere (red: erfaring fra ELF)

D.3 Eksempel 3 – Subtyping og utvidelser av INSPIRE modell (original)

Dette er et eksempel fra EU prosjektet E.L.F (European Location Framework).
Dette vedlegg beskriver hvordan objekter, egenskaper og assosiasjoner i et fagområde utvider de offisielle INSPIRE (Geodatalov) spesifikasjonene ved bruk av modelleringsmetodikken <<subtype>>. Her vil offisielle INSPIRE modell elementer inngå som en del av den norske modellen, ikke i form av en kopi, men med direkte tilgang til INSPIRE SVN. Det er i regi av prosjektet (ELF) utviklet egne modelleringsregler (modeling guidelines) for å beskrive utvidelser og restriksjoner i henhold til INSPIRE modellelementer. XML skjema (GML) som utvikles vil gjenbruke INSPIRE GML skjema komponenter og derav enklere implementasjon!

D.3.1 Pakketilknytning

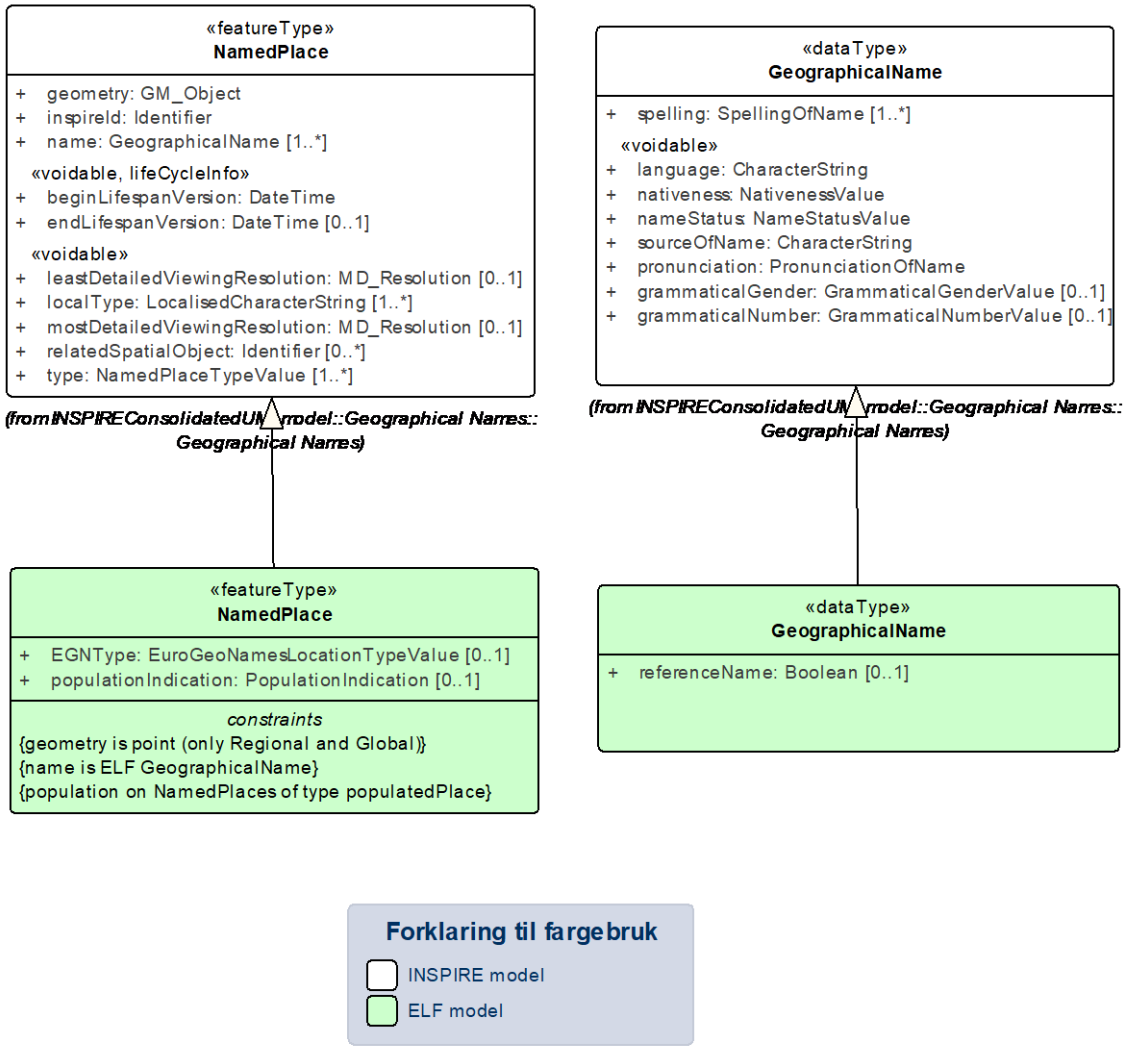
Figur D.6 beskriver hvilke modellelementer som inngår i ELF applikasjonsskjema GeographicalNames (høyre).



Figur D.6 – Pakketilknytning

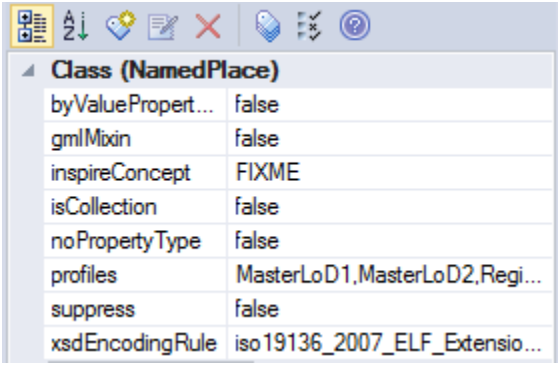
D.3.2 Subtyping av pakker

Figur D.7 viser hvordan ELF klasser er subtypet fra INSPIRE pakker.



Figur D.7 – Eksempel på subtyping av INSPIRE pakker i ELF

Nærmere konfigurering av klasser og egenskaper skjer gjennom bruk av TaggedValues. Figur D.8 beskriver TaggedValues for ELF klassen NamedPlace. Forklaringen til disse verdiene er beskrevet i eget dokument "ELF modelling guidelines". For tagged values brukt i Norge, se kapittel 13.



Figur D.8 – Konfigurering i form av tagged values

D.3.3 Oppsummering

Fordeler

- Enkelt å holde oppdatert med tanke på endringer i INSPIRE (Geodatalov) spesifikasjoner.
- Forenklet implementasjon, gjenbruker INSPIRE GML skjema, kun nasjonale skjema for det som begrenser/utvider INSPIRE.
- Verktøystøtte. ShapeChange tar hensyn til tagged values ved generering av både GML skjema og objektkatalog.

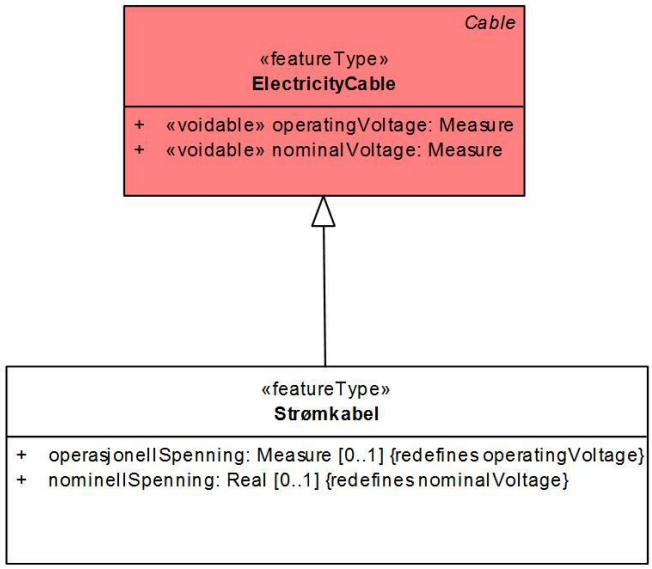
Ulemper:

- Mer kompleks modellering
- Mer bruk av Tagged values, kommer ikke fram i diagrammene. (Flytter det som naturlig vil være en del av den grafiske notasjonen inn som Tagged Values, men fullt lovlig jfr UML's metamodell.

D.4 Eksempel 4 – Subtyping av INSPIRE modeller i form av «redefine»

UML versjon 2 har en mekanisme for å redefinere egenskaper, redefine. På denne måten er det mulig å subtype INSPIRE (uten å ha skrivetilgang til modellen, samt redefinerte engelske egenskaper til å bruke norske navn.

Figur D.9 viser et eksempel på Strømkabel som en subtype av ElectricityCable, samtidig som egenskapene til ElectricityCable redefineres til norske egenskaper.



Figur D.9 – Redefiniering av arvede egenskaper fra INSPIRE

Det er imidlertid uklart i hvor stor grad det er verktøystøtte for denne type modellering. Nærmere utprøving må skje før vi anbefaler denne metodikken.

Vedlegg E (informativt) Tekstlig beskrivelse av UML-modeller

E.1 Tekstlig beskrivelse av SOSI_Fellesegenskaper og SOSI_Objekt

Feature Type: SOSI_Fellesegenskaper

SOSI_Fellesegenskaper	
Definisjon:	abstrakt objekttype som bærer sentrale egenskaper som er anbefalt for bruk i produktspesifikasjoner. Merknad: Disse egenskapene skal derfor ikke modelleres inn i fagområdemodeller.
Description:	This type is abstract.
Type:	Feature Type
Egenskap:	
Navn:	identifikasjon
Definisjon:	unik identifikasjon av et objekt
Multiplisitet:	1
Verditype:	Identifikasjon (data type)
Egenskap:	
Navn:	oppdateringsdato
Definisjon:	tidspunkt for siste endring på objektet Merknad: Oppdateringsdato kan være forskjellig fra datafangsdato ved at data som er registrert kan bufres en kortere eller lengre periode før disse legges inn i datasystemet (databasen).
Multiplisitet:	1
Verditype:	DateTime
Egenskap:	
Navn:	gyldigFra
Definisjon:	Tidspunktet når objektet oppstod i den virkelige verden
Multiplisitet:	0..1
Verditype:	DateTime
Egenskap:	
Navn:	gyldigTil
Definisjon:	Tidspunktet når objektet opphørte å eksistere i den virkelige verden
Multiplisitet:	0..1
Verditype:	DateTime

Feature Type: SOSI_Objekt

SOSI_Objekt

Definisjon:

abstrakt objekttype som bærer en rekke egenskaper som er fagområde-uavhengige og kan benyttes for alle objekttyper

Merknad:

Spesielt i produktspesifikasjonsarbeid vil en velge egenskaper og av grensningslinjer fra denne klassen.

Description:

This type is abstract.

Subtype av:

SOSI_Fellesegenskaper

Type:

Feature Type

Egenskap:

Navn:	datafangstdato
Definisjon:	dato når objektet siste gang ble registrert/observert/målt i terrenget Merknad: I mange tilfeller er denne forskjellig fra oppdateringsdato, da registrerte endringer kan bufres i en kortere eller lengre periode før disse legges inn i databasen. Ved førstegangsregistrering settes datafangstdato lik førsteDatafangstdato.
Multiplisitet:	0..1
Verditype:	DateTime

Egenskap:

Navn:	førsteDatafangstdato
Definisjon:	dato når data ble registrert/observert/målt første gang, som utgangspunkt for første digitalisering Merknad: førsteDatafangstdato brukes hvis det er av interesse å forvalte informasjon om når en ble klar over objektet. Dette kan for eksempel gjelde datoen for første flybilde som var utgangspunkt for registrering i en database.
Multiplisitet:	0..1
Verditype:	DateTime

Egenskap:

Navn:	førsteDigitaliseringsdato
Definisjon:	dato når en representasjon av objektet i digital form første gang ble etablert Merknad: førsteDigitaliseringsdato kan skille seg fra førsteDatafangstdato ved at den første datafangsten skjedde analogt og gjort om til digital form senere i en produksjonsprosess. Eventuelt at innlegging i databasen skjedde på et senere tidspunkt enn registreringen /observasjonen / målingen av objektet.
Multiplisitet:	0..1

Verditype:	DateTime
Egenskap:	
Navn:	verifiseringsdato
Definisjon:	dato når dataene er fastslått å være i samsvar med virkeligheten Merknad: Verifiseringsdato er identisk med ..DATO i tidligere versjoner av SOSI
Multiplisitet:	0..1
Verditype:	DateTime
Egenskap:	
Navn:	sluttdato
Definisjon:	Dato (og tid) for når denne versjonen av objektet var erstattet eller opphørt å eksistere.
Multiplisitet:	0..1
Verditype:	DateTime
Egenskap:	
Navn:	datauttaksdato
Definisjon:	dato for uttak fra en database Merknad: Skiller seg fra Kopidato ved at en ikke skiller på om det er uttak fra en originaldatabase eller en kopidatabase.
Multiplisitet:	0..1
Verditype:	DateTime
Egenskap:	
Navn:	endringsflagg
Definisjon:	endringsinformasjon om et objekt Merknad: Reglene knyttet til bruken av endringsflagg er for denne versjonen ikke avklart. Utdypes nærmere i produktspesifikasjonen basert på 4.0. Merknad: Endringsflagg kan benyttes til å merke slettede "objekter". Eksempel: Dersom en eiendomsgrense endres skal endringsflagg også legges inn på eiendomsteigen
Multiplisitet:	0..1
Verditype:	Endringsflagg (data type)
Egenskap:	
Navn:	kvalitet
Definisjon:	beskrivelse av kvaliteten på stedfestingen Merknad: Denne er identisk med ..KVALITET i tidligere versjoner av SOSI.
Multiplisitet:	0..1
Verditype:	Posisjonskvalitet (data type)
Egenskap:	
Navn:	status

Definisjon: objektets tilstand

Eksempel: Brukes, drift, foreldet, planlagt etc

Multiplisitet: 0..1

Verditype: Status (code list)

Koder:

brukes	Brukes
drift	Drift
eksisterende	Eksisterende (default). Identisk med tidligere SITSTAT = 3
foreldet	Foreldet. Identisk med tidligere SITSTAT = 4 historisk
fjernet	Fjernet
kondemnert	Kondemnert
nedlagt	Nedlagt
ombygd	Ombygd
planlagt	Planlagt
underArbeid	Under arbeid
iForfall	I forfall
planlagtIllustrert	Planlagt illustrert. Illustrert fremtidig situasjon (Tidligere SITSTAT = 1)
planlagtOgProsjektert	Planlagt, prosjektert. Prosjektert fremtidig situasjon (Tidligere SITSTAT = 2)
vedtatt	Vedtatt
tenktTattIBruk	Tenkt tatt i bruk

Egenskap:

Navn: medium

Definisjon: objektets beliggenhet i forhold til jordoverflaten

Eksempel:

På bro, i tunnel, inne i et bygningsmessig anlegg, etc.

Multiplisitet: 0..1

Verditype: Medium (code list)

Koder:

iBygning	I bygning/bygningsmessig anlegg
tidvisUnderVann	Tidvis under vann
påIsbre	På isbre
underIsbre	Under isbre
iLuft	I luft
påVannoverflaten	På vannoverflaten
påSjøbunnen	På sjøbunnen
påTerrenget	På terrenget/på bakkenivå. default
underTerrenget	Under terrenget
alltidIVann	Alltid i vann
underSjøbunnen	Under sjøbunnen

	ukjent	ukjent
Egenskap:		
Navn:	opphav	
Definisjon:	referanse til opphavsmaterialet, kildematerialet, organisasjons/publiseringskilde Merknad: Kan også beskrive navn på person og årsak til oppdatering	
Multiplisitet:	0..1	
Verditype:	CharacterString	
Egenskap:		
Navn:	digitaliseringsmålestokk	
Definisjon:	kartmålestokk registreringene/ datene er hentet fra/ registrert på Eksempel: 1:50 000 = 50000.	
Multiplisitet:	0..1	
Verditype:	Integer	
Egenskap:		
Navn:	prosesshistorie	
Definisjon:	beskrivelse av de prosesser som dataene er gått gjennom som kan ha betydning for kvaliteten og bruken av dataene Merknad: Prosesshistorie vil kunne inneholde informasjon om transformasjoner. Hva slags informasjon som angis er ofte gitt i andre standarder, f.eks kvalitet og kvalitetsikring.	
Multiplisitet:	0..*	
Verditype:	CharacterString	
Egenskap:		
Navn:	kopidata	
Definisjon:	angivelse av at objektet er hentet fra et kopidatasett og ikke fra originaldatasettet Merknad: Inneholder informasjon om når kopidatasettet ble kopiert fra originaldatasettet og hvem som er originaldataansvarlig	
Multiplisitet:	0..1	
Verditype:	Kopidata (data type)	
Egenskap:		
Navn:	informasjon	
Definisjon:	generell opplysning Merknad: mulighet til å legge inn utfyllende informasjon om objektet	
Multiplisitet:	0..*	
Verditype:	CharacterString	
Egenskap:		
Navn:	registreringsversjon	
Definisjon:	angivelse av hvilken produktspesifikasjon som er utgangspunkt for dataene	
Multiplisitet:	0..1	
Verditype:	Registreringsversjon (data type)	

Egenskap:	
Navn:	link
Definisjon:	referanse til et informasjonselement, enten lokalt eller globalt
Multiplisitet:	0..*
Verditype:	URI (data type)
Egenskap:	
Navn:	geodataeier
Definisjon:	rettighetshaver til datasettet/tjenesten
Multiplisitet:	0..1
Verditype:	CharacterString
Egenskap:	
Navn:	geodataprodusent
Definisjon:	organisasjon som har produsert datasettet/tjenesten
Multiplisitet:	0..1
Verditype:	CharacterString
Egenskap:	
Navn:	kontaktperson
Definisjon:	person som kan kontaktes i forbindelse med en forespørsel
Multiplisitet:	0..1
Verditype:	CharacterString
Egenskap:	
Navn:	høydeOverBakken
Definisjon:	objekts høyde over bakken Merknad: Kan være aktuelt i forbindelse med ulike typer objekter med utstrekning i høyde, slik som telefonstolper, gjerde, etc. Må brukes med forsiktighet og det må komme klart fram hvilke detalj av objektet eller objektets overbygning høyden relateres til.
Multiplisitet:	0..1
Verditype:	Real
Egenskap:	
Navn:	høydereferanse
Definisjon:	koordinatregistering utført på topp eller bunn av et objekt
Multiplisitet:	0..1
Verditype:	Høydereferanse (code list)
Koder:	topp: Høyden målt til toppen av objektet fot: Høyden målt til foten av objektet

Data Type: Identifikasjon

Identifikasjon
Definisjon:
Unik identifikasjon av et objekt i et datasett, forvaltet av den ansvarlige produsent/forvalter, og kan benyttes av eksterne applikasjoner som stabil referanse til objektet. Merknad 1: Denne objektidentifikasjonen må ikke forveksles med en tematisk objektidentifikasjon, slik som f.eks bygningsnummer.

Merknad 2: Denne unike identifikatoren vil ikke endres i løpet av objektets levetid, og ikke gjenbrukes i andre objekt.	
Type:	
Data Type	
Egenskap:	
Navn:	lokalId
Definisjon:	lokal identifikator av et objekt Merknad: Det er dataleverendørens ansvar å sørge for at den lokale identifikatoren er unik innenfor navnerommet.
Multiplisitet:	1
Verditype:	CharacterString
Egenskap:	
Navn:	navnerom
Definisjon:	navnerom som unikt identifiserer datakilden til et objekt, anbefales å være en http-URI Eksempel: http://data.geonorge.no/SentraltStedsnavnsregister/1.0 Merknad : Verdien for navnverom vil eies av den dataprodusent som har ansvar for de unike identifikatorene og må være registrert i data.geonorge.no eller data.norge.no
Multiplisitet:	1
Verditype:	CharacterString
Egenskap:	
Navn:	versjonId
Definisjon:	identifikasjon av en spesiell versjon av et geografisk objekt (instans)
Multiplisitet:	0..1
Verditype:	CharacterString

Data Type: Kopidata

Kopidata	
Definisjon:	
angivelse av at objektet er hentet fra en kopi av originaldata Merknad: Kan benyttes dersom man gjør et uttak av en database som ikke inneholder originaldataene.	
Type:	
Data Type	
Egenskap:	
Navn:	områdeId
Definisjon:	identifikasjon av område som dataene dekker Merknad: Kan angis med kommunenummer eller fylkesnummer. Disse bør spesifiseres nærmere.
Multiplisitet:	1
Verditype:	Integer
Egenskap:	

Navn:	originalDatavert
Definisjon:	ansvarlig etat for forvaltning av data
Multiplisitet:	1
Verditype:	CharacterString
Egenskap:	
Navn:	kopidato
Definisjon:	dato når objektet ble kopiert fra originaldatasettet Merknad: Er en del av egenskapen Kopidata. Brukes i de tilfeller hvor en kopidatabase brukes til distribusjon. Å kopiere et datasett til en kopidatabase skal ikke føre til at Oppdateringsdato blir endret. Eventuell redigering av data i et kopidatasett medfører ny oppdateringsdato, datafangstdato og/eller verifiseringsdato.
Multiplisitet:	1
Verditype:	DateTime

Data Type: Posisjonskvalitet

Posisjonskvalitet	
Definisjon:	beskrivelse av kvaliteten på stedfestingen. Merknad: Posisjonskvalitet er ikke konform med kvalitetsmodellen i ISO slik den er defineret i ISO19157:2013, men er en videreføring av tildligere brukte kvalitetsegenskaper i SOSI.
Type:	Data Type
Egenskap:	
Navn:	målemetode
Definisjon:	metode for måling i grunnriss (x,y), og høyde (z) når metoden er den samme som ved måling i grunnriss
Multiplisitet:	1
Verditype:	Målemetode (code list)
Lenke til ekstern kodeliste:	http://skjema.geonorge.no/SOSI/generelleKonsepter/generelleTyper/5.1/Målemetode
Egenskap:	
Navn:	nøyaktighet
Definisjon:	punktstandardavviket i grunnriss for punkter samt tverravvik for linjer Merknad: Oppgitt i cm
Multiplisitet:	0..1
Verditype:	Integer
Egenskap:	
Navn:	synbarhet
Definisjon:	hvor godt den kartlagte detalj var synbar ved kartleggingen
Multiplisitet:	0..1

Verditype:	Synbarhet (code list)	
Koder:	fulltSynligOgGjenfinnbarITerrenget	Fullt ut synlig/gjenfinnbar i terrenget Default
	dårligGjenfinnbarITerrenget	Dårlig gjenfinnbar i terreng. Forøvrig grei å innmåle. (Benyttes bl.a. for innmåling av ledninger på lukket grøft)
	middelsSynligIFlybilde	Middels synlig i flybilde/modell
	dårligSynligIFlybilde	Dårlig/ikke synlig i flybilde/modell

Egenskap:

Navn:	målemetodeHøyde
Definisjon:	metode for å måle høyden
Multiplisitet:	0..1
Verditype:	MålemetodeHøyde (code list)
Lenke til ekstern kodeliste:	http://skjema.geonorge.no/SOSI/generelleKonsepter/generelleTyper/5.1/MålemetodeHøyde

Egenskap:

Navn:	nøyaktighetHøyde
Definisjon:	nøyaktighet for høyden i cm
Multiplisitet:	0..1
Verditype:	Integer

Egenskap:

Navn:	maksimaltAvvik
Definisjon:	absolutt toleranse for geometriske avvik
Multiplisitet:	0..1
Verditype:	Integer

Data Type: Registreringsversjon

Registreringsversjon

Definisjon:

angir hvilken versjon av registreringsinstruksen som ble benyttet ved datafangst

Eksempel:

I et datasett kan det finnes objekter som er etablert fra ulike registreringsversjoner. For eksempel har registreringsinstruksen for objekttypen Takkant i FKB blitt endret fra SOSI/FKB-versjon 3.4 til versjon 4.0. Dersom en kommune ønsker å ajourføre Takkant for et delområde av kommunen etter FKB/SOSI-versjon 4.0, vil han etter ajourføring ha et kommunedekkende datasett der Takkant er registrert med forskjellig registreringsinstruks. I disse tilfellene kan det være nyttig å kunne skille på objektnivå hvilken registreringsversjon som er benyttet ved datafangst. Egenskapen kan benyttes til dette.

Type:

Data Type

Egenskap:

Navn:	produkt
Definisjon:	entydig navn på produktet i form av et kortnavn
Multiplisitet:	1
Verditype:	CharacterString

Egenskap:	
Navn:	versjon
Definisjon:	versjonsnummer
Multiplisitet:	1
Verditype:	CharacterString

Data Type: Endringsflagg

Endringsflagg	
Definisjon:	
endringsinformasjon om et objekt Merknad: Inntil videre vil hele objektet merkes med endringsflagget. I det videre arbeidet (framtidige versjoner) vil denne kunne utvides, f.eks ved å angi om endringen er knyttet til geometrien, egenskapene eller relasjoner	
Type:	
Data Type	
Egenskap:	
Navn:	typeEndring
Definisjon:	endringsstatus for objektet
Multiplisitet:	1
Verditype:	TypeEndring (code list)
Koder:	endret: Endret nytt: Nytt slettet: Slettet
Egenskap:	
Navn:	tidspunktEndring
Definisjon:	tidspunkt for endring av objektet
Multiplisitet:	1
Verditype:	DateTime

E.2 Symbol og punkt med tekst

Feature Type: Tekstobjekt

Tekstobjekt	
Definisjon:	
abstrakt objekttype som ved subtyping vil ta vare på tekstinformasjon fra alle hisoriske datasett på SOSI-format	
Description:	
This type is abstract.	
Type:	
Feature Type	

Egenskap:	
Navn:	tekstplassering
Definisjon:	plassering av formatert tekst, starter ikke med objektkoordinat
Multiplisitet:	0..1
Verditype:	GM_Primitive
Egenskap:	
Navn:	formatering
Definisjon:	formatering som er ulik for hvert objekt
Multiplisitet:	1
Verditype:	Tekstformatering (data type)
Egenskap:	
Navn:	streng
Definisjon:	uformatert originaltekst
Multiplisitet:	0..1
Verditype:	CharacterString

Feature Type: Symbolobjekt

Symbolobjekt
Definisjon:
abstrakt objekttype som ved subtyping vil ta vare på symbolinformasjon fra alle hisoriske datasett på SOSI-format
Description:
This type is abstract.
Type:
Feature Type

Egenskap:	
Navn:	symbolplassering
Definisjon:	plassering av symbolet
Multiplisitet:	0..1
Verditype:	GM_Point
Egenskap:	
Navn:	symbolisering
Definisjon:	formatering av symbolet
Multiplisitet:	1
Verditype:	Symbolformatering (data type)

Data Type: Tekstformatering

Tekstformatering
Definisjon:
presentasjonsegenskaper knytta til tekst
Type:
Data Type

Egenskap:	
Navn:	formatertStreng

Definisjon:	overføring av tekstutforming, basert på HTML - lignende koder. Merknad: Foreløpig er bare 3 formateringselementer implementert: Hevet/senket skrift ^{=hevet skrift,}=ikke lenger hevet skrift, _{=senket skrift,}=ikke lenger senket skrift streng: "TekstHevetVanligSenket" formatertStreng "Tekst^{Hevet}Vanlig_{Senket}" Eksempel på utskrift: TekstHevetVanligSenket Fet skrift (bold) <B>=fet skrift, </B>=ikke lenger fet skrift streng "TekstFetVanlig" formatertStreng "Tekst<B>Fet</B>Vanlig" Eksempel på utskrift: TekstFetVanlig Kursiv skrift (italic) <I>=kursiv, </I>=ikke lenger kursiv skrift streng: " TekstKursivVanlig" formatertStreng " Tekst<I>Kursiv</I>Vanlig" Eksempel på utskrift: TekstKursivVanlig
Multiplisitet:	0..1
Verditype:	CharacterString
Egenskap:	
Navn:	tekstdimensjon
Definisjon:	bokstavenes bredde og høyde i millimeter på kartet pr. bokstav Merknad: Høyde regnes fra bunnlinje til øvre kant. Se TREF. Merknad: Dersom bredde ikke er oppgitt, benyttes standard bredde for den gitte teksthøyden jfr. fonten. Eksempel: Eksempel: tekstdimensjon 6.2 4.0
Multiplisitet:	0..1
Verditype:	Tekstdimensjon (data type)
Egenskap:	
Navn:	tekstdimensjonTerreng

Definisjon:	bokstavenes høyde og bredde i meter i terrenget pr. bokstav.
Multiplisitet:	0..1
Verditype:	TekstdimensjonTerreng (data type)
Egenskap:	
Navn:	tekstReferansepunkt
Definisjon:	Tekstens referansepunkt er det stedet på teksten hvor en tekstplassering refererer seg til.
Multiplisitet:	0..1
Verditype:	TekstReferansepunkt (data type)
Egenskap:	
Navn:	tekstforskyvning
Definisjon:	forskyvning av startpunktet for visning av en tekststreng, enhet er meter i terrenget.
Multiplisitet:	0..1
Verditype:	Real
Egenskap:	
Navn:	tegnavstand
Definisjon:	avstanden mellom bokstavene i teksten, enhet er prosent
Multiplisitet:	0..1
Verditype:	Integer
Egenskap:	
Navn:	skriftkode
Definisjon:	produktavhengig koplingsnøkkel mot presentasjonsinformasjon
Multiplisitet:	0..1
Verditype:	Integer
Egenskap:	
Navn:	skrifttype
Definisjon:	angivelse av den skrifttype eller font som skal benyttes. Default skrifttype er ARIAL Merknad: For samiske tegn anbefales SK Sans Serif, nedlastbart fra Statens kartverks nettsider
Multiplisitet:	0..1
Verditype:	CharacterString
Egenskap:	
Navn:	referansemålestokk
Definisjon:	egenskap som beskriver hvilken målestokk (oppgitt som målestokkstall) denne teksten er redigert for, både størrelse og plassering. Kan benyttes for å velge hvilke tekster som skal tegnes ut i ulike målestokker.
Multiplisitet:	0..1
Verditype:	Integer

Data Type: Symbolformatering

Symbolformatering
Definisjon:
presentasjonsegenskaper knytta til symbolisering
Type:

Data Type	
Egenskap:	
Navn:	tekstdimensjon
Definisjon:	symbolets bredde og høyde i millimeter på kartet
Multiplisitet:	0..1
Verditype:	Tekstdimensjon (data type)
Egenskap:	
Navn:	tekstdimensjonTerreng
Definisjon:	symbolets bredde og høyde i meter i terrenget
Multiplisitet:	0..1
Verditype:	TekstdimensjonTerreng (data type)
Egenskap:	
Navn:	tekstReferansepunkt
Definisjon:	symbolets referansepunkt er det stedet på symbolet hvor en plassering refererer seg til.
Multiplisitet:	0..1
Verditype:	TekstReferansepunkt (data type)
Egenskap:	
Navn:	retningsvektor
Definisjon:	retningsvektor i terrenget
Multiplisitet:	0..1
Verditype:	Vector

Data Type: Tekstdimensjon

Tekstdimensjon	
Definisjon:	
bokstavenes eller symbolenes bredde og høyde i millimeter på kartet pr. bokstav. Høyde regnes fra bunnlinje til øvre kant. Merknad: Dersom bredde ikke er oppgitt, benyttes standard bredde for den gitte teksthøyden jfr. fonten.	
Type:	
Data Type	
Egenskap:	
Navn:	teksthøyde
Definisjon:	bokstavenes eller symbolenes høyde i millimeter på kartet pr. bokstav. Høyde regnes fra bunnlinje til øvre kant Eksempel: 6.2
Multiplisitet:	1
Verditype:	Real
Egenskap:	
Navn:	tekstbredde
Definisjon:	bokstavenes eller symbolenes bredde i millimeter på kartet pr. bokstav Eksempel: 4.0
Multiplisitet:	1
Verditype:	Real

Data Type: TekstdimensjonTerreng

TekstdimensjonTerreng	
Definisjon:	bokstavenes eller symbolenes høyde og bredde i meter i terrenget pr. bokstav.
Type:	Data Type
Egenskap:	
Navn:	teksthøyde
Definisjon:	Dimensjon i terrenget - høyde
Multiplisitet:	1
Verditype:	Real
Egenskap:	
Navn:	tekstbredde
Definisjon:	Dimensjon i terrenget - bredde
Multiplisitet:	1
Verditype:	Real

Data Type: TekstReferansepunkt

TekstReferansepunkt	
Definisjon:	Tekstens referansepunkt er det stedet på teksten hvor en tekstplassering refererer seg til. Merknad: Default er (i motsetning til tekst), midtpunkt. Grunnlinje er ikke tillatt angitt for symbol. Merknad: Hvis ikke andre verdier er oppgitt, er default plassering av TREF som følger: For tekst: tekstreferanseNord = 1, tekstreferanseØst = 0, dvs nedre venstre punkt til første bokstav. For Symbol: tekstreferanseNord = 1, tekstreferanseØst = 1, dvs midt symbol. Merknad: For tekstReferansepunkt knyttet til SYMBOL er det ikke lovlig å angi bunnlinje, denne benyttes bare for tekst.
Type:	Data Type
Egenskap:	
Navn:	tekstreferanseNord
Definisjon:	Tekstens referansepunkt nord
Multiplisitet:	1
Verditype:	TekstreferanseNord (enumeration)
Koder:	øvreKant: øvre kant av tekstfeltet midtlinje: linje midt i tekstfeltet bunnlinje: nedre kant av normalbokstaver grunnlinje: nedre kant av bokstaver som går under grunnlinja

Egenskap:	
Navn:	tekstreferanseØst
Definisjon:	Tekstens referansepunkt øst
Multiplisitet:	1
Verditype:	TekstreferanseØst (enumeration)
Koder:	

venstreKant	venstre kant av teksten
midtI	midten av teksten
høyreKant	høyre kant av teksten

E.3 Tekstlig beskrivelse av egenskaper med tydelige fellestrekk

Egenskaper med tydelige fellestrekk er egenskaper som ikke hører naturlig til et spesielt fagområde.

E.3.1 Avgrensningslinjer

Feature Type: Dataavgrensning

Dataavgrensning	
Definisjon:	
generell avgrensningslinje, f.eks. mellom datasett med ulik kvalitet, innhold eller detaljering	
Type:	
Feature Type	
Egenskap:	
Navn:	grense
Definisjon:	forløp som følger overgang mellom ulike fenomener
Multiplisitet:	1
Verditype:	Kurve

Feature Type: FiktivDelelinje

FiktivDelelinje	
Definisjon:	
linje for å dele opp store flateobjekter Merknad: En del produktspesifikasjoner benytter spesifikke fiktive delelinjer.	
Type:	
Feature Type	
Egenskap:	
Navn:	grense
Definisjon:	forløp som følger overgang mellom ulike fenomener
Multiplisitet:	1
Verditype:	Kurve

Feature Type: KantUtsnitt

KantUtsnitt	
Definisjon:	
avgrensning av et utsnitt	
Type:	
Feature Type	
Egenskap:	
Navn:	grense
Definisjon:	forløp som følger overgang mellom ulike fenomener
Multiplisitet:	1
Verditype:	Kurve

Feature Type: Temakartavgrensning

Temakartavgrensning	
Definisjon:	
avgrensningslinje for et temakart	
Type:	
Feature Type	
Egenskap:	
Navn:	grense
Definisjon:	forløp som følger overgang mellom ulike fenomener
Multiplisitet:	1
Verditype:	Kurve

E.3.2 Kartblad og rutenett

Feature Type: Kartblad

Kartblad	
Definisjon:	
dekning av et nærmere angitt geografisk område, ofte basert på en offentlig kartbladinndeling	
Type:	
Feature Type	
Egenskap:	
Navn:	område
Definisjon:	objektets utstrekning
Multiplisitet:	1
Verditype:	Flate
Egenskap:	
Navn:	karttype
Definisjon:	type kartbladinndeling
Multiplisitet:	0..1
Verditype:	Karttype (code list)

Koder:

m711ngo1948	Norge 1 til 50000_M711 i NGO1948
m711euref89	Norge 1 til 50000_M711 i EUREF89
turkart50000euref89	Turkart 1 : 50000 EUREF89
Norge250000ngo1948	Norge 1 til 250000 i NGO1948
Norge250000utm	Norge 1 til 250000 i UTM
øk500ngo1948	ØK Tekn kart 1til 500 NGO1948 Eksempel: CX035-05-50-4
øk1000ngo1948	ØK Tekn kart 1til 1000 NGO1948 Eksempel: CX035-1-60
øk5000ngo1948	ØK Tekn kart 1 til 5000 NGO1948 Eksempel: CX035-5-1
øk500euref89	ØK Tekn kart 1 til 500 EUREF89 Eksempel: 33-05-499-304-70-01
øk1000euref89	ØK Tekn kart 1 til 1000 EUREF89 Eksempel: 33-1-499-304-61
øk2000euref89	ØK Tekn kart 1 til 2000 EUREF89 Eksempel: 33-2-499-304-21
øk5000euref89	ØK Tekn kart 1 til 5000 EUREF89 Eksempel: 33-5-499-304-00
øk10000euref89	ØK Tekn kart 1 til 10000 EUREF89 Eksempel: 33-10-499-305
øk20000euref89	ØK Tekn kart 1 til 20000 EUREF89 Eksempel: 33-20-498-304
møre56A	Møre - NGO 56A NGO1948
møre56B	Møre - NGO 56B NGO1948
møre64A	Møre - NGO 64A NGO1948
møre64B	Møre - NGO 64B NGO1948
lokaltNettOslo	
lokaltNettBærum	
lokaltNettAsker	
lokaltNettLillehammer	
lokaltNettDrammen	
lokaltNettBergen_Askøy	
lokaltNettTrondheim	
lokaltNettBodø	
lokaltNettKristiansund	
lokaltNettÅlesund	

Egenskap:

Navn: målestokk

Definisjon: forhold mellom en avstand på et kart og den tilsvarende avstand i terrenget, angitt som målestokkstall

Merknad: Målestokk 1:100 000 angitt som 100000

Multiplisitet: 0..1

Verditype: Integer

Egenskap:

Navn:	kartbladindeks
Definisjon:	offisiell kartbladreferanse
Multiplisitet:	0..1
Verditype:	CharacterString
Egenskap:	
Navn:	kartbladnavn
Definisjon:	ord som noen eller noe kalles ved
Multiplisitet:	0..1
Verditype:	CharacterString
Assosiasjonsrolle:	
Navn:	hjørne
Multiplisitet:	4
Verditype:	Kartbladhjørne (feature type)
Restriksjon:	
KanAvgrensesAv Kartbladkant,KartbladkantUTM: Kan avgrenses av geometriene til Kartbladkant og KartbladkantUTM.	

Feature Type: Kartbladhjørne

Kartbladhjørne	
Definisjon:	
hjørne i en kartbladkant	
Type:	
Feature Type	
Egenskap:	
Navn:	posisjon
Definisjon:	sted som objektet eksisterer på
Multiplisitet:	1
Verditype:	Punkt

Feature Type: Kartbladkant

Kartbladkant	
Definisjon:	
avgrensningslinje for et kart som dekker et nærmere angitt geografisk område, ofte basert på en offentlig kartbladinndeling	
Type:	
Feature Type	
Egenskap:	
Navn:	grense
Definisjon:	forløp som følger overgang mellom ulike fenomener
Multiplisitet:	1
Verditype:	Kurve
Egenskap:	
Navn:	karttype
Definisjon:	type kartbladinndeling
Multiplisitet:	0..1
Verditype:	Karttype (code list)

Feature Type: KartbladkantUTM

KartbladkantUTM	
Definisjon:	
avgrensningslinje for et kart i henhold til kartbladinndelingen for UTM	
Subtype av:	
Kartbladkant	
Type:	
Feature Type	
Egenskap:	
Navn:	grense
Definisjon:	forløp som følger overgang mellom ulike fenomener
Multiplisitet:	1
Verditype:	Kurve
Egenskap:	
Navn:	karttype
Definisjon:	type kartbladinndeling
Multiplisitet:	0..1
Verditype:	Karttype (code list)

Feature Type: Rutenett

Rutenett		
Definisjon:		
teknisk inndeling av et geografisk område i ruter		
Type:		
Feature Type		
Egenskap:		
Navn:	grense	
Definisjon:	forløp som følger overgang mellom ulike fenomener	
Multiplisitet:	1	
Verditype:	Kurve	
Egenskap:		
Navn:	rutenettype	
Definisjon:	ruter basert på geografiske eller projiserte koordinater, bestående av horisontale og vertikale linjer	
Multiplisitet:	0..1	
Verditype:	Rutenettype (code list)	
Koder:	ngoLokalAkse	rektangulært rutenett i aktuell projeksjon, basert på NGO48
	euref89LokalUtmSone	rektangulært rutenett i aktuell projeksjon, basert på Euref89
	gradnett	tilordning av ellipsoiden av den fysiske jord i form av et nettverk med utgangspunkt i lengde- og breddegrad
	annetLokalt	

Feature Type: Rutenettflate

Rutenettflate	
Definisjon:	
flate i et rutenett Merknad: Brukes blant annet for griddede data.	
Type:	
Feature Type	
Egenskap:	
Navn:	område
Definisjon:	objektets utstrekning
Multiplisitet:	0..1
Verditype:	Flate
Egenskap:	
Navn:	posisjon
Definisjon:	sted som objektet eksisterer på
Multiplisitet:	0..1
Verditype:	Punkt
Egenskap:	
Navn:	rutenettype
Definisjon:	ruter basert på geografiske eller projiserte koordinater, bestående av horisontale og vertikale linjer
Multiplisitet:	0..1
Verditype:	Rutenettype (code list)
Constraints:	
KanAvgrensesAv Rutenett: Kan avgrenses av geometriene til Rutenett.	

Feature Type: Sonedele

Sonedele	
Definisjon:	
teknisk inndeling av et geografisk område i soner, basert på UTM kartbladinndeling	
Type:	
Feature Type	
Egenskap:	
Navn:	senterlinje
Definisjon:	forløp som følger objektets sentrale del
Multiplisitet:	1
Verditype:	Kurve
Egenskap:	
Navn:	sonetype
Definisjon:	teknisk inndeling av et geografisk område i soner, basert på en offentlig kartbladinndeling
Multiplisitet:	0..1

Verditype:	Sonetype (code list)
Koder:	euref89utm: Geometri-egenskap ngo: ed50utm: møre64:

E.3.3 Retning

Data Type: Retning

Retning	
Definisjon:	linjestykke i planet med retning
Type:	Data Type
Egenskap:	
Navn:	retningsverdi
Definisjon:	generelt element med angivelse av retning
Multiplisitet:	1
Verditype:	Real
Egenskap:	
Navn:	retningsenhet
Definisjon:	enhet for retning
Multiplisitet:	1
Verditype:	Retningsenhet (code list)
Koder:	gon: 400 graders deling med positiv retning med sola grader: 360 graders deling med positiv retning med sola radianer: Radianer med positiv retning med sola
Egenskap:	
Navn:	retningsreferanse
Definisjon:	referansesystem for retning
Multiplisitet:	1
Verditype:	Retningsreferanse (code list)
Koder:	lokal: magnetiskNord: santNord: (default)

Utgitt av:
Statens kartverk

ISBN 978-82-7945-554-7