SOSI Del 1: Retningslinjer for modellering i UML

INNHOLDSFORTEGNELSE

SOSI Del 1: Retningslinjer for modellering i UML

1 Introduksjon

1.1 Bakgrunn

Bakgrunnen for dokumentet er at dagens nasjonale SOSI standard vil gjennomgå endringer for å kunne være konform med ISO 19100 serien av standarder. Referansegruppe K176 har en klar strategi på å konvergere SOSI mot internasjonale standarder, da spesielt ISO 19100 serien. Modelleringsreglene i dette dokumentet er spesielt rettet mot dette, men vil også være retningsgivende for generell modellering av geografisk informasjon.

1.2 Formål

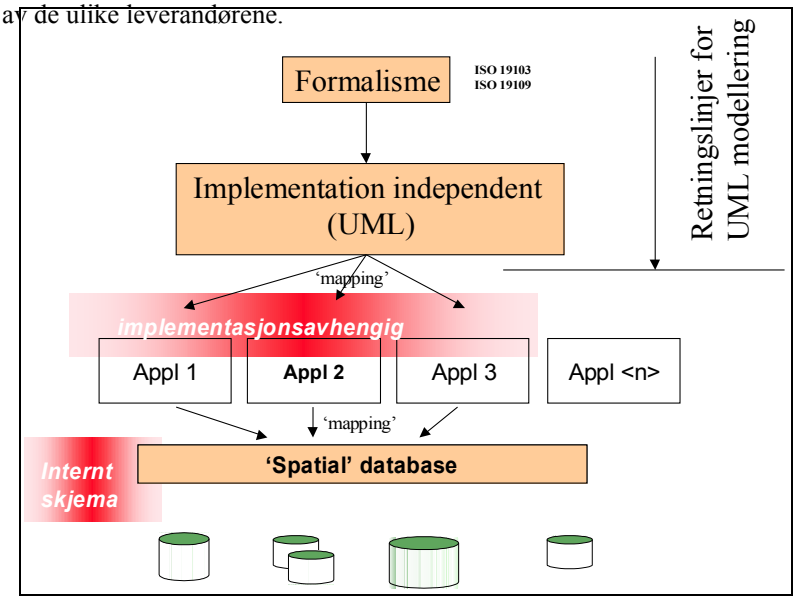
Formålet med dokumentet er å gi retningslinjer for implementasjonsuavhengig modellering av geografiske objekter i et applikasjonsskjema. Spesifikasjon av systemløsninger, samt modellering for en systemavhengig implementasjon ligger utenfor rammen av retningslinjene.

Dokumentet gir ingen fullstendig innføring i UML. Til dette anbefales generell UML-litteratur. Dokumentet forsøker å gi regler og eksempler på modellering av geografisk informasjon basert på regler og retningslinjer i de internasjonale standardene samt erfaring med modellering av SOSI Generell objektkatalog. I tillegg til eksempler og regler er det tatt med formell syntaks for enkelte av de mest grunnleggende konseptene.

Et applikasjonsskjema (informasjonsmodell) beskrevet i UML har to formål:
Den skal gi en korrekt menneskelig forståelse av objekter, egenskaper, relasjoner og eventuelt operasjoner innenfor sitt fagområde. (Med fagområde menes her vegnett, jernbane, topografi, etc.)
Den skal være leselig av en datamaskin, for å kunne anvende automatiske rutiner i henhold til implementasjon, dataforvaltning og utveksling.

UML er et omfattende språk der det samme kan beskrives på flere måter. Det som omtales her er anbefalinger, andre måter å modellere på kan være korrekt i henhold til de normative referansene (ikke minst ISO 19501 UML). Imidlertid er det ønskelig å modellere mest mulig ensartet, for å sikre interoperabilitet.

UML er et implementasjonsuavhengig språk, modeller beskrevet ved hjelp av UML må 'mappes' mot ulike implementasjoner på en gitt plattform med en spesifikk teknologi. Disse retningslinjene beskriver ikke systemimplementasjoner, dette må gjøres av de respektive systemleverandører, på samme måte som de i dag gjør i forhold til SOSI. Forskjellen nå ligger i at vi legger en internasjonalt akseptert objektmodell (general feature model) til grunn for selve modelleringen, med forventninger om at denne også understøttes av de ulike leverandørene.



Figur 1.Gyldighetsområde for retningslinjene

For å sikre at modellene enkelt lar seg implementere, legger vi stor vekt på i disse retningslinjene å forholde oss til de mest vanlige mekanismene i UML, men uten å legge skjul på at det finnes en rekke andre mekanismer.

Disse retningslinjene omfatter UMLs static structure diagram med de retningslinjer som er definert i ISO 19103 Conceptual Schema Language. Med static structure diagrams mener vi de diagrammer eller den delen av UML som brukes for å beskrive strukturen til informasjon i et system.
Det er en målsetting at dokumentet skal være retningsgivende for det videre arbeidet med SOSI Generell objektkatalog og

produktspesifikasjoner basert på denne, både i forbindelse med temadataspesifikasjoner, FKB-produktspesifikasjoner og andre produktspesifikasjoner for geografisk informasjon. Dokumentet gir imidlertid ikke føringer for hvor raskt denne endringen skal skje. Retningslinjene er i stor grad anvendt for gjeldende versjon av SOSI, og er førende for det videre arbeidet framover, både i utviklingen av nye objektkataloger og produktspesifikasjoner, samt for konvergeringen mot internasjonale standarder i regi av ISO/TC 211.

1.3 Referanser

ISO 19103, Geographic information — Conceptual schema language [CSL]
ISO 19104, Geographic information — Terminology
ISO 19107, Geographic information — Spatial Schema
ISO 19108, Geographic information — Temporal Schema
ISO 19109, Geographic information — Rules for application schema [RAS]
ISO 19110, Geographic information — Methodology for feature cataloguing.
ISO 19113, Geographic information — Quality Principles
ISO 19115, Geographic information — Metadata
ISO 19136, Geographic information — Geography Markup Language
ISO 11179, part 5, *Naming and Identification Principles for Data Elements.*
ISO 19501, UML (Unified Modelling Language) Specification version 1.4

2 Ord, termer og forkortelser

2.1 Definisjoner

I dette kapitlet er de fleste ord og termer definert, 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.

Siden dokumentet blant annet gir føringer for hvordan en generell beskrivelse av objekter avbildet fra en nærmere angitt virkelighet skal modelleres, er det tatt med begreper fra både generell modellering og det skjemaspråk som er valgt innenfor geografisk informasjon, UML.

applikasjonsskjema
et 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]

aggregering (svak aggregering)
en spesiell form for assosiasjon mellom et generelt element og mer spesifikt element. Angir et strukturelt forhold mellom et 'hele' og de respektive 'deler'

aggregation
a special form of association that specifies a whole-part relationship between the aggregate (the whole) and a component (the part)

assosiasjon
forhold mellom objekter i form av en vanlig assosiasjon, aggregering eller komposisjon

feature association
relationship between features
NOTE 1 A feature association may occur as a type or an instance. Feature association type or feature association instance is used when only one is meant.
NOTE 2 Feature associations include aggregation of features.

beskrankninger
semantisk betingelse eller restriksjon representert som et uttrykk i form av tekst eller OCL

constraint
semantic condition or restriction represented as an expression. [UML] [ISO 19103]

datatype
spesifikasjon av et lovlig verdidomene
Eksempel: Integer, Real, Boolean, String, Date

data type
specification of a legal value domain and legal operations on values in this domain
EXAMPLE: Integer, Real, Boolean, String, Date and GM_Point.
NOTE A data type is identified by a term, e.g. Integer [ISO 19115, ISO 19118]

klasse
beskrivelse av et sett objekter som deler de samme attributter, operasjoner, metoder, forhold og semantikk
Merknad: Med objekter her menes ikke nødvendigvis kun forekomster av objekttyper

class

*description of a **set** of **objects** that share the same attributes, operations, methods, relationships, and semantics [ISO/TS 19103]*
Note A class may use a set of interfaces (named set of operations that characterize the behavior of an element) to specify collections of operations it provides to its environment. The term was first used in this way in the general theory of object oriented programming, and later adopted for use in this same sense in UML.

kodeliste
spesifiserte verdier i et åpent verdidomene
Merknad: Dette betyr at verdidomenet kan endre seg. Eksempelvis vil verdidomenet til mange av egenskapene i SOSI være definert i form av en kodeliste.

codelist
Can be used to describe a more open enumeration. Codelist is a flexible enumeration that uses string values through a binding of the Dictionary type key and return values as string types; e.g. Dictionary (String, String). Code lists are useful to for expressing a long list of potential values. If the elements of the list are completely known, an enumeration should be used; if the only likely values of the elements are known, a code list should be used. Enumerated code lists may be encoded according to a standard, such as the ISO standard for two-letter country codes or for two-letter language code. Code lists are more likely to have their values exposed to the user, and are therefore often mnemonic. Different implementations are likely to use different encoding schemes (with translation tables back to other encoding schemes available).
[ISO CD 19119 CSL]

komposisjon (sterk aggregering)
en streng aggregering mellom modellelementer der "delen" kun eksisterer i levetiden til "helheten", dvs. at "delen" ikke har eksistens på egen hånd

composition
A form of aggregation which requires that a part instance be included in at most one composite at a time, and that the composite object is responsible for the creation and destruction of the parts. Composition may be recursive.
Synonym: composite aggregation.

lukket kodeliste
spesifiserte verdier i et lukket verdidomene, f.eks. sann/usann for boolske operatorer
Merknad: Denne benyttes når samtlige verdier i verdidomenet er spesifisert.

enumeration
A data type whose instances form a list of named literal values. Enumeration means a short list of well-understood potential values within a class. Classic examples are Boolean that has only 2 (or 3) potential values TRUE, FALSE, (and NULL). Most enumerations will be encoded as a sequential set of Integers, unless specified otherwise. The actual encoding is normally only of use to the programming language compilers.
[ISO CD 19119 CSL]

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.

objektegenskap
egenskap
en objekttypes eller et objekts karakteristikk
Merknad: En egenskap har ett navn og en datatype.

feature attribute
characteristic of a feature
NOTE 1 A feature attribute may occur as a type or an instance. Feature attribute type or feature attribute instance is used when only one is meant.

NOTE 2 A feature attribute type has a name, a data type and a domain associated to it. A feature attribute instance has an attribute value.

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

objektoperasjon
operasjon
en abstraksjon av noe man kan utføre på et objekt og som deles av alle objekter av samme objekttype
Eksempel: En operasjon på objekttypen "Dam" er å heve vannstanden. Resultatet av operasjonen er å heve vannstanden og øke mengden vann i et "reservoar".

feature operation
operation that may be performed upon or by all instances of a feature type [ISO 19110]

EXAMPLE 1 *A feature operation upon the feature type "dam" is to raise the dam. The results of this operation are to raise the height of the "dam" and the level of water in a "reservoir".*
EXAMPLE 2 *A feature operation by the feature type "dam" might be to block vessels from navigating along a watercourse.*

objekttype
geografisk objekttype
en klasse av objekter med felles egenskaper, forhold mot andre objekttyper og funksjoner [SOSI 1_2]

feature type
abstraction of real world phenomena [ISO 19101]

subtype
i et generaliseringsforhold spesialiseringen av en annen type, supertypen

subtype
In a generalization relationship the specialization of another type, the supertype. See generalization. Contrast: supertype. [UML Semantics, version 1.5]

supertype
i et generaliseringsforhold spesialiseringen av en annen type, subtypen

supertype
In a generalization relationship the generalization of another type, the subtype. See generalization Contrast: subtype. [UML Semantics, version 1.5]

UML-pakke
mekanisme for å gruppere modellelementer
Eksempelvis vil ett SOSI-fagområde danne en pakke. Pakker kan nestes, slik at en pakke kan inneholde en annen pakke. En pakke kan benytte innhold i en annen pakke, dette beskrives ved et avhengighetsforhold. Se avhengighet (dependency).

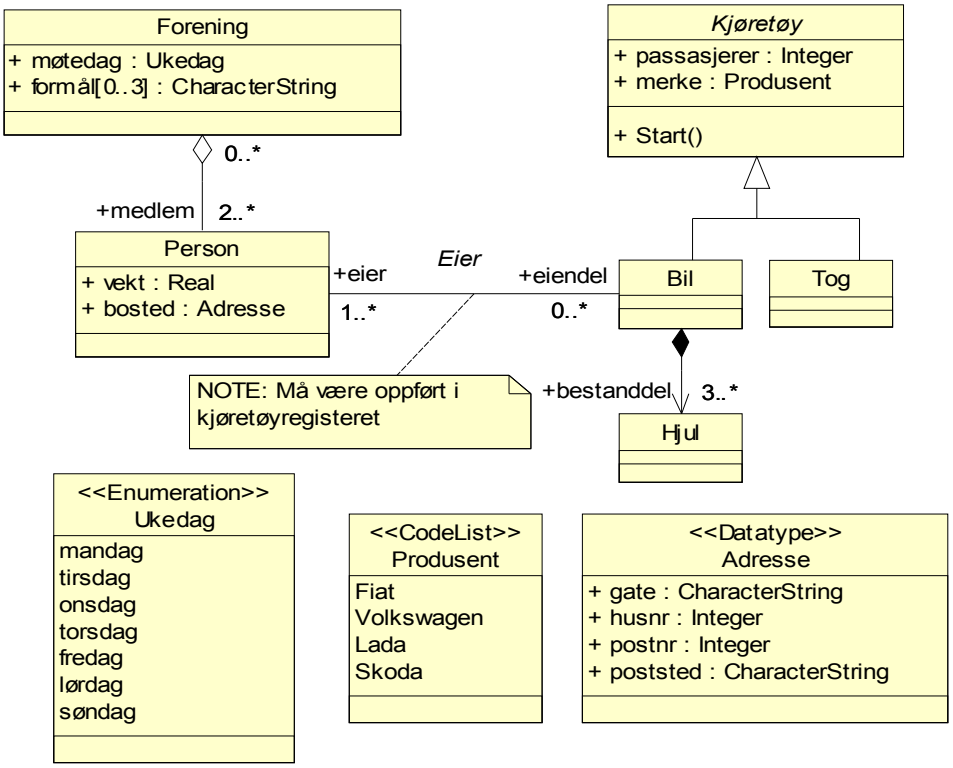
package
general-purpose mechanism for organizing elements into groups. Packages may be nested within other packages. Both model elements and diagrams may appear in a package. [ISO 19103 terminology]

2.2 Forkortelser

RAS - ISO 19109, Geographic information — Rules for application schema
MFC - ISO 19110, Geographic information — Methodology for feature cataloguing
CSL - ISO 19103, Geographic information — Conceptual schema language
GML - ISO 19136, Geographic information — Geography Markup Language
GFM - General Feature Model
UML - Unified Modeling Language
OCL - Object Constraint Language
XML - Extensible Markup Language
XMI - XML Metadata Interchange

3 Introduksjon til UML

Eksempel på noen av de viktigste hovedelementene i et UML klassediagram.



Figur 2.Introduksjon til UML

Objekttyper:
Interesseområdet kan deles inn i objekttyper som har samme egenskaper, assosiasjoner og oppførsel. Forening, Person, Bil, Tog, Hjul.

Egenskaper:
En forening kan ha minst en, og inntil tre (av egenskapen) formål
Foreningen har en fast møtedag, som er tatt ifra en lukket liste over ukedager.
En Person kan ha en angitt egenskapsverdi for sin vekt, av typen desimaltall.
Et kjøretøy kan ta et antall passasjerer, av typen heltall.
Et kjøretøy kan ha et fabrikat, med verdi tatt ifra en åpen liste.

Åpen aggregering:
En Forening består av medlemmer som er Personer.
Foreninger må ha minst to medlemmer. (2..*).
En Person kan være medlem av flere Foreninger.

Vanlig assosiasjon:
Personer Eier Biler. Assosiasjonsnavn er i verbs form. Null eller flere biler (0..*) kan ha rollen som en eiendel til en Person. En Bil må ha minst en Person (1..*) som eier. Roller skal være i substantivsform.

Komposisjon:
En Bil har (som eide komponenter) minimum tre Hjul. Pilen angir at instanser av Hjul ikke kan finne ut hvilken Bil de sitter på.

Subtyping:
Bil er en subtype av Kjøretøy. (Arver egenskaper, assosiasjoner og operasjoner).
Kjøretøy er supertype til Bil og Tog. Abstrakte supertyper instansieres ikke. (Navnet er i kursiv).

Oppførsel:
Et Kjøretøy kan utføre en operasjon Start().

Lukket liste – Enumeration:
Ukedag er en lukket liste hvor ingen nye verdier kan legges inn.

Åpen liste – CodeList:
Produsent er ei liste over kjente produsenter, åpen for nye produsenter.

Datatype:
Adresse er en egendefinert datatype som beskriver verdidomenet til bostedet for en Person.

Note:
Noten koblet til eierskapet mellom Person og Bil er en beskrankning på assosiasjonen.

4 Inndeling av et interesseområde i objekttyper

4.1 Innledning

Det har fra flere hold fremkommet ønske om å få retningslinjer for hva som er geografiske objekter.
En kunne selvsagt ha én objekttype 'geografiskObjekt', med alle egenskaper som i detalj beskriver ethvert objekt. Men dette er lite ønskelig. På den annen side kan vi også la hvert enkelt grensemerke bli en egen objekttype, eksempelvis 'bolt i rør', 'kors i fjell', etc., noe som vil resultere i et stort antall objekttyper.

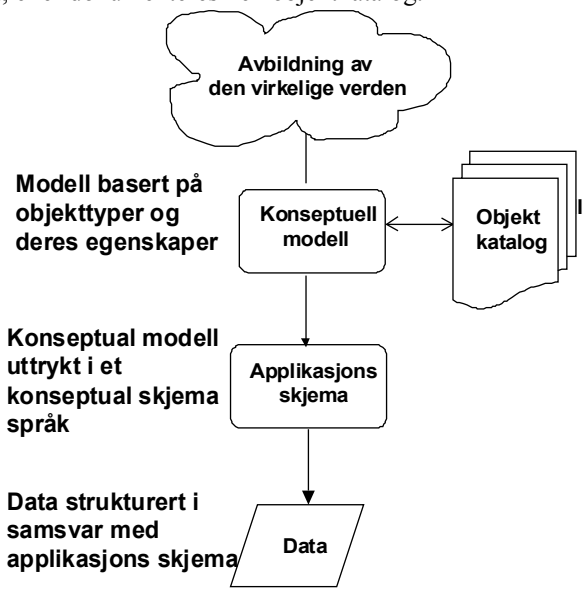
ISO 19109 uttrykker følgende:
The classification of real world phenomena as features depends on their significance to a particular universe of discourse.

Definer ditt interesseområde ('universe of discourse'). En må erkjenne om en virkelig modellerer den virkelige verden og ikke sine gamle datastrukturer, men skal benytte objekttyper som en kjenner igjen innenfor de respektive domener.
Sjekk om det allerede finnes standarder som dekker samme eller deler av interesseområdet.
Objekter med felles definisjon og samme egenskaper er kandidater for en objekttype (klasse). Dersom en objekttype blir vag og favner mange fenomener som også har egne dagligdagse navn, bør denne vurderes splittet opp i henhold til disse dagligdagse navnene.
En bør først identifisere de viktigste objekttypene og beskrive disse. Mer perifere objekttyper vil oftest forholde seg til de mest sentrale.
Der samme egenskaper og roller gjentas for flere objekttyper, må en vurdere å lage egne datatyper, eller å opprette en supertype som flere andre objekttyper arver samme felles egenskaper og roller ifra.
Er det behov for å generalisere data, kan en benytte en instansierbar (ikke abstrakt) supertype.

4.2 Avbildning av den virkelige verden

Enhver modellering av geografiske objekter tar utgangspunkt i en avbildning av den virkelige verden.

4.2 viser prosessen fra en avbildning av den virkelige verden ('universe of discourse') til et geografisk datasett. Definisjonen av objekttyper og deres egenskaper innenfor dette applikasjonsområde, avledes av denne avbildningen av den virkelige verden. Objekttypene kan hentes fra, eller dokumenteres i en objektkatalog.



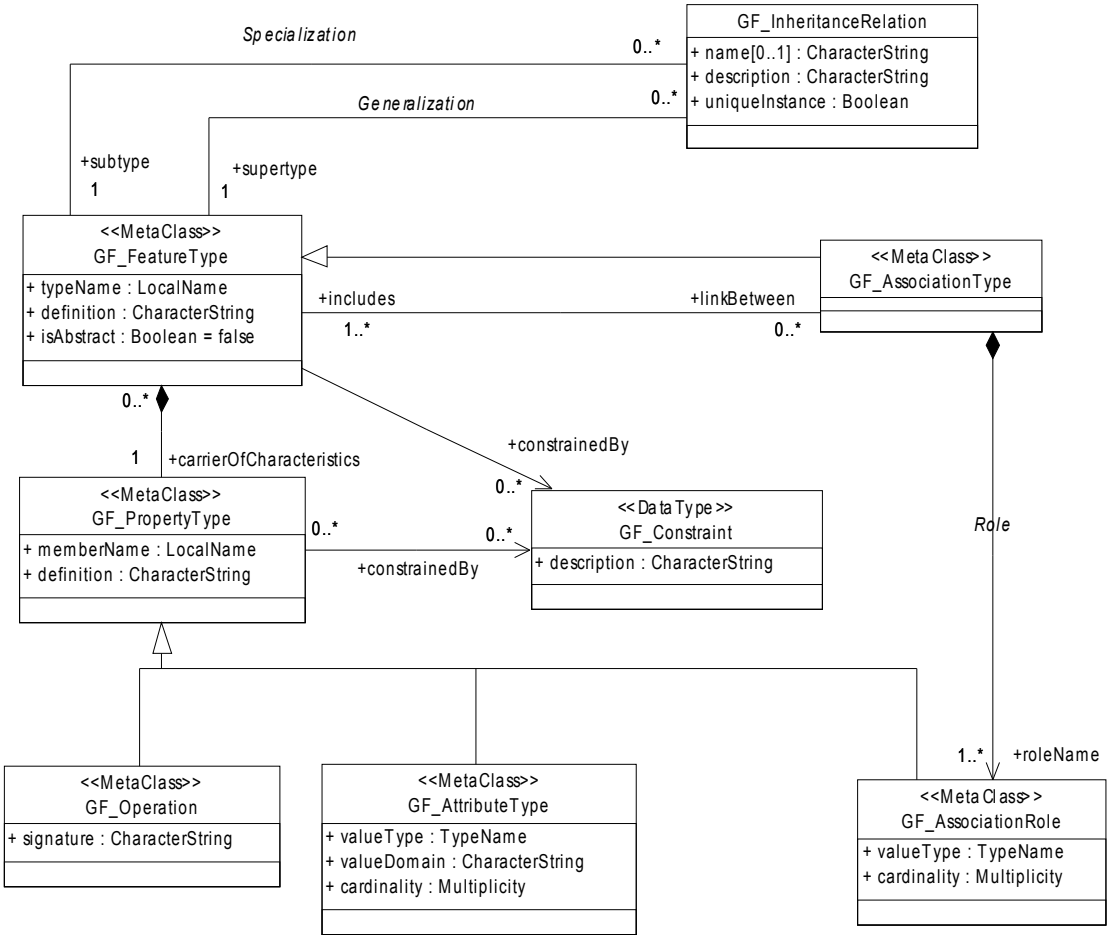
Figur 3.Avbildning av den virkelige verden

4.3 Generell objektmodell (General Feature Model)

ISO 19109 spesifiserer en generell objektmodell (GFM) for beskrivelse av geografiske objekter.
GFM er en modell av konsepter nødvendige for å klassifisere et bilde (avbildning) av den virkelige verden. Den er uttrykt i UML. UML har sin egen modell av konsepter (metamodell). Siden både GFM og UML's metamodell beskriver klassifikasjoner, er konseptene ganske like. Men det er en stor forskjell; konseptene i GFM danner basis for klassifikasjon av objekttyper (metamodell for objekttyper) mens UML's metamodell danner basis for all slags modellering, ikke bare objekttyper.

De objektene vi klassifiserer kalles objekttyper, forholdene mellom objekttyper kalles forhold, nærmere presisert gjennom ulike typer assosiasjoner og subtyper/supertyper. Objekttyper har attributter som kalles objektegenskaper eller bare egenskaper, objektoperasjoner og assosiasjoner angitt gjennom rollenavn. Alle disse konseptene er beskrevet som UML metaklasser i GFM.

Konklusjon:
GFM er en metamodell for definisjon av "objekttype" samt definerer en struktur for objekttyper, mens UML er en metamodell for selve applikasjonsskjema. Siden det her er snakk om et applikasjonsskjema for objekttyper, må GFM's struktur taes hensyn til ved modellering av applikasjonsskjemaet.



Figur 4.Utdrag fra den generelle objektmodellen (ISO 19109)

4.3 viser et utdrag av den generelle objektmodellen som er definert i ISO 19109 Rules for application schema, for komplett modell med alle detaljer henvises det til denne.

4.4 Regler

Reglene under er utdrag fra ISO 19109 Rules for Application Schema (RAS). Det er tatt med henvisning til kapitler i denne standarden samt ISO 19103 Conceptual Schema Language (CSL).

De følgende regler er konforme med ISO 19109 RAS og ISO 19103 CSL med unntak spesifisert i Annex A.

1.	Oversikt over datamodellen Oversikten lages ved å benytte UML's pakkemekanisme. Bruk UML avhengigheter for å vise hvilke andre standardpakker som modellen benytter seg av. Bruk noter i UML til å vise hvilke definisjoner som benyttes i de andre modellene.	Ref. RAS 8.2.5
2.	Identifisering av objekttyper og etablering av UML klasser Studèr den del av virkeligheten som skal modelleres ('Universe of Discourse') og identifiser de klasser av geografiske objekter (objekttyper) som tilhører applikasjonen. Konkrete geografiske objekttyper (eksempel: Veg, Bygning,..) Abstrakte (faglige) geografiske objekttyper (eksempel: Veglenke, Vegnode,…) Objekttyper modelleres som UML klasser.	Ref. RAS 8.1 Ref. RAS 8.3
3.	Supertype / Subtype Organiser objekttypene i en struktur hvor generalisering og spesialisering av klassene framkommer. Kriterier for dette er: En objekttype er en subtype av en annen. (Se eksempel: Europaveg, Riksveg, … kan modelleres som subtyper av veg). Flere objekttyper har flere felles egenskaper. Det kan være nødvendig å introdusere nye klasser som representer supertypen. (Eksempel: KilometrertVeg).	
4.	Egenskaper Egenskaper til objekttyper modelleres som UML attributter i de klasser som representerer objekttypen. Dette gjelder også egenskaper som refererer til klasser i andre standardmodeller.	
5.	Assosiasjoner mellom objekttyper Assosiasjoner mellom objekttyper modelleres som UML assosiasjon mellom tilhørende klasser. Roller skal angis (eksempel: Rollen "sammenknytning" i relasjonen mellom Vegnode og Veglenke). Multiplisitet skal angis (eksempel: i assosiasjonen mellom Vegnode og Veglenke er en Veglenke alltid knyttet til 2 Vegnoder).	Ref. RAS 8.3
6.	Kvalitet og andre metadataelementer Kvalitet knyttet direkte til en objekttype modelleres i form av en egenskap i klassen som representerer objekttypen. (Se eksempel: Egenskapen "klassifikasjonsnøyaktighet"). Stedfestingsnøyaktighet knyttes til geometri ved at ISO geometritypen subtypes og at kvalitetsegenskapen legges inn som egenskap.	Ref. RAS 8.5 Ref. RAS 8.5.3 Ref. RAS 7.4.2 RAS 8.2.6 fig.11 Ref RAS 8.5.3.1 Fig 15 Ref: RAS 8.7.6

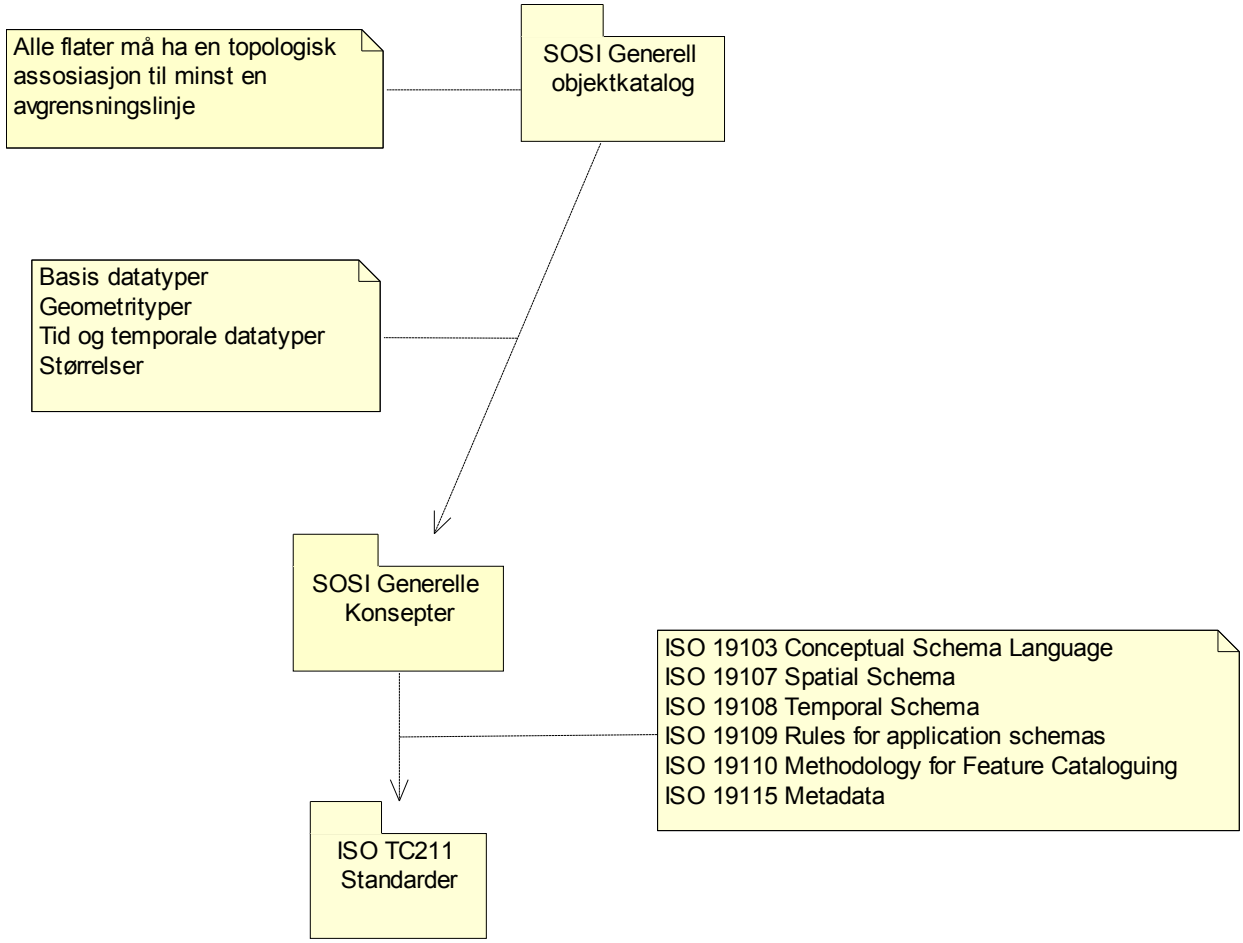
	Kvalitet knyttet til en egenskap modelleres ved at egen klasse etableres, hvor både egenskapen for objekttypen og dens kvalitetsinformasjon modelleres inn som egenskaper i denne klassen, som 'stereotypes' 'dataType'.	Ref. RAS 8.5.3
7.	Geometri Geometri modelleres i applikasjonsmodellen som egenskaper til objekttyper, med bruk av de klasser som er definert i den norske geometriprofilen (eksempel: posisjon: Punkt) Felles geometri, dvs. objekttyper deler geometri-beskrivelser modelleres ved å bruke Punkt, Kurve og Flate. Hvis objekttyper har en forretningsregel seg imellom, skal dette modelleres. Hvis sammenhengen kun er deling av geometrisk forløp så ivaretaes det i geometrimodellen.	Ref. RAS 8.7 Ref. RAS 8.7.2 Ref. RAS 8.7.5
8.	Erstatt underforståtte sammenhenger med eksplisitt beskrivelse Assosiasjoner som ofte ligger i sammenhenger på geometrinivå, kan komme til uttrykk i modellen (eksempel: I gamle VBASE var Vegnode implisitt beskrevet via KP i SOSI data, samt PTEMA langs med en linje). Unngå egenskaper som bærer mer enn en opplysning.	
9.	Definisjon av verdidomener: Kodeliste [CodeList] Brukes når man vil definere et verdidomene som man regner med vil kunne utvides (eksempel: Kommunenummer).	Ref. CSL 7.4.3
10.	Definisjon av verdidomener: Lukket kodeliste [Enumerasjon] Brukes til å definere et gitt lukket verdidomene for en gitt egenskap (eksempel: Dag i ukedag). Dersom en applikasjon bare skal utnytte et gitt subsett av et større sett av en verdi domene, etableres en egen enumerasjon som et subsett hvor lovlige verdier angis (eksempel: Vegmedium som spesialisering av Medium).	Ref. CSL 7.4.3
11.	Definisjon av verdidomener: 'Gazetteer' Brukes når det er verdier som ligger i andre databaser (f.eks. Postnr, Adresse). Referanser til 'Gazetteer' skal benytte SI_LocationInstance som er definert i Gazetteer Schema.	Ref. RAS 8.10
12	Definisjon av enheter Brukes for å presisere måledomene(enheter) for egenskaper, f.eks. med International System of **Units** (SI)	Ref. CSL 7.4.3
13	Beskrive beskrankninger (Constraints) Beskrankninger knyttet til egenskaper og relasjoner beskrives i "noter" og knyttes til det element i modellen som den beskriver (eksempel: Note knyttet til Veglenke). Beskrankninger skal beskrives om mulig i OCL (eksempel: Innhold i note knyttet til Veglenke).	
14	Navngiving Bruk regler for små og store bokstaver som beskrevet i kapittel 6.3 og CSL 7.9	Ref. CSL 7.9

4.5 Eksempel UML-modell basert på SOSI Jernbane

Denne modellen er en eksempelmodell for å illustrere konseptene beskrevet i kapittel 5.1, og må ikke regnes som komplett modell for SOSI Jernbane. Vi viser blant annet hvordan oppbygningen av SOSI Objektkatalog kan struktureres ved hjelp av pakker samt hvordan innholdet i pakkene kan presenteres.

4.5.1 SOSI Objektkatalog overordnet pakkestruktur og avhengigheter

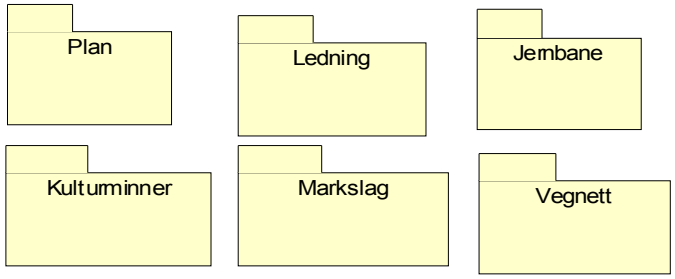
Figuren under viser det øverste nivå i pakkestrukturen for SOSI Generell Objektkatalog. Vi ser at pakken SOSI Generell Objektkatalog er avhengig (stiplet pil) av SOSI Generelle konsepter. Noten koblet til avhengigheten sier hvilke konkrete elementer en pakke bruker fra pakken den er avhengig av. Her ser vi at SOSI Generelle konsepter definerer geometrityper (Flate, Kurve ,etc.) og metadata som brukes av pakken SOSI Generell Objektkatalog. Videre avhenger SOSI Generelle konsepter igjen av standardene definert i ISO TC211, typisk eksempel er ISO19107 Spatial Schema.



Figur 5.Pakker og avhengigheter SOSI Standard UML-modell

4.5.2 Innholdet i pakken SOSI Generell objektkatalog

Pakker inneholder applikasjonsskjemaer eller nye pakker, det siste er tilfellet med pakken SOSI Generell objektkatalog. Denne pakken inneholder nye pakker, hver og en av dem vil typisk inneholde applikasjonsskjema for ett fagområde i SOSI, for eksempel Jernbane.



Figur 6.Eksempel på pakker i pakken SOSI Generell objektkatalog

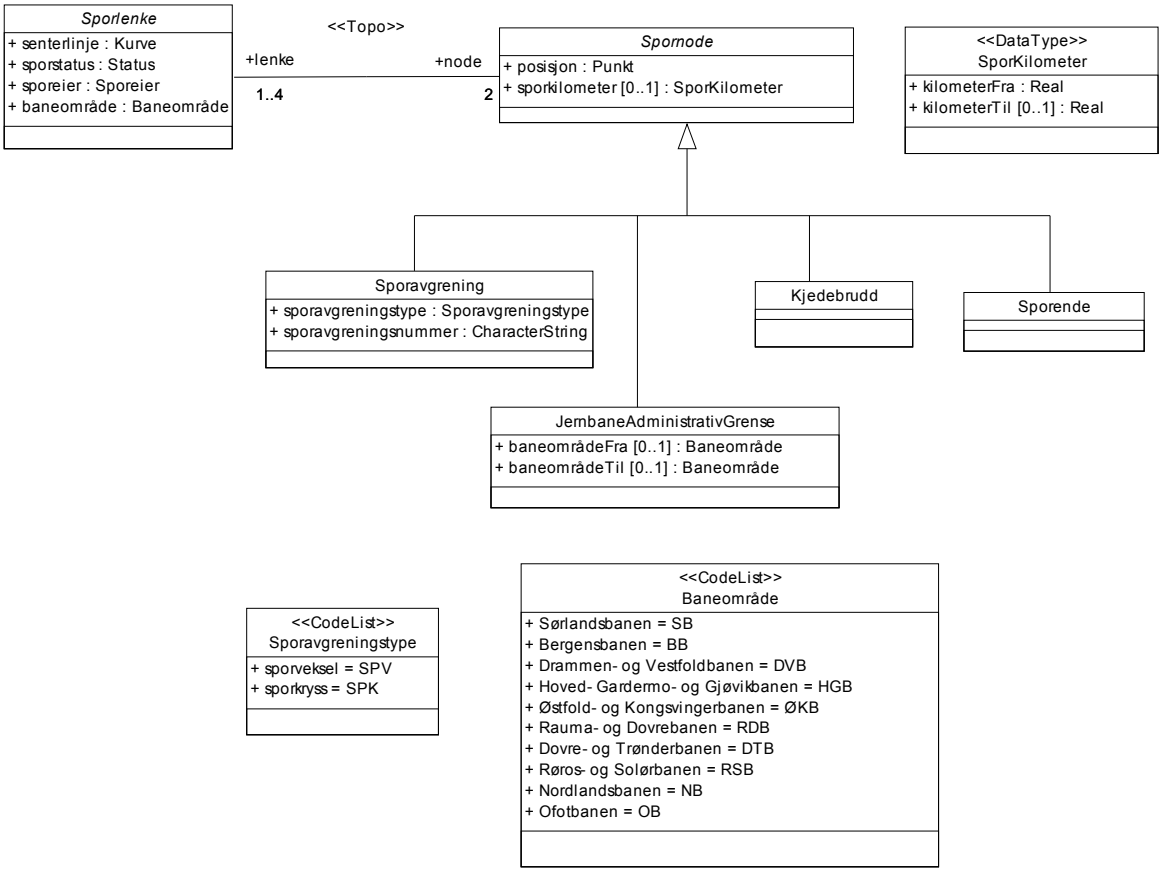
4.5.2 viser et eksempel på et utdrag av innholdet i pakken SOSI Generell objektkatalog.

4.5.3 Pakken SOSI Jernbane

Innholdet i pakken Jernbane er en versjon av applikasjonsskjemaet for jernbane i SOSI. Applikasjonsskjemaet beskriver fagområdet Jernbane ved hjelp av objekttyper med egenskaper og forhold. Fagområdet Jernbane er relativt komplekst med mange objekttyper og flere forhold objekttypen imellom. Når en skal presentere slike komplekse applikasjonsskjemaer er det ofte hensiktsmessig å dele opp modellen. På den måten kan en fremheve deler av modellen om gangen for presentasjon (ofte kalt et view/oversikt). Det er ofte lettere å forstå helheten hvis man har et klart bilde av delene. Den følgende presentasjonen av jernbanemodellen er eksempel hvordan et applikasjonsskjema kan presenteres.

4.5.3.1 Spornoder

Jernbanemodellen består i hovedsak av en node-lenke struktur der nodene er punkter i jernbanenettet som knytter sammen jernbanestrekninger (Sporlenker). 4.5.3.1 viser hovedsakelig objekttypen Spornode og de typer spornoder jernbanenettet har samt datatyper brukt av objekttypene. For å vise koblingen mellom spornoder og selve jernbanestrekningen er også objekttypen Sporlenke med i modellen.

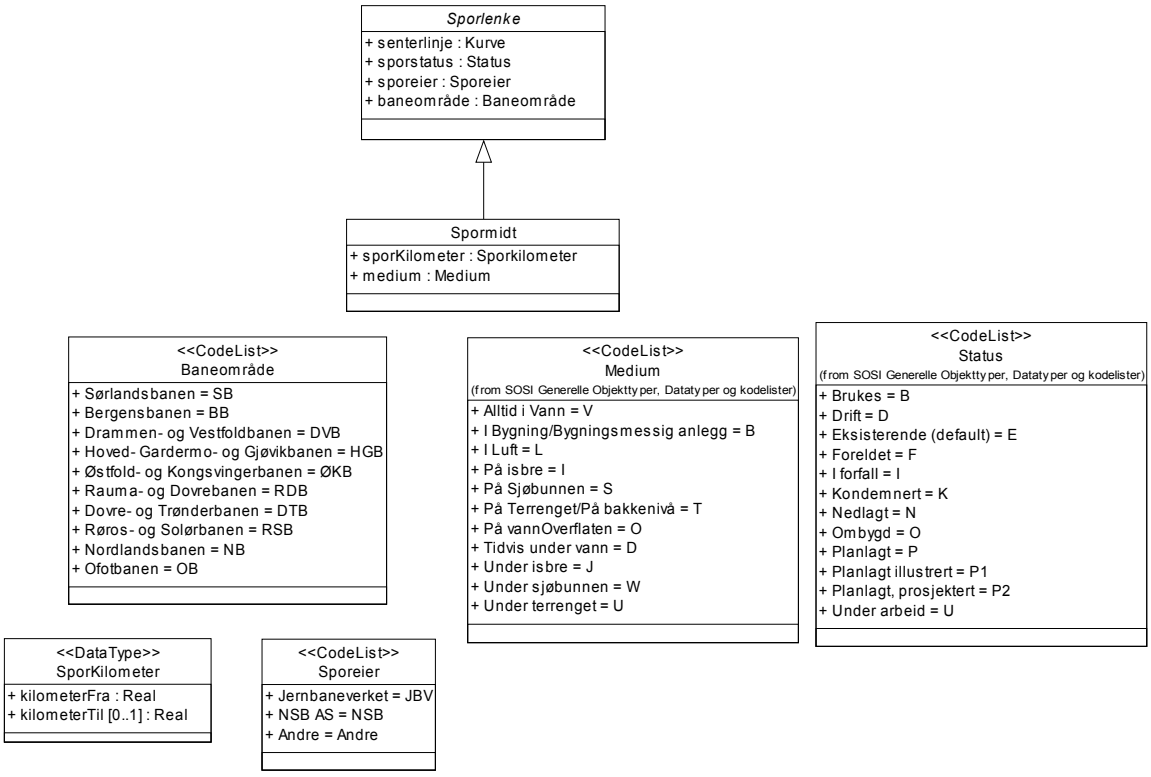


Figur 7.Eksempel på spornoder i en jernbanemodell

Objekttypen Sporlenke er her abstrakt noe som vises ved at klassenavnet er i kursiv. Abstrakte objekttyper vil aldri finnes som en forekomst i for eksempel et datasett, de vil alltid kun eksistere som en av sine definerte subtyper. I modellen ser vi at f.eks Kjedebrudd og Sporavgrening er to forskjellige typer spornoder. To kodelister er tatt med i modellen, Sporavgreningstype og Baneområde, kodelistene brukes av forskjellige objekttyper og vi er her tatt med siden de brukes av spornode objekttypene. Vi kan også se datatypen Sporkilometer som er definert i objekttypen Spornode og dermed arves av alle subtypene til Spornode.

4.5.3.2 Sporlenke

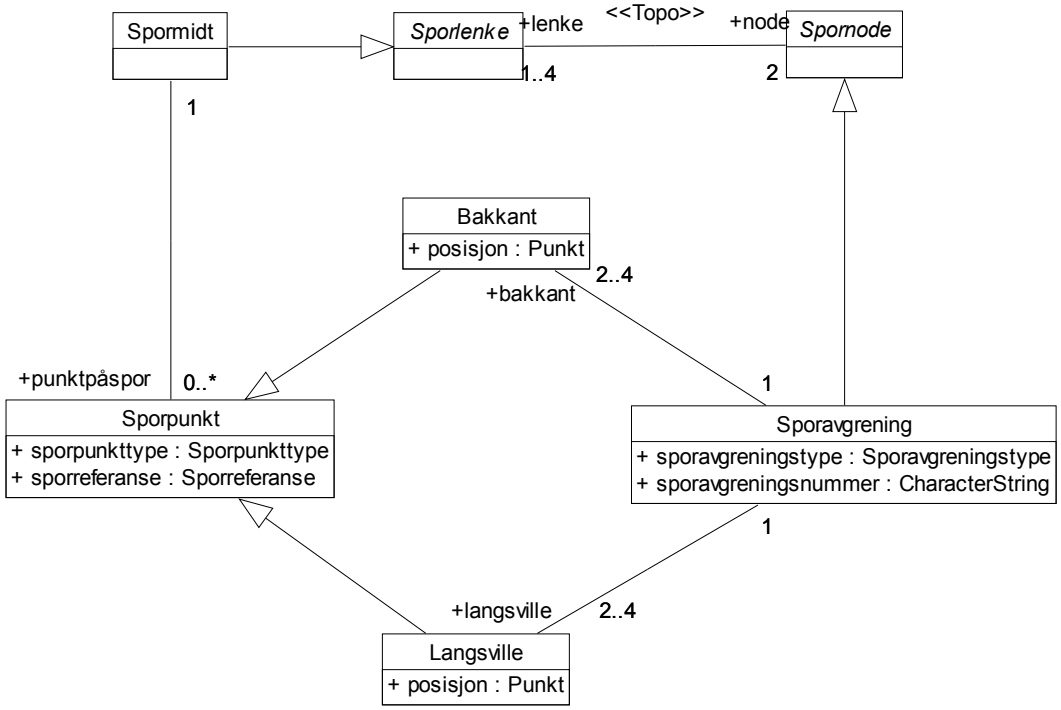
4.5.3.2 viser modellen for jernbanens sporlenker. Objekttypen Sporlenke er her en abstrakt objekttype som egentlig kun er et begrep i og med at Spormidt er den eneste subtypen. Egentlig kunne vi klart oss med objekttypen Spormidt og latt den ha alle egenskapene til Sporlenke samt koblingen til spornodene, men siden sporlenke er et begrep i node-lenke strukturen har vi modellert det slik som under. Sporlenke og Spormidt refererer til en rekke kodelister og en datatype som også vises i modellen. Kodelistene Medium og Status er generelle SOSI kodelister hentet fra en annen pakke som inneholder generelle SOSI datatyper og egenskaper, navnet på pakken de er hentet fra er listet opp i parantes under klassenavnet i modellen.



Figur 8.Eksempel på sporlenke med tilhørende kodelister i en jernbanemodell

4.5.3.3 Forhold mellom punkter på spor og spornoder

4.5.3.3 viser en annen del av modellen. Her beskrives forholdet mellom Spornode og Sporlenke igjen samtidig som den beskriver objekttyper som verken er Spornoder eller Sporlenker, men som har et forhold til disse. Objekttypene Bakkant og Langsville er punkter (subtyper av Sporpunkt) som finnes på sporlenken i forbindelse med sporavgreninger, men de deler ikke opp jernbanenettet slik de objekttypene som er spornoder gjør.



Figur 9.Eksempel: Jernbanemodell med punkter på spor og spornoder

5 Praktiske regler for modellering

5.1 Introduksjon

Hensikten med dette kapitlet er å gi forslag til løsning på en rekke praktiske problemstillinger knyttet til modellering. Hvert kapittel beskriver et emne, gir en teoretisk beskrivelse, foreslår regler samt viser hvordan dette kan løses i modellen.

Kapitlet inneholder mange eksempler (figurer) fra modeller i eksisterende standarder. Disse er imidlertid ikke nødvendigvis komplette eller ajourførte i henhold til siste versjon (eksempelvis SOSI 4.0), og må ikke betraktes som annet enn eksempler.

5.2 Hva er en objekttype

Med objekttype mener vi en avbildning av et fenomen i den virkelige verden. Stort sett er det snakk om en avbildning av et stedfestet objekt, f.eks. hus, vann, vei, etc. Objekttyper modelleres om klasser i UML. Men det er også mange andre konstruksjoner som modelleres som klasser, som for eksempel datatyper og kodelister, men disse er stereotypet med henholdsvis DataType og CodeList som gir klassen spesiell mening (se 5.10.1). Hovedregelen er at alle klasser i modellen som ikke er 'stereotypet' er objekttyper.

5.3 Regler for navning i UML

- Objekttypenavn skrives med stor forbokstav (eks. Eiendomsteig)
- Egenskapsnavn skrives med liten forbokstav (eks. grunnidentifikasjon)
- Rollenavn skrives med liten forbokstav (eks. eiendom)
- Datatypenavn skrives med stor forbokstav (eks. Grunnidentifikasjon)
- Kodelistenavn skrives med stor forbokstav (eks. Eiendomskode)

Dersom navnet er sammensatt av flere ord, benyttes STOR førstebokstav fra og med andre ord (eksempel: engelskArtsnavn). Navn skal være mest mulig folkelige begrep uten unødig bruk av store bokstaver. De norske karakterene Æ,æ.Ø,ø,Å,å er tillatt brukt. Retningslinjer for navning til bruk i SOSI Generell objektkatalog er nærmere beskrevet i dokumentet ”Regler for navning av geografisk informasjon”.

I arbeidet med modelleringen av den generelle objektkatalogen benyttes ikke ’underscore (_)’. Men dette er derimot benyttet i flere av modellene som finnes i ISO 191xx standarder og vil derfor kunne integreres i et applikasjonsskjema (informasjonsmodell). Det er altså ingen regel som forbyr bruk av underscore.

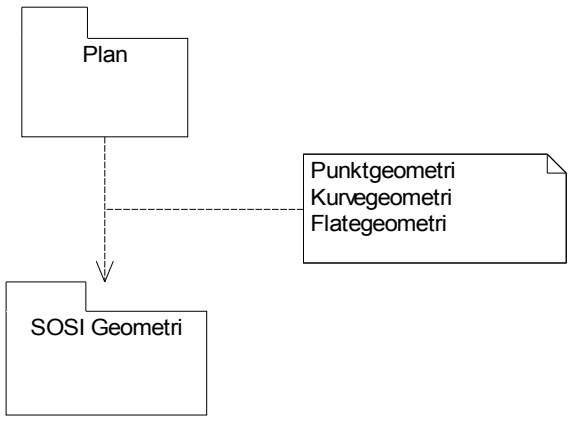
5.4 Hovedstrukturering av modeller

5.4.1 Integrering av modeller

Ved utvikling av en større informasjonsmodell er det fordelaktig å bryte denne ned i mindre komponenter. Eksempelvis er selve applikasjonsskjema en del, men et applikasjonsskjema vil gjerne i tillegg bruke andre standardiserte komponenter (modeller) som er andre deler. Eksempler på slike andre komponenter er kvalitetsmodell, geometri/topologi modell, metadata elementer etc. Eksempelvis har de fleste objekttyper egenskaper som knytter en geometri til seg, men selve geometritypen som brukes vil være definert et annet sted enn i selve fagmodellen. Vi skal se eksempler på hvordan dette gjøres.

5.4.1 viser en modell kalt Plan (SOSI fagområde Plan). Denne benytter komponenter i SOSI_Geometri profil, som inneholder geometriske primitiver som er nødvendig for å gi en geometrisk posisjon og/eller utstrekning til objekttypene i Plan. Disse geometritypene er beskrevet kun en gang, og integreres ved bruk av 'dependency' (avhengighet) i UML. (stiplet pil).

Det er også mulig å beskrive hvilke komponenter i de respektive pakkene som benyttes. Noten koblet til avhengigheten i 5.4.1 viser at pakken Plan benytter seg av Punkt, Kurve og Flate.



Figur 10.Eksempel på pakkeavhengighet

Dette betyr at disse 3 klassene er benyttet som datatype i pakken Plan, og er nærmere definert i pakken SOSI_Geometri profil.

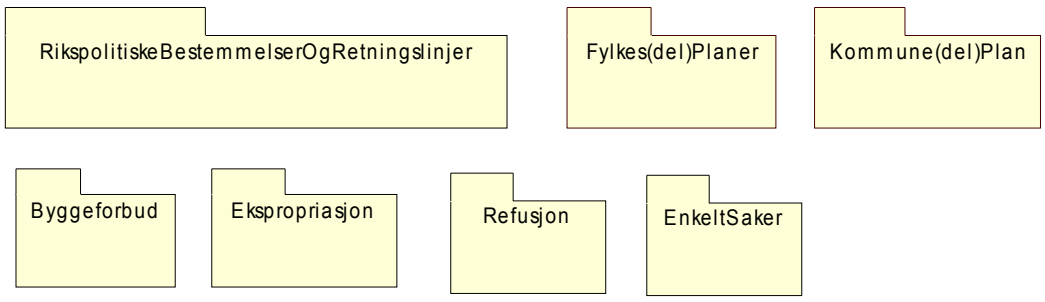
Det er også mulig å anvende klasser i ISO 19107 Spatial Schema (geometri/topologi modellen) direkte, men det er en fordel å forholde seg til en nærmere definert profil av denne, ikke minst med tanke på implementasjon.

Regler:

- Selve modellen skal modelleres i et applikasjonsskjema (informasjonsmodell).
- Dependency (Avhengighet) mekanismen i UML skal benyttes for å integrere andre skjemaer som er nødvendig for å lage en komplett modell.
- Dersom modellen også skal være grunnlag for datautveksling, må alle klasser i den komplette modellen være instansierbare. Klasser man normalt bruker, slik som objekttyper, datatyper, kodelister og enumereringer er alle instansierbare. Dog finnes det stereotyper som gir en klasse helt spesiell mening, se 5.10.1 for mer om stereotyper.
- Hvilke komponenter i de integrerte skjemaene som skal benyttes kan angis, men det er ingen krav om dette. Dette er kun ut fra lesbarheten i form av menneskelig tolkning av den grafiske modellen. Det er ikke nødvendig med tanke på maksinlesbarheten.

5.4.2 Oppdeling av modeller

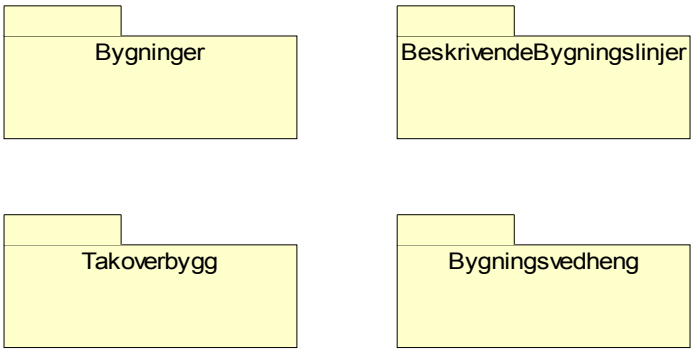
På samme måte som en kan integrere flere skjema, kan en også dele opp en applikasjonsmodell i flere skjema. Hensikten med en slik oppdeling er å gjøre de respektive delmodellene enklere å lese (for mennesker). Det har ingen betydning for maskinlesbarheten. Utgangspunktet for en slik oppdeling er å lage fornuftige enheter som er oversiktlige og enklere å forholde seg til. Dette gjelder både under selve modelleringsfasen, samt med tanke på dokumentasjon. Oppdeling av modellen gjør ofte dokumentasjonen enklere. 5.4.2 viser et eksempel fra SOSI Plan der modellen er delt opp i logiske enheter, UML pakker.



Figur 11.Eksempel på oppdeling i pakker

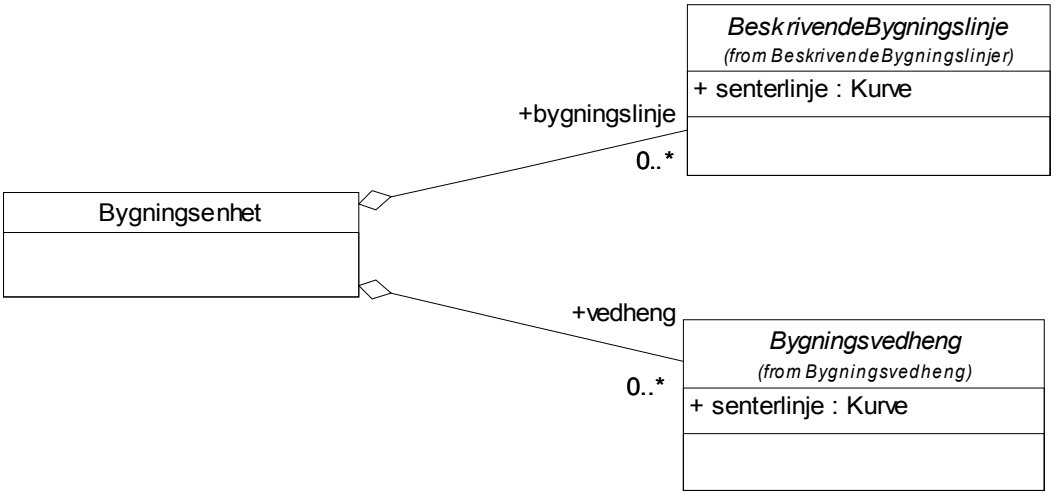
Eksempler på slike pakker er Byggeforbud, Ekspropriasjon, etc. Hver av disse inneholder underliggende klasser, dvs. klasser som er objekttyper, datatyper og kodelister. I henhold til den grafiske notasjonen er alle pakkene likeverdige, dvs. ingen av pakkene er overordnet eller underordnet andre. Utgangspunktet for en slik oppdeling er at hver pakke inneholder objekttyper, datatyper og kodelister som har et nærmere forhold til hverandre (hører sammen). Men det er fullt mulig å referere fra en objekttype i en pakke til en objekttype i en annen pakke.

5.4.2 viser et eksempel fra bygninger. Modellen for bygninger er delt inn i 4 pakker, Bygninger, BeskrivendeBygningslinjer, Takoverbygg og Bygningsvedheng.



Figur 12.Eksempler på fagområde som er delt opp fire pakker

I 5.4.2 viser vi hvordan objekttypen BygningsEnhet i Bygninger forholder seg til objekttypen BeskrivendeBygningslinje i pakken BeskrivendeBygningslinjer samt objekttypen Bygningsvedheng i pakken Bygningsvedheng. Legg merke til at objekttypene BeskrivendeBygningslinje og Bygningsvedheng er abstrakte (objekttypenavn i kursiv). For mer om abstrakte objekttyper se kapittel 5.5.3.3. Egenskapene til objekttypen Bygningsenhet vises heller ikke.



Figur 13.Objekttypen Bygningsenhet med to aggregerte objekttyper fra andre pakker

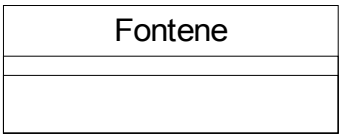
Regler:

- Det er vanskelig å gi klare regler for inndeling i pakker. Dette må gjøres ut fra fagmiljøenes vurdering av hva som oppfattes som fornuftig, og ut fra behovet for dokumentasjon.

5.5 De enkelte komponenter i et applikasjonsskjema (informasjonsmodell)

5.5.1 Objekttyper

Objekttypene er selve grunnstenen i applikasjonsskjema. Disse modelleres som klasser.



Figur 14.Eksempel på en enkel objekttype definert som klasse

Figuren viser objekttypen Fontene modellert som klasse (egenskapene til objekttypen vises ikke i dette eksempelet).

Som nevnt tidligere representerer klasser i UML også gruppeegenskaper som datatyper samt åpne og lukkede kodelister. Disse klassene skal alltid stereotypes spesielt (se kapittel 5.10.1 om stereotyping) slik at de skiller seg fra objekttyper. Alle klasser som ikke har en stereotype er å anse som objekttyper.

5.5.2 Egenskaper

En egenskap består av et navn, egenskapsnavnet, og en type (datatype) som beskriver verdidomenet instanser av egenskapen kan ha. En objekttype kan ha ingen eller flere egenskaper, det er ingen begrensning på antallet. 5.5.2 viser objekttypen Fiskebruksområde der objekttypen for eksempel har en egenskap med navn **område** med datatype **Flate**, datatypen beskriver verdidomenet egenskapen kan anta og er dette tilfellet en geometri med utstrekning som flate.

Figur 15.Eksempel på en klasse med egenskaper

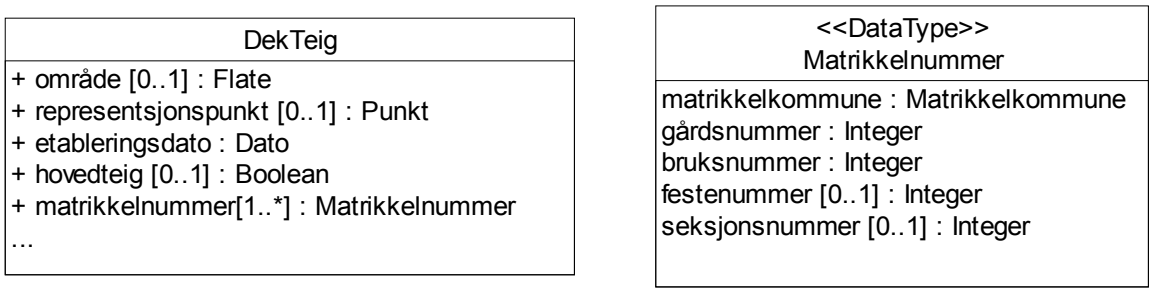
Videre har objekttypen egenskapene representasjonspunkt, temakode, fiskeribrukstype, periode, fiskeriressursområdeart, navn, informasjon og fiskeriredskap. Egenskapene område og representasjonspunkt beskriver geometrien til objekttypen og er egenskaper på lik linje med de andre egenskapene. Datatypene og kodelistene som brukes av objekttypen, for eksempel kodelisten Fiskeribrukstype og datatypen Periode, vises ikke i sin helhet i dette eksempelet. Disse er beskrevet i samme eller eventuelt andre delmodeller, et eksempel på hvordan dette ser ut finnes i neste avsnitt.

5.5.2.1 Modellering av SOSI basisegenskaper og gruppeegenskaper

En objekttype i SOSI kan ha egenskaper i form av basisegenskaper og gruppeegenskaper der basisegenskaper er enkle egenskaper mens gruppeegenskaper er sammensatt av flere basisegenskaper. Basisegenskap og gruppegenskap er begreper som er knyttet til SOSI-formatet, vi bruker de her kun for å eksemplifisere hvordan de kan modelleres i UML. Eksempel på en basisegenskap er egenskapen informasjon i eksempelet i 5.5.2, denne egenskapen består av et egenskapsnavn, informasjon, og en type, CharacterString. Matrikkelnummer (tidligere grunnidentifikasjon,GID) er et eksempel på en SOSI-gruppeegenskap i som kodes på følgende måte i SOSI-formatet:

```
.MATRIKKELNUMMER *
    ..MATRIKKELKOMMUNE H4
        ...GNR H4
        ...BNR H4
        ...FNR H4
        ...SNR H4
```

Gruppeegenskaper i henhold til SOSI-formatet modelleres som egne klasser (stereotypet DataType), med basisegenskapene som attributter. Matrikkelnummer er en egenskap til eksempelvis objekttypenDekTeig som vises i 5.5.2.1. Dekteig har en egenskap med navn matrikkelnummer og verdidomene (type) Matrikkelnummer. Verdidomenet refererer til datatypen Matrikkelnummer som er en egen klasse stereotypet DataType med navnet Matrikkelnummer.



Figur 16.Eksempel på modellering av gruppeegenskap

Datatypen Matrikkelnummer har egenskapene matrikkelkommune, gårdsnummer, bruksnummer, festenummer og seksjonsnummer. Alternativt kan disse egenskapene legges direkte på objekttypen Eiendomsteig, men en mister da anledningen til å spesifisere multiplisitet overfor gruppen av egenskaper samtidig som gjenbruk av datatypen (begrepet) ikke være mulig. UML gir mulighet til en mer detaljert spesifikasjon enn vi har gjort i SOSI i dag.

5.5.2.2 Multiplisitet for egenskapene

Alle egenskaper har en multiplisitet. Multiplisiteten angir hvor mange ganger egenskapen kan forekomme for en instans av objekttypen. Det kan benyttes en verdi, ett sett verdier skilt med komma, eller et intervall skilt med to prikker imellom og en Asterix (*) betyr mange.

I eksempelet over med eiendomsteigen ser vi at det etter egenskapen **hovedteig** står [0..1] som betyr at egenskapen **hovedteig** kan forekomme minimum 0 ganger og maksimum en gang. Egenskapen **etableringsdato** har i dette eksempelet ingen angitt multiplisitet og forekommer derfor per definisjon en gang (må være med). Videre ser vi at **matrikkelnummer** kan forkomme null eller flere ganger.

Eksempler:

+ **gårdsnummer : Integer** // egenskapen er obligatorisk (ingen angitt multiplisitet)
+ **festenummer [0..1] : Integer** //multiplisitet 0..1 betyr at denne er opsjonell (trenger ikke forekomme), og kan forekomme maksimalt 1 gang.

Multiplisiten angir altså hvor mange instanser som er lovlige. I SOSI er vi vant til å snakke om påkrevde og opsjonelle egenskaper, egenskaper som er påkrevde i SOSI vil ha en multiplisitet større enn null (normalt 1), mens opsjonelle vil ha multiplisitet fra null og oppover, for eksempel [0..1].

5.5.2.3 Syntaks for egenskaper

Egenskaper modelleres som attributter til klasser, og har følgende notasjon:

[/] [visibility] name [multiplicity] [:type] [= initial value] [{property-string}]

Forklaring:

[/]	Angir at egenskapen er avledet. En egenskap som kan bli beregnet fra et annet modellelement, men som vises i modellen av informasjonshensyn betegnes som avledet.
[visibility]	Spesifiserer tilgjengelighet. Det er tre muligheter (public, protected og private). For modellering av geografiske objekter for en objektkatalog er i utgangspunktet alle egenskaper tilgjengelige, dvs. public. Public angis med en + før egenskapsnavnet.
Name	Navnet på egenskapen.
[multiplicity]	Antall forekomster av en egenskap, kan være alt fra opsjonell (valgfri) til mange. Se kapittel om multiplisitet. Dersom ingen verdi er gitt, er dette identisk med 1, dvs. en forekomst av egenskapen pr. klasse.
[:type]	Datatypen til objektet. Kan være 'basic' (f.eks. Integer, characterString), en egendefinert datatype modellert som en egen klasse stereotypet DataType, for eksempel GID eller en kodeliste som f.eks Kommunenummer.
[=initial value]	Default verdi.
{property string}	Det er 3 definerte egenskaper som kan knyttes til egenskaper (changeable, addonly og frozen).

For modellering av geografiske objekttyper i et applikasjonsskjema er det ikke behov for å benytte den fulle syntaksen for egenskaper. Det er tilstrekkelig å angi 'visibility' lik public (+), selve navnet på egenskapen (name), eventuelt multiplisitet (hvor mange instanser som kan forekomme), samt datatype.

5.5.3 Forhold

UML har fire ulike måter å angi forhold (relationships):

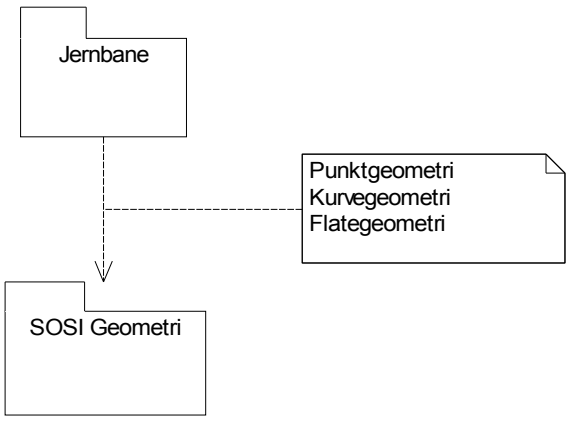
- Avhengighet (dependency)
- Assosiasjon (association)
- Generalisering (generalization)
- Realisering (realization)

De tre første typer forhold er relevante for generell informasjonsmodellering slik vi gjør i forbindelse med objektkatalogen. Forholdet som kalles realisering er noe mer avansert og tas ikke med i den generelle delen, en beskrivelse finnes i Annex 8.

5.5.3.1 Avhengighet (dependency)

Avhengighet er et semantisk forhold mellom to elementer, hvor en endring i det ene elementet (det uavhengige), påvirker semantikken i det andre elementet (det avhengige). En avhengighet angis grafisk som en rettet stiplet linje og kan gis et navn om man eksplisitt ønsker det.

I kapittel 5.4så vi eksempler på hvordan en kan bruke avhengigheter til å strukture modellene våre, her gis et lignende eksempel. I 5.5.3.1 er innholdet i pakken Jernbane avhengig av innholdet i pakken SOSI Geometri. Dersom f.eks. spesifikasjonen av Kurve i pakken SOSI Geometri endres, vil dette medføre endring i modellen for Jernbane.



Figur 17.Eksempel på angivelse av avhengighet

Merknad
Må ikke forveksle avhengighet med 'anker' notasjonen for merknader (noter). I den grafiske modellen ser disse like ut dersom en angir en avhengighet uten retning.

Regler:
Benyttes for å angi at et element er avhengig av et annet element, hovedsakelig for å angi avhengighet mellom pakker, jamfør kapittel 5.4.1.

5.5.3.2 Assosiasjoner

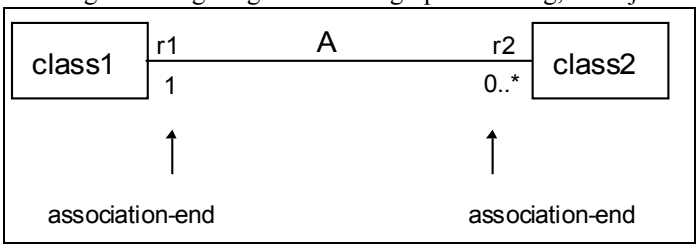
En assosiasjon i UML er det semantiske forhold mellom to eller flere modellelementer, og brukes hovedsakelig for å ivareta sammenhenger mellom objekttyper.

Assosiasjoner tillegges rollenavn for å forklare en klasses rolle i forhold til en annen klasse, og skal angis som substantiv. ISO 19103 sier at alle assosiasjoner skal ha multiplisitet definert for begge endene til assosiasjonen og minst et rollenavn skal være spesifisert

En assosiasjon er beskrevet som følger (CSL):
*An association is used to describe a relationship between two or more classes. In addition to an **ordinary association**, UML defines two special types of associations called **aggregation** and **composition**. The three types have different semantics. An ordinary association shall be used to represent a general relationship between two classes. The aggregation and composition associations shall be used to create part-whole relationships between two classes.*

5.5.3.2.1Generell assosiasjon (ordinary association)

Objekttyper kan ha assosiasjoner til andre objekttyper. De angis med en strek mellom de to impliserte objekttypene. En assosiasjon kan ha et navn (verb) som skal stå midt på forbindelsesstreken, men vi regner assosiasjonens "rollenavn" som viktigere. Objektets rolle mot andre objekttypen angir hva det betyr for det andre objektet gjennom denne assosiasjonen. En pil i den ene enden angir at navigering kun er mulig i pilas retning, i seksjon 5.5.3.2.5 er navigering beskrevet mer presist.



Figur 18.Assosiasjon

Dette leses som:
Class1 ser class2 gjennom rollen r2.
Class 1 i assosiasjonen A ser class2 gjennom rollen r2.
(Invers for r1)

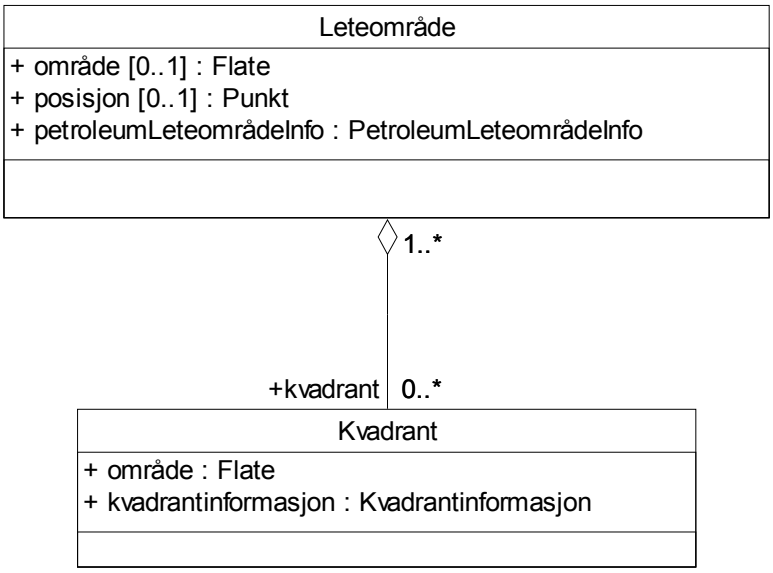
En generell assosiasjon benyttes mellom likestilte (organisatoriske) klasser.

5.5.3.2.2Aggregering (svak aggregering)

En åpen "diamant" angir en svak aggregering, hvor objekttypen på "diamantsiden" består av, og eksisterer i kraft av sine "medlemmer". Objekter kan således være medlemmer eller bestanddeler av flere slike helheter/åpne aggregeringer samtidig.

Aggregering er en spesiell form for assosiasjon og benyttes for å angi et strukturelt forhold mellom et 'hele' og de respektive 'deler'. Aggregering er i utgangspunktet et enkelt konsept. Det eneste som ligger i denne spesialiseringen fremfor en ordinær assosiasjon, er å skille mellom 'hovedelementer' og deres 'delelementer', dvs. hvem av klassene som organisasjonsmessig er overordnet. Bruk av aggregering fremfor en generell assosiasjon gir ingen endring i navigering i modellen.

Eksemplet i 5.5.3.2.2 viser en svak aggregering mellom objekttypene Leteområde og Kvadrant.



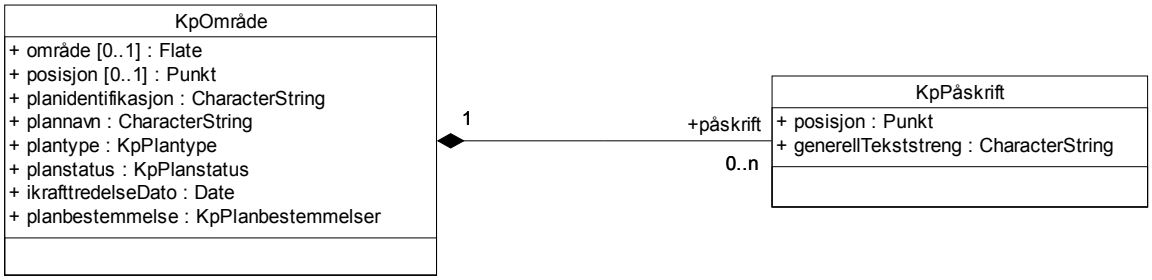
Figur 19.Eksempel på en svak aggregering mellom Leteområde og Kvadrant

Meningen med assosisasjonen og den svake aggregeringen i eksemplet over er at en kvadrant er en del av et leteområde.

5.5.3.2.3Komposisjon (sterk aggregering)

En fylt "diamant" angir at objekter av objekttypen på "diamantsiden" har, eller eier objekter av objekttypen på "strek-siden" som komponenter. De eide objektenes oppretting og eksistens følger eierobjektets eksistens, en assosiasjon som utrykker dette forholdet kalles en komposisjon eller sterk aggregering.

5.5.3.2.3 viser et eksempel på en sterk aggregering mellom KpOmråde og KpPåskrift. KpPåskrift kan bare tilhøre KpOmråde. Dersom KpOmråde slettes, slettes også KpPåskrift.



Figur 20.Eksempel på en sterk aggregering /komposisjon

5.5.3.2.4Multiplisitet for assosiasjoner

Som for egenskaper angis også multiplisitet for assosiasjoner, se 5.5.2.2. For assosiasjoner beskrives multiplisitet ved enden av assosiasjonene og aggregeringene, og angir hvor mange instanser som lovlig kan knyttes til et objekt via assosiasjonen. 5.5.3.2.4 viser syntaksen UML bruker for å uttrykke kardinalitet/multiplisitet for assosiasjoner:

Forklaring	Notasjon
Obligatorisk, kun en instans	1 Klasse1
Opsjonell, kan ha mange instanser	0..* Klasse1
Opsjonell, men ikke mer enn 1 instans.	0..1 Klasse1
Obligatorisk, minst en instans	1..* Klasse1
Obligatorisk, et gitt antall instanser	1,3,7 Klasse1

Figur 21.Angivelse av ulike typer multiplisitet

Vi bruker den samme notasjonen som for egenskaper.

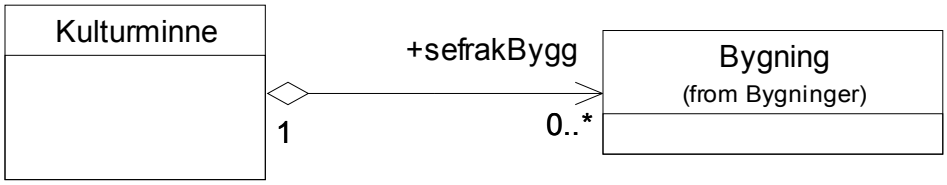
5.5.3.2.5Retning på assosiasjoner

Man kan angi retning på assosiasjonene, retningen på assosiasjonen beskriver det vi kaller navigering i UML. Navigering angir hvilken vei assosiasjonen er rettet, dvs. man angir hvilken klasse som kjenner til forholdet (assosiasjonen). Eksempelet i 5.5.3.2.5 viser at assosiasjonen mellom Bruker og Passord er rettet mot Passord.

Figur 22.Eksemper på assosiasjon med retning

Man sier at Bruker vet om Passord via rollen løsen (rollen løsen til Passord sier at passordet er ”løsenet” eller ordet som må oppgis for at brukeren skal få tilgang til systemet), men Passord kan ikke ”navigere” til Bruker. I denne sammenheng er assosiasjonen rettet. Grunnen er at et system gitt en bruker, må kunne finne brukerens passord, men gitt et passord ønsker man ikke å kunne finne brukeren som er assosiert med passordet.

Retning på assosiasjoner er nyttig i flere tilfeller. Spesielt hvis vi modellerer med objekttyper i ulike fagområder der et fagområde kan knyttes til det andre mens det omvendt ikke blir riktig. Se for eksempel



Figur 23.Enda et eksempel på assosiasjon med retning

Kulturminne i 5.5.3.2.5, her ønsker man å si at et Kulturminne kan bestå av eller knyttes til bygninger (sefrakBygg), men Kulturminne som objekttype har ingen tilknytning til selve modellen for bygninger. Av den grunn er assosiasjonen rettet i modellen. Vi bruker vanligvis ikke retning på assosiasjonene såfremt det ikke er noe vi spesifikt ønsker å angi, jamfør eksemplene i 5.5.3.2.5 og 5.5.3.2.5.

5.5.3.2.6Regler knyttet til assosiasjoner

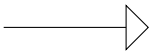
- Rollenavn angis i substantivs form og skal angis for alle assosiasjoner. Minst ett rollenavn skal angis for hver assosiasjon (ISO 19103 CSL). Denne rollen bør ligge på 'mange' siden av assosiasjonen (0..*, 1..*).

- Rollenavn er unike innenfor den klassen disse står til. Hvis en har behov for like rollenavn er det mest sannsynlig noe feil med selve modellen. En rolle kan oppfattes som en egenskap til en klasse. I eksemplene over er f.eks. rollen r2 å oppfatte som en egenskap til klassen class1.

- Retning på assosiasjonene angis kun såfremt det er nødvendig

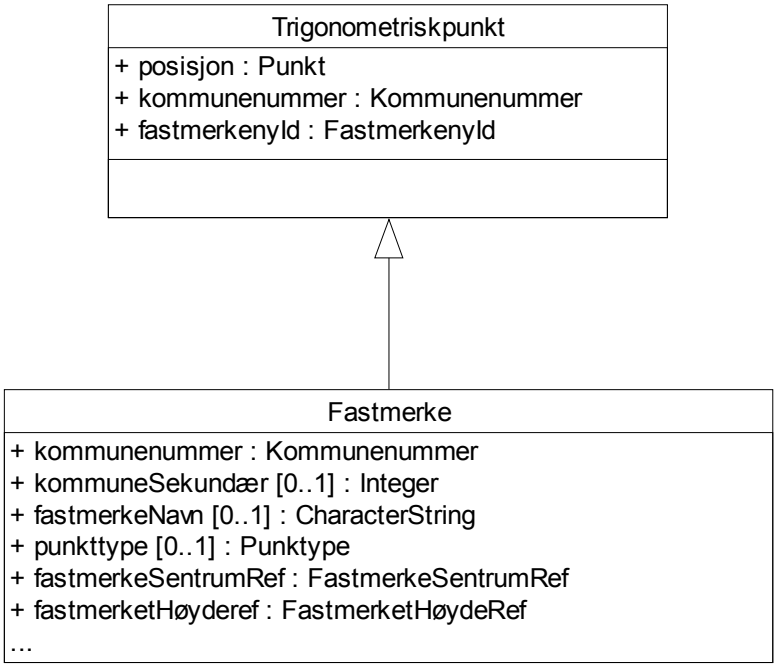
5.5.3.3 Generalisering/spesialisering

Generalisering og spesialisering angir et forhold hvor alle subtyper (barn) arver egenskaper, operasjoner og assosiasjoner av sine supertyper (foreldre).

Generalisering angis med heltrukken strek med åpen trekant i enden 

Generalisering/spesialisering kan brukes i mange tilfeller, her beskrives to eksempler der det kan være hensiktsmessig å bruke dette forholdet.

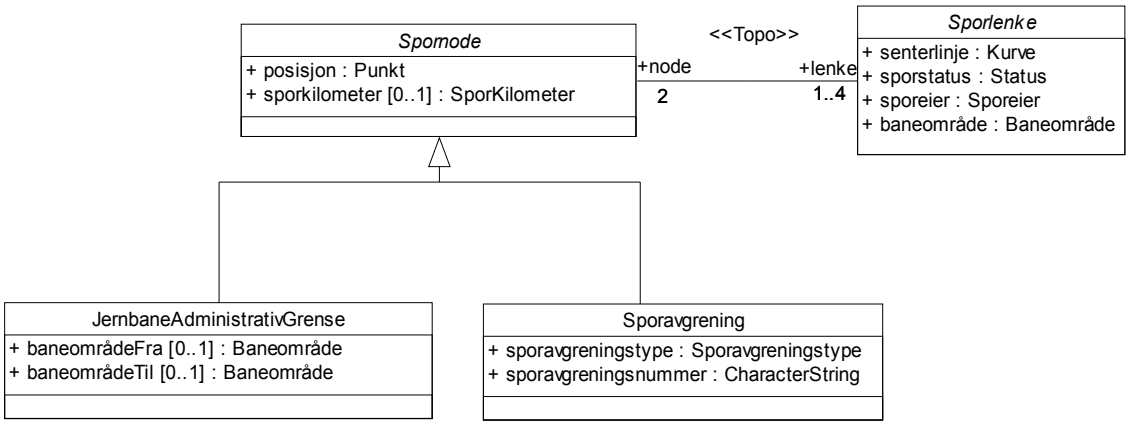
1. For å kunne beskrive ting på ulike nivåer.



Figur 24.Eksempel på bruk av spesialisering for å angi ulik detaljeringsgrad.

5.5.3.3 beskriver fastmerke som en spesialisering av et trigonometrisk punkt. For enkel bruk er det tilstrekkelig å benytte den generaliserte objekttypen Trigonometrisk punkt, mens det for mer spesialisert bruk kan være nødvendig å benytte objekttypen Fastmerke, som arver egenskapene til trigonometrisk punkt og i tillegg har egne egenskaper.

2. Modelleringsteknisk for å forenkle modeller, hvor spesialiseringen arver alle egenskaper og assosiasjoner fra supertypen.



Figur 25.Eksempel på modelleringsteknisk bruk av spesialisering

5.5.3.3 gir et eksempel på en spesialisering av *Spornode* i form av to subtyper JernbaneAdministrativGrense og Sporavgrening. Subtypene arver egenskapene fra den generelle objekttypen *Spornode* (posisjon og sporKilometer) samt assosiasjonen til *Sporlenke*. *Spornode* er abstract og forekomster av objekttypen vil derfor alltid være enten Sporavgrening eller JernbaneAdministrativGrense. Her utnyttes generalisering/spesialiseringsforholdet til å skape et begrep (Spornode) med generelle egenskaper som gjelder for begrepet og alle spesialiseringer av dette. Sporavgrening og JernbaneAdministrativGrense har egne egenskaper som gjør de til spesialtilfeller av av en spornode.

Regler for generalisering/spesialisering:

- Brukes i utgangspunkt der det er behov for å angi en spesialisering, men hvor hver av de spesialiserte klassene har egne egenskaper eller spesielle krav til multiplisitet på egenskaper eller assosiasjoner.
- Abstrakte supertyper (dvs. objekttyper som ikke instansieres) skal angis med klassenavn i kursiv. (Jfr. UML-syntaks beskrivelsen).
- Det er viktig å tenke generalisering/spesialisering i en objektkatalog, da denne skal være utgangspunkt for flere produkter av ulik detaljeringsgrad.

5.5.4 Datatyper

En datatype beskriver et lovlig verdidomene for en egenskap. Et eksempel er egenskapen gårdsnummer som har verdidomene heltall, dvs. datatypen til egenskapen gårdsnummer må være heltall. I UML bruker vi datatypen Integer for å beskrive heltall.

Ved modellering av applikasjonsskjema kan datatypen til en egenskap være en basal (basic) datatype, en brukerdefinert datatype eller en kodeliste. Siden vi modellerer objekttyper som har en stedfesting kan datatypen til en egenskap også være en geometritype definert i SOSI. Se kapittel 5.6om modellering av geometri for regler angående geometriegenskaper og lovlige geometrityper.

5.5.4.1 Basale datatyper (ISO/TS 19103 Conceptual Schema Language)

Basale datatyper er enkle fundamentale datatyper som ikke er sammensatt. Basale datatyper er implementasjonsavhengige og ved implementasjon av en modell må datatypene 'mappes' over til lovlige datatyper innenfor den aktuelle plattformen. Eksempelvis vil en implementasjonsuavhengig CharacterString (tekst) implementeres som en VARCHAR2 i en Oracle database.

Regler:
Alle datatyper beskrevet i ISO 19103 CSL kan benyttes. Imidlertid anbefales følgende datatyper:

- Integer Et fortegnsbestemt heltall, lengden av en Integer er avhengig av innkapsling og bruk.
  Eksempel : 29, -34567
  A signed integer number, the length of an integer is encapsulation and usage dependent.
- Real Et fortegnsbestemt reelt tall bestående av en mantisse og en eksponent. Lengden av en Real er avhengig av innkapsling og bruk.
  Eksempel: 23.501, -1.234E-4, -23.0
  A signed real (floating point) number consisting of a mantissa and an exponent, the length of a real is encapsulation and usage dependent.
- CharacterString En CharacterString er en vilkårlig lang sekvens av tegn inkludert aksenter og spesielle tegn fra et av de angitte tegnsettene spesifisert i ISO/IEC 10646-1, ISO/IEC 10646-2.

Eksempel: 'SOSI'
A CharacterString is an arbitrary-length sequence of characters including accents and special characters from repertoire of one of the adopted character sets ISO/IEC 10646-1, ISO/IEC 10646-2.

- Date En Date gir verdier for år, måned og dag. Date kodes som en tekststreng som skal følge datoformatering spesifisert i ISO 8601Eksempel : 1998-12-21 eller 19981221.
  A date gives values for year, month and day. Character encoding of a date is a string which shall follow the format for date specified by ISO 8601.
- Time En Time angis med time, minutt og sekund. Time kodes som en tekststreng som skal følge formatering spesifisert i ISO 8601. Tidssoner i henhold til UTC er opsjonelt.
  Eksempel; 18:30:59 or 18:30:59+01:00.
  A time is given by an hour, minute and second. Character encoding of a time is a string that follows the ISO 8601 format. Time zone according to UTC is optional.
- DateTime En DateTime er en kombinasjon av typene Date og Time. Kodes i henhold til ISO 8601.
  Eksempel : 1998-12-21 18:30:59.
  A DateTime is a combination of a date and a time type. Character encoding of a DateTime shall follow ISO 8601.
- Boolean Boolean angir verdien sann eller usann.
  A value specifying TRUE or FALSE.

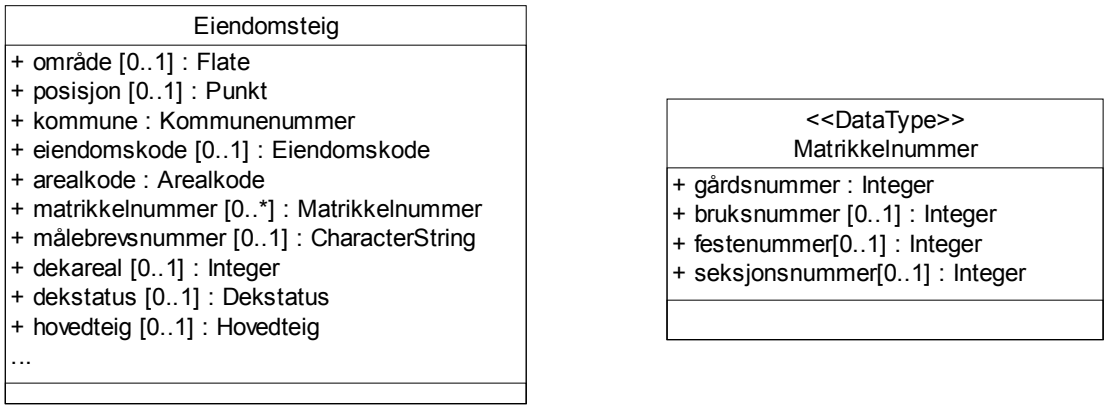
5.5.4.2 Brukerdefinerte datatyper

Dette er datatyper som modelleres av de som er ansvarlige for de ulike domenene (fagområdene). Eksempel på brukerdefinerte datatyper er SOSI gruppeegenskaper. Gruppeegenskaper er egenskaper som består av flere basisegenskaper og modelleres som datatyper i UML.

Brukerdefinerte datatyper benyttes for å:

- Angi multiplisitet til en gruppe egenskaper
- Visualisere avgrensning
- Gjøre det lettere å bruke de samme egenskapene i ulike sammenhenger
- Modellere ulike multiplisitet /avhengighet innenfor en rekke egenskaper

Eksempel:



Figur 26.Eksempel på brukerdefinerte datatyper

5.5.4.2 viser matrikkelnummer modellert som en av egenskapene til Eiendomsteig, med multiplisitet [0..*]. Denne egenskapen har datatype Matrikkelnummer. Selve datatypen Matrikkelnummer er definert som egen klasse til høyre for objekttypen med fire egenskaper: gårdsnummer, bruksnummer, festenummer og seksjonsnummer. Alle brukerdefinerte datatyper defineres i klasser som er stereotypet DataType.

Forholdet mellom basisegenskaper og gruppeegenskaper i SOSI-formatet, og hvordan disse realiseres i form av egenskaper og datatyper i UML, kan være forvirrende.

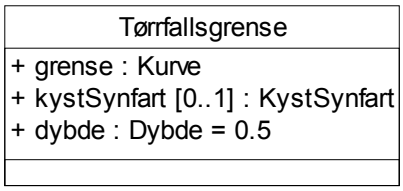
5.5.4.2.1Lengde på datatyper

I UML kan en ikke spesifisere verdidomenet for en egenskap like presist som i SOSI-formatet, f.eks. at en tekststreng kun kan være 16 karakterer lang. Dette ligger i dokumentasjonen utenfor modellen, eksempelvis SOSI_db eller i dokumentasjonsfeltet i Rational Rose.

Det har i flere diskusjoner vært presisert at eksempelvis lengde på tekststrenger, definisjoner i form av tekst og figurer, etc., vil ligge i et register (registry). Det nærmeste vi kommer er SOSI_db. Denne skal utvikles i henhold til ISO 19110 Methodology for Feature Cataloguing, og vil kunne inneholde tilleggsinformasjon til UML-modellen.

5.5.4.2.2Initialverdier

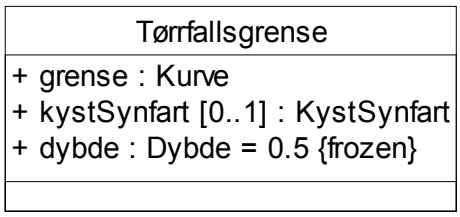
Med initialverdier i UML menes 'start'-verdier, dvs. at startverdien gjelder dersom den ikke er blitt endret.
Objekttypen i 5.5.4.2.2 er Tørrfallsgrense med en angitt dybde som initialverdi. I dette tilfellet kan dybden endres til en annen verdi om det er ønskelig.



Figur 27.Dybde som initialverdi (kan endres)

Vi kan imidlertid overstyre dette gjennom å bruke 'property' lik {frozen} for initial verdien, da sier vi at initialverdien er fryst og kan ikke endres På denne måten kan vi ivareta at en tørrfallsgrense alltid vil ha en dybde på 0,5 meter

Eksempel:

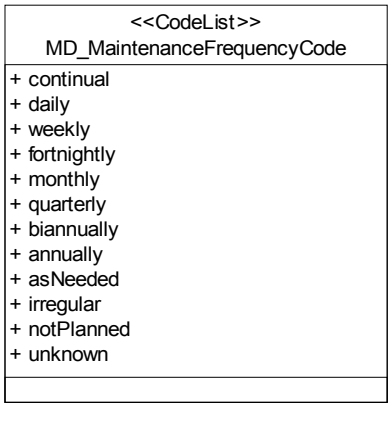


Figur 28.Tørrfallsgrense med fryst verdi for dybdeegenskapen

viser objekttypen Tørrfallsgrensev med egenskapen dybde av (data) typen Dybde. Dybde er gitt initialverdi = 0.5 og property lik frozen, dvs. at verdidomenet til egenskapen dybde for Tørrfallsgrense er forhåndsatt til å være ’0.5’.

5.5.4.3 Kodelister

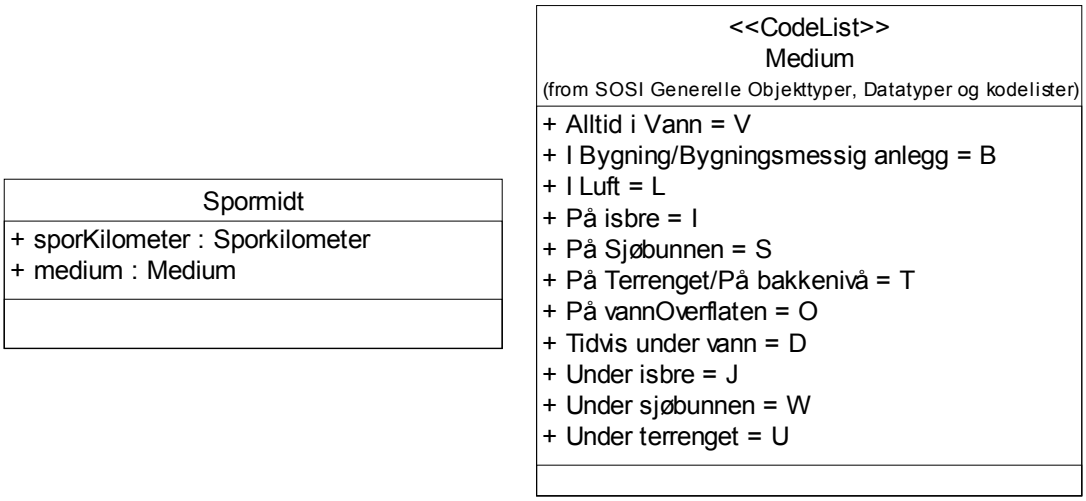
Kodelister brukes til å beskrive en åpen ’enumeration’ (liste av verdier), åpen vil si at verdier kan legges til listen. Kodelister i SOSI består av en liste av par (verdi/kode), i UML representeres disse kodelistene som en klasse steretypet CodeList. Listen av par (verdi,kode) legges inn i kodeliste klassen der hver verdi blir et egenskapsnavn og koden dens initialverdi. I de tilfeller hvor bare egenskapsnavnet er vist, er verdien og koden identiske. Eksempelet under viser en kodeliste fra ISO 19115 Metadata der det siste er tilfelle.



!

Figur 29.Eksempel på kodeliste fra ISO 19115 Metadata

Koden i en kodeliste er altså ikke obligatorisk i UML-modellen, men vi tar de gjerne med når vi modellerer kodelister i SOSI siden de fleste her er definert med både verdi og kode. Nye kodelister bør beskrives som kodelisten i 5.5.4.3 med kun verdi. 5.5.4.3 viser objekttypen Spormidt som har egenskapen medium der verdidomenet er kodelisten Medium. Kodelisten Medium har både verdi og kode slik de fleste kodelistene i SOSI er definert.



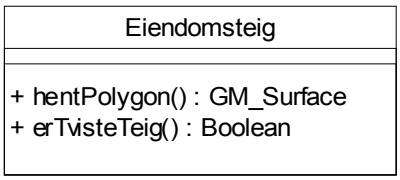
Figur 30.Eksempel på representasjon av kodeliste

Merknad: Verdier i kodelistene følger navnereglene, men med visse unntak. Egennavn endres ikke, disse må få lov til å begynne med store bokstaver, og uten sammentrekning. Eksempelvis skal ikke Øvre Eiker endres til øvreEiker.

5.5.5 Operasjoner

Operasjoner er funksjoner som kan utføres på alle instanser av den klassen hvor tjenesten er spesifisert. Et eksempel på en slik operasjon er +hentPolygon() (se 5.5.5), som kan utføres for alle objekter som er uttrykt som en flate. Dette er jo en standard operasjon for alle GIS systemer, og det har ikke vært nødvendig å spesifisere slike enkle operasjoner. Et annet eksempel kan være hvorvidt det er mulig for en lastebil å kjøre over en bro, hvor dette kan utføres som en operasjon basert på akseltrykk og dato.

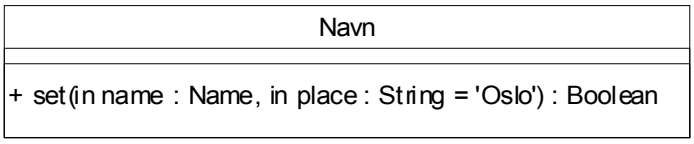
Eksemplet under viser to operasjoner knyttet til objekttypen Eiendomsteig, den første (hentPolygon) returnerer Flate i henhold til geometrimodellen. Den andre operasjonen sier om det er en tvisteteig eller ikke, og returnerer True eller False.



Figur 31.Eksempel på angivelse av operasjoner for en klasse (her objekttypen Eiendomsteig)

Egenskapene til Eiendomsteig vises ikke i eksempelet over siden eksemplet fokuserer på operasjonene.

Et annet eksempel kan være en operasjon som sjekker om et navn ligger innenfor en nærmere angitt entitet. Under vurderes om et angitt navn ligger i Oslo ved hent av returverdien True/False.



Figur 32.Eksempel på angivelse av en operasjon med 'signatur'

Regler:
Vi har lite erfaring knyttet til modellering av operasjoner, og vi bør foreløpig være forsiktige med å 'krydre' modellene med ulike operasjoner. Dette er selvsagt greit når en skal modellere et system for å håndtere datamodellen, slik som i matrikkelsammenheng. I andre sammenhenger skal data i henhold til modellene kunne overføres mellom ulike systemer, og operasjoner på data er i stor grad avhengig av funksjonaliteten til de systemene som dataene skal behandles i.

5.5.5.1 Syntaks

[visibility] name [(parameter-list)] [:return-type] [{property-string}]
[(parameter-list element)]::=
[direction] name : type [=default-value]
[direction]::= in | out |inout

Med en operasjons signatur forstås navnet på operasjonen sammen med de nødvendige parameterene i 'parameter-list', inkludert 'return-type'.

Forklaring:

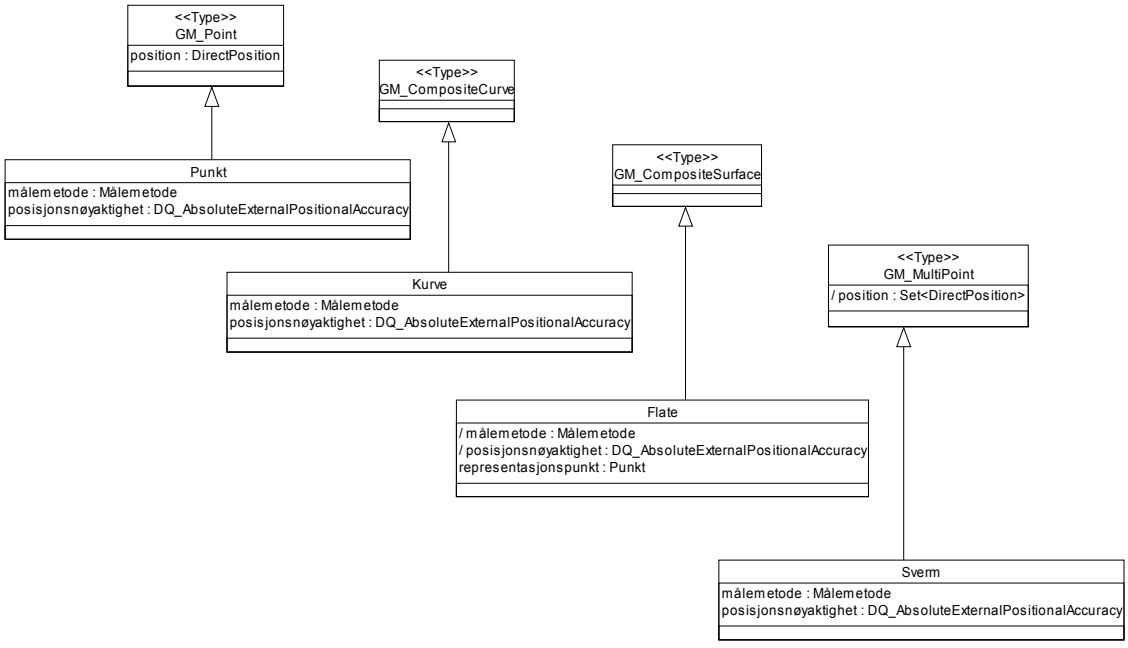
[visibility]	Spesifiserer tilgjengelighet. Det er tre muligheter (public, protected og private).Name Navnet på egenskapen.
name	Navnet på operasjonen.
[parameter-list]	Alle parametere som benyttes for operasjoner, både 'in' og 'out' parametere.
[:return-type)	Den verdien som operasjonen resulterer i. Kan være basic type eller brukerdefinert type.
{property string}	Det er 4 definerte egenskaper knyttet til operasjoner (isQuery, sequential, guarded og concurrent. Se UML for nærmere forklaring).

5.6 Modellering av geometri

5.6.1 SOSI Geometrimodell

ISO 19107 Spatial Schema inneholder klasser som dekker geometriske og topologiske primitiver. I SOSI formatet kan vi angi om en objekttype skal angis som Punkt, Sverm, Flate, Kurve, Linje, Bue, Buep, Sirkel eller SirkelP. Disse geometritypene er SOSI format spesifikke og egner seg derfor ikke i en implementasjons-uavhengig modell.

Vi tar derfor utgangspunkt i geometritypene definert i Spatial Schema og lager egne implementasjons-uavhengige geometrityper basert på disse. Fire basisgeometrityper er definert på denne måten, Punkt, Kurve, Flate og Sverm. Alle typene er subtyper av geometrityper definert i Spatial Schema, se 5.6.1. Disse geometritypene tillater deling av geometri, dvs. at en geometri kan deles av flere objekter.



Figur 33.SOSI geometrimodell basert på typer i Spatial Schema

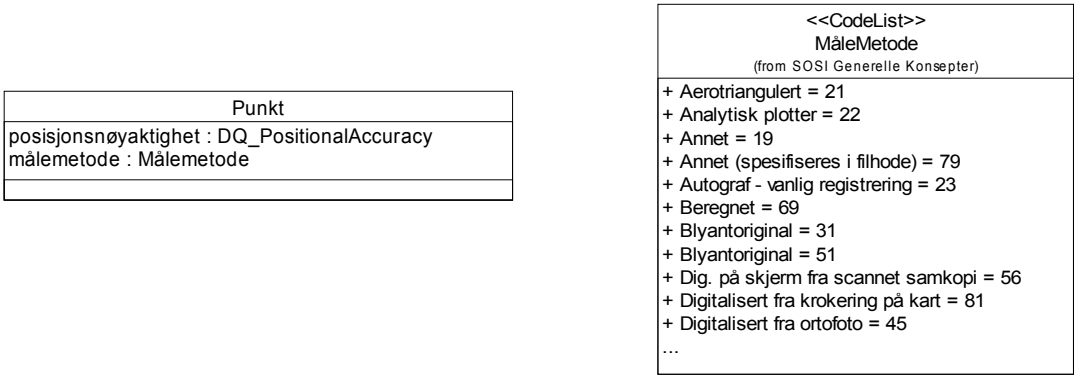
5.6.2 Kvalitet og interpolasjonsmetode

5.6.2.1 Kvalitet

Det arbeides for tiden med å lage en kvalitetsmodell. Dette kapitlet vil inneholde retningslinjer for hvordan en modellerer kvalitet inn på de respektive objekttypene i en produktspesifikasjon, samt hvordan dette kan dokumenteres for et datasett.

I SOSI-standarden står det beskrevet at alle geografiske objekter skal ha tilstrekkelig kvalitet. Dette er da tolket til at kvalitet (i form av målemetode og nøyaktighet) skal ligge på punkt og linjer.

De geometritypene vi skal forholde oss til i UML-modellene har målemetode og posisjonsnøyaktighet som egenskaper.



Figur 34.Geometritypen Punkt med kvalitetsattributter

Eksemplet over viser hvordan målemetode og posisjonsnøyaktighet er lagt på geometritypen Punkt. Den fullstendige strukturen til DQ_PositionalAccuracy er definert i ISO19115 Metadata.
Interpolasjonsmetode
SOSI definerer et sett SOSI formatspesifikke geometriegenskaper, for eksempel BUE og BUEP, som ikke egner seg i en implementasjonsuavhengig modell for en objektkatalog. Hvilken interpolasjonsmetode som skal brukes for Kurve er produktspesifikt og opp til implementasjonen.

5.6.3 Modellering av geometri og egenskapsnavn

Geometrien til objekttyper modelleres som egenskaper til objekttypen på lik linje som andre egenskaper.

Figur 35.Eksempel på bruk av geometrityper

Eksemplet i 5.6.3 viser angivelse av geometri for objekttypen Fiskebruksområde angitt med egenskapen område som representerer flaten (type Flate) og egenskapen posisjon (type Punkt) som representer plasseringen. I UML (og GML) kan en objekttype ha flere geometriegenskaper og vi kan også definere dette i SOSI objektkatalog. Det er opp til produktspesifikasjonene å begrense om bare en av geometriegenskapene skal benyttes.

Figur 36.Eksempel på flere geometriegenskaper definert for en objekttype

5.6.3 viser et annet eksempel der en objekttype har tre definerte geometriegenskaper. Kulturminner kan representeres som område, punkt eller kurve (for eksempel steingjerde).

Det er viktig å bruke folkelige, fornuftige og presise egenskapsnavn og egenskapsdefinisjoner for alle egenskaper, inkludert de som representerer geometrien, egenskapsnavnet bør si noe om hvilken rolle denne geometrien spiller for objekttypen. Generelle egenskapsnavn som 'geometri' og 'flate' holder ikke. Følgende oversikt viser ledetråder til egenskapsnavn som kan brukes for de respektive geometritypene:

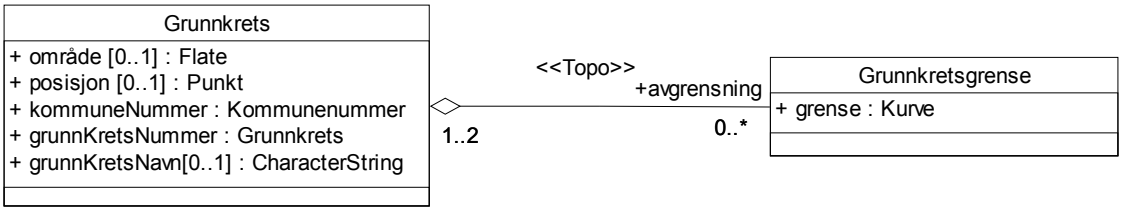
Egenskapsnavn og datatype	Definisjon
posisjon : Punkt	sted som objektet eksisterer på
senterlinje : Kurve	forløp som følger objektets sentrale del
grense : Kurve	forløp som følger overgang mellom ulike fenomener
område : Flate	objektets utstrekning

Regler:

- Objekter kan ha null, en eller flere geometriegenskaper
- Geometrien kan deles av flere objekter via sin underliggende struktur (Composite)
- Geometriegenskapene skal ha fornuftig egenskapsnavn og definisjon

5.7 Modellering av avgrensningslinjer

I SOSI formatet er det slik at flaten til et objekt dannes av avgrensningslinjene knyttet til objektet der disse avgrensingslinjene selv er objekter. Av den grunn har de fleste objekttyper som danner flater i objektkatalogen en "tilhørende" avgrensningsobjekttype. Et eksempel er objekttypene Grunnkrets og Grunnkretsgrense. 5.7 viser hvordan vi modellerer disse to objekttypene og forholdet de har seg i mellom.



Figur 37.Eksempel på modellering av avgrensninglinje for en flate

Vi sier at det er en assosiasjon mellom Grunnkrets og Grunnkretsgrense der Grunnkretsgrense spiller rollen som avgrensning for objekttypen Grunnkrets. Hvis vi tenker nøyere over semantikken (eller meningen) i denne assosiasjonen ser vi at den beskriver mer et topologisk forhold enn en "vanlig" assosiasjon. Av den grunn har vi stereotypet assosiasjonen med stereotypen <<Topo>> som beskriver at dette ikke er en helt vanlig assosiasjon, men beskriver et topologisk forhold mellom to objekttyper.

Alle objekttyper som er såkalte avgrensningsobjekttyper skal enten modelleres på denne måten med en topologisk assosiasjon til objekttypen som danner flaten eller med topologiske egenskaper som er definert i Rules for Application Schema (ISO 19109). På denne måten skaper vi mer uttrykksfulle modeller, samt at vi fanger opp hvilke avgrensningsobjekter et flateobjekt normalt vil/kan ha. Mer om topologi finnes i kapittel 5.8der vi beskriver topologimodellen i ISO 19107 Spatial Schema og hvordan vi kan nyttegjøre oss av denne i modelleringsarbeidet. Andre topologiske forhold mellom objekttyper som modellering av nettverk er også beskrevet i det avsnittet.

Ved uttrekk av en sømløs database vil alle objekter kunne avgrenses av fiktive linjer. For eksempel vil et hus kunne avgrenses av en fiktiv linje i tillegg til takkant.

Regler

- Flateobjekttyper og tilhørende avgrensingsobjekttyper modelleres med topologisk assosiasjon, se 5.7.

- En ny objektttype med flategeometri trenger nødvendigvis ikke en tilsvarende avgrensningsobjekttype, den kan avgrenses av generelle eller eksisterende avgrensningsobjekttyper
- Avgrensningslinjer som har egne egenskaper må modelleres som en objekttype

Merk: Rollen avgrensning burde ha hatt multiplisitet 1..* mange siden en Grunnkrets må ha minst en avgrensning. Siden Grunnkrets også kan forekomme som punkt lar vi multiplisiteten være 0..* mange av praktiske årsaker, multiplisiteten kan strammes inn i produktspesifikasjoner.

5.8 Topologi

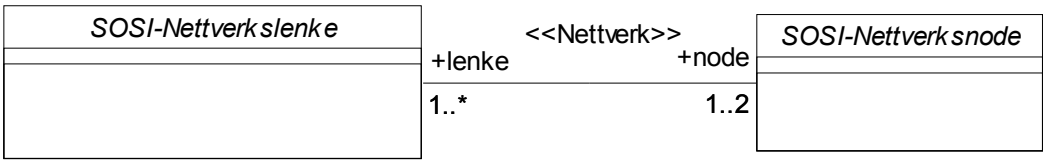
Topologi er benevnelsen på den del av matematikken som studerer de egenskaper ved geometriske former som holder ved kontinuerlige transformasjoner. De fleste objekttypene i objektkatalogen har kun tematiske (ikke-geometriske egenskaper) samt en eller flere geometriegenskaper og ingen topologiske egenskaper. Imidlertidig finnes det objekttyper som i tillegg til tematiske og geometriske egenskaper også har topologiske egenskaper. Slike objekttyper finnes for eksempel innenfor fagområder som beskriver nettverksstrukturer av varierende slag, eksempler er ledningsnett (vannforsyning, elektrisitetsnett) og vegnett. Slike topologiske egenskaper relateres til det vi kan kalle romlige forhold, altså forhold som eksisterer grunnet forhold mellom objekttypenes geometriegenskaper og ikke forholdet mellom objekttypen selv.

Noen romlige forhold kan beskrives ved hjelp av topologi, f.eks de forhold som kan beskrives med tradisjonelle romlige operatorer som *touch*, *in*, *cross*, *overlap* og *disjoint*, andre forhold som *ovenfor*, *øst for* og *foran* kan ikke beskrives ved hjelp av topologi.

ISO 19107 Spatial Schema gir oss byggeklossene til å beskrive de topologiske forhold mellom objekttyper. Dessverre har ikke SOSI formatet de samme byggeklossene og dessuten er topologidelen av Spatial Schema "upløyd mark" for de fleste. Neste avsnitt beskriver hvordan vi kan kan forenklet modeller nettverk, men avsnitt 5.8.2 beskriver kort hvordan topologi skal modelleres i henhold til ISO 19109 Rules for Application Schema.

5.8.1 Modellering av nettverk

Som beskrevet over gir ISO 19107 Spatial Schema tilgang til byggeklosser for å modellere Topologi. Vi har liten erfaring med disse begrepene og presenterer her en enkel løsning for modellering av nettverk. På tilsvarende måte som *Topo* stereotypen angir et forhold mellom flate og flatedanner (avgrensningen) angir *Nettverk* at assosiasjonen beskriver et node-lenke forhold mellom to objekttyper.

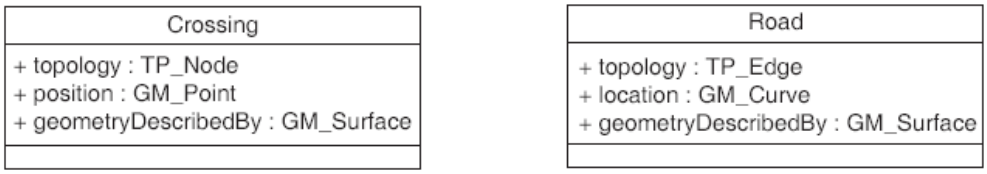


Figur 38.Modellering av nettverk med stereotypen Nettverk

Figuren over viser hvordan dette ser ut på to abstrakte node-lenke objekttyper. Denne måten å gjøre ting på vil erstattes av topologi-begrepene fra Spatial schema beskrevet i neste avsnitt.

5.8.2 Regler for modellering av romlige forhold

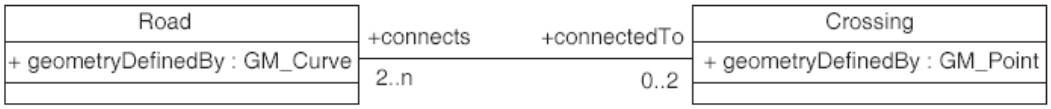
ISO 19109 Rules for Application Schema sier at romlige forhold som kan beskrives ved hjelp av topologi bør beskrives implisitt ved hjelp av de toplogiske konstruksjonene definert i Spatial Schema.



Figur 39.Eksempel på bruk av topologi konstruksjoner fra Spatial Schema

5.8.2 gir et eksempel på hvordan et enkelt vegnettverk kan modelleres med topologiske egenskaper.

Andre romlige forhold skal beskrives ved hjelp av standard UML assosiasjoner mellom objekttypene. 5.8.2 viser et eksempel på dette det romlige forholdet mellom Road og Crossing nå beskrives eksplisitt ved hjelp av en assosiasjon.



Figur 40. Eksempel på eksplisitt modellering av topologi

I modelleringen av objektkatalogen har vi valgt å kun modellere romlige forhold med eksplisitte assosiasjoner foreløpig. For å skille på vanlige assosiasjoner mellom objekttyper og romlige forhold har vi valgt å innføre stereotypen *Topo*. Bruk av denne stereotypen på en assosiasjon betyr at assosiasjonen beskriver et topologisk forhold og ikke er en tradisjonell assosiasjon. Se kapittel 5.7der vi bruker denne stereotypen for å modellere avgrensningslinjer.

5.9 Temporal Schema og andre standardmodeller

Dette kapitlet er ment å ha retningslinjer for bruk av klasser for å dekke det temporale aspektet, samt bruk av geografiske identifikatorer, dvs. ikke koordinatbasert stedfesting. Kapitlet er ikke ferdig utarbeidet. Vi refererer til følgende standarder:

ISO 19108 Geographic information – Temporal Schema
ISO 19112 Geographic information – Spatial referencing by geographic identifiers

5.10 Utvidelser av UML

UML er et standard modelleringsspråk. Ingen standardspråk er rikt nok til å dekke alle nyanser som er nødvendig innenfor en rekke ulike modelleringsområder til enhver tid. Av denne grunn har UML spesifisert hvordan en kan lage utvidelser på en kontrollert måte. Disse utvidelser omfatter:

- Stereotyper (Stereotypes)
- Tagged values
- Beskrankninger (Constraints)

Dersom en kommer opp i en situasjon hvor en ikke klarer å uttrykke seg basert på generelle UML-elementer, må en se nærmere på utvidelsesmekanismene.

5.10.1 Stereotyper (Stereotypes)

Dette er en utvidelsesmekanisme for eksisterende UML konsepter, som utvider vokabularet. Det er et modellelement som brukes for å spesialisere UML elementer slik at de behandles på en spesiell måte. Et eksempel på dette <<DataType>>, som forteller at verdidomenet til en egenskap er modellert som en egen klasse med en eller flere attributter.

Stereotyper kan anvendes på en rekke UML-elementer, men det er mest vanlig å benytte de på klasser.
En rekke stereotyper er definert i UML, vi beskriver de mest vanlige stereotypene i dette avsnittet.

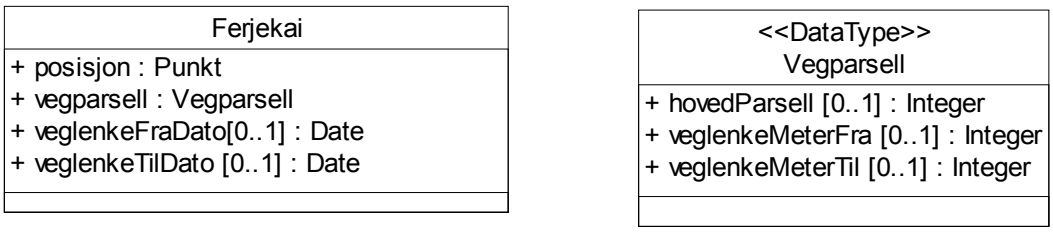
Ytterligere beskrivelse av generelle stereotyper finner i kapittel 8. Det henvises til dokumentene 'UML Specification version 1.3' og 'ISO 19103 Conceptual Schema Language'.

<<Enumeration>> Liste over predefinerte verdier

Enumeration beskriver en liste med predefinerte verdier som ikke kan utvides. Et eksempel på en slik liste er Ukedag som vist i 5.10.1

Figur 41.Eksempel på en Enumeration)

<<DataType>>
En deskriptor for et sett verdier som ikke har egen identitet, og som eksisterer uavhengig av den klassen (for eksempel en objekttype) som benytter datatypen. <<DataType>> inkluderer primitive predefinerte typer og brukerdefinerte typer.



Figur 42.Eksempel på egendefinert datatype

5.10.1 viser at en av egenskapene til objekttypen Ferjekai heter vegparsell. Verdidomenet til egenskapen er typen Vegparsell, angitt som egen klasse, Vegparsell, stereotypet <<Datatype>>.

<<CodeList>>
Beskriver en åpen liste (enumerasjon). De fleste egenskapene i SOSI - standarden er kodelister. Dette fordi nye verdier kan tildeles. Det betyr ikke at brukeren selv kan definere nye verdier, men at en gjennom en registreringsautoritet kan tildele nye verdier.

Kodelister benyttes oftest for å angi lengre lister av potensielle verdier. Verdier i en kodeliste angis ofte for brukeren, og er i sin natur mer mnemoniske (lette å huske). Ulike implementasjoner kan benytte ulike interne representasjoner for verdiene i kodelista, men da med oversettelsestabeller slik at de opprinnelige verdier kan gjenskapes. Det er i mange tilfeller ønskelig å benytte SOSI-kodeverdiene også i en implementasjon, da dette ivaretaes i standarden (utvekslingsformatet).

5.10.1 viser at en av egenskapene i objekttypen FpArealbrukOmråde heter arealbruksstatus. Denne er modellert som en egen klasse, Arealbruksstatus, stereotypet <<CodeList>.

Figur 43.Eksempel på bruk av CodeList

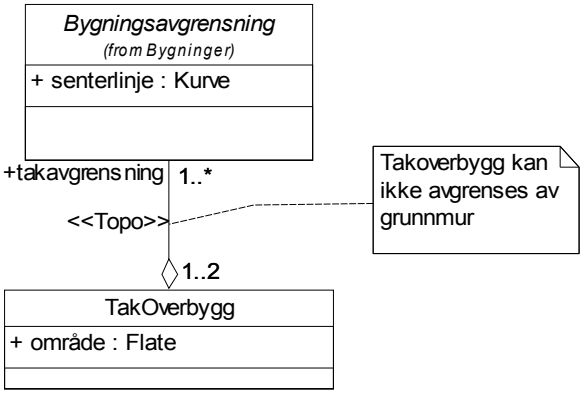
Denne kodelista består av 3 verdier, Framtidig, Nåværende og Videreutvikling av nåværende. Dersom en er helt sikker på at den ikke kan få flere verdier, burde denne være stereotypet som 'Enumeration'. Imidlertid kan en tenke seg verdier som 'kondemnert', 'ukjent', etc. Navnekonvensjoner benyttes ikke i kodelister. Her kan vi benytte både mellomrom og andre karakterer.

5.10.2 Tagged values

Tagged values er en type egenskap som kan tillegges ethvert modellelement. Består av et par (navn,verdi) i tekstform. Kan sees på som metadata informasjon som tillegges et modellelement. Vi har foreløpig ingen erfaring med bruk av denne utvidelsesmekanismen. Bør brukes med forsiktighet, fortrinnsvis unngås.

5.10.3 Restriksjoner (Constraints)

En restriksjon på modellen beskrives ved å definere denne i tilknytning til et element.



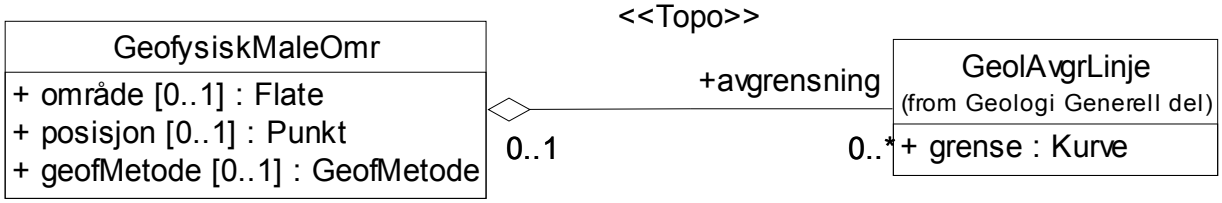
Figur 44.Eksempel på angivelse av avhengighet som note

En måte å angi dette på er å plassere beskrankninger i en 'note', og knytte noten til klassen eller assosiasjonen det gjelder. Figuren viser at Takoverbygg kan avgrenses av en bygningsavgrensning, men ikke av en bygningsavgrensning som er grunnmur. Dette er vist gjennom en stiplet linje koblet til elementet som beskrankningen gjelder for.
Merknad:
For at beskrankninger skal være maskinlesbare, må disse angis i henhold til en spesiell syntaks. Det er utviklet et eget språk , OCL (Object Constraint Language), nettopp for dette formål. Denne versjon av retningslinjene for datamodellering gir ingen retningslinjer for bruk av OCL. I de fleste eksemplene er slike avhengigheter bare angitt som ren tekst, dvs. for at vi mennesker skal forstå modellen.

5.11 Modellering av objektyper med tilhørighet i andre fagområder

Mange av fagbeskrivelsene i objektkatalogen er eller har vært tilnærmet lik en produktspesifikasjon. Av den grunn har betegnelsen 'lånte objekttyper' blitt et begrep der objekttyper som ikke er definert i et fagområde tas med i fagbeskrivelsen fordi de kompletterer en produktspesifikasjon. Nå som skillet mellom generell objektkatalog og en produktspesifikasjons objektkatalog er klarere faller disse tilfellene bort.

Det finnes imidlertidig andre tilfeller der en objekttype definert i ett fagområde for eksempel har et forhold til en objekttype i et annet fagområde. I disse tilfellene tas den eksterne objekttypen med i modellen og beskrivelsen av fagområdet. 5.11 viser et eksempel på bruk av objekttype med tilhørighet i ett annet kapittel der objekttypen modelleres ett sted, men benyttes på tvers av fagområder (her GeolAvgLinje).

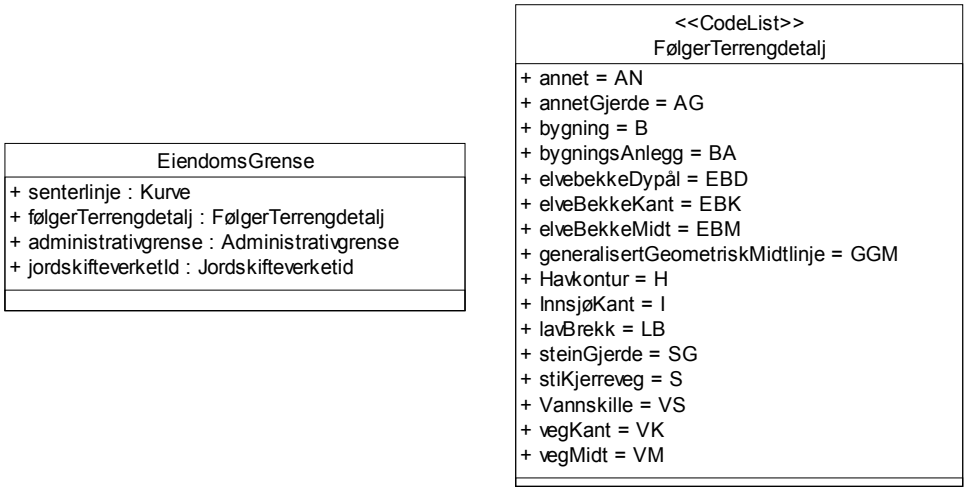


Figur 45.Eksempel på bruk av objekttype med tilhørighet i andre kapitler

I figuren vises objekttypen GeofysiskMaleområde fra fagområdet Geofysikk der objekttypen har en avgrensningsassosiasjon til GeolAvgrLinje. GeolAvgrLinje er en generell avgrensningslinje definert i fagområde Geologi som avgrenser ulike områder fra fagområder som råstoffutvinning, løsmasser og geofysikk. I denne modellen er GeolAvgrLinje hentet inn fra pakken Geologi (derav 'from Geologi' under klassenavnet) og vises i modellen for Geofysikk.

5.11.1 Spesielle tilfeller av knytninger mot andre fagområder

En annen måte å relatere objekttyper til objekttyper eller begreper i andre fagområder, er å bruke egenskapene til objekttypen. En objekttype kan være ganske generell, men ha en egenskap som klassifiserer objekttypen ytterligere. 5.11.1 viser hvordan man kan bruke en egenskap til objekttypen Eiendomsgrense for å angi hva som faktisk utgjør eiendomsgrensen.



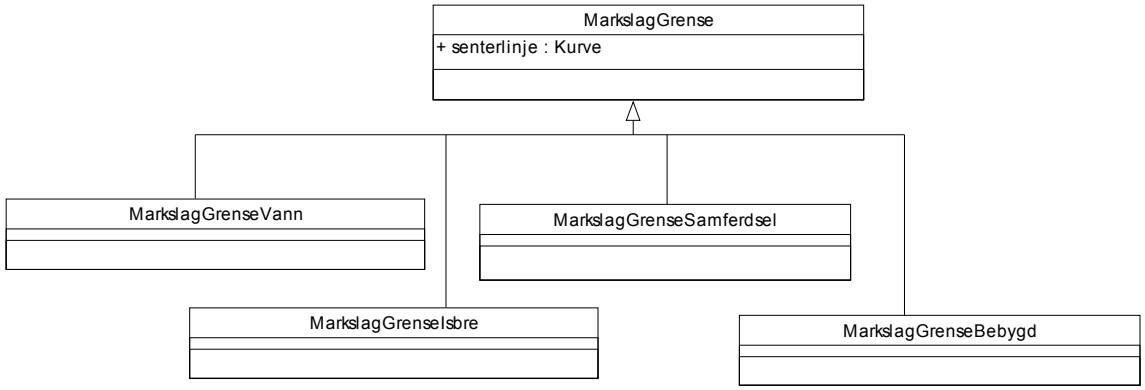
Figur 46.Eksempel på at avgrensningstype er angitt som en egenskap til objekttypen

Figuren viser en eiendomsgrense med egenskap følgerTerrengdetalj som har datatype FølgerTerrengdetalj. Her kan en modellere inn at en eiendomsgrense følger innsjøkant, bygning, steingjerde, etc., men det er ingen relasjon mellom disse objektene.

Denne måten å gjøre det på er i overensstemmelse med dagens forvaltningsløsning, men det må presiseres at dette er en veldig løs kobling mellom de respektive virkelige-verden objekter.

I objektkatalogen finner vi også avarter av eksemplet med eiendomsgrensen. Et eksempel er fra Markslag (DMK) der tilsvarende forhold modelleres inn i objekttypenavnet.

5.11.1 viser en modell der markslagsgrenser som er sammenfallende med for eksempel samferdsel og vann, er modellert inn i objekttypenavnene som MarkslagGrenseSamferdsel og MarkslagGrenseVann.



Figur 47.Eksempel på angivelse som en del av objekttypenavnet

Denne måten å gjøre det på fører til en uhensiktsmessig definering av nye objekttyper, her bør man gjøre noe tilsvarende eiendomsgrenseeksempelet over og innføre for eksempel en egenskap *grensetype*.

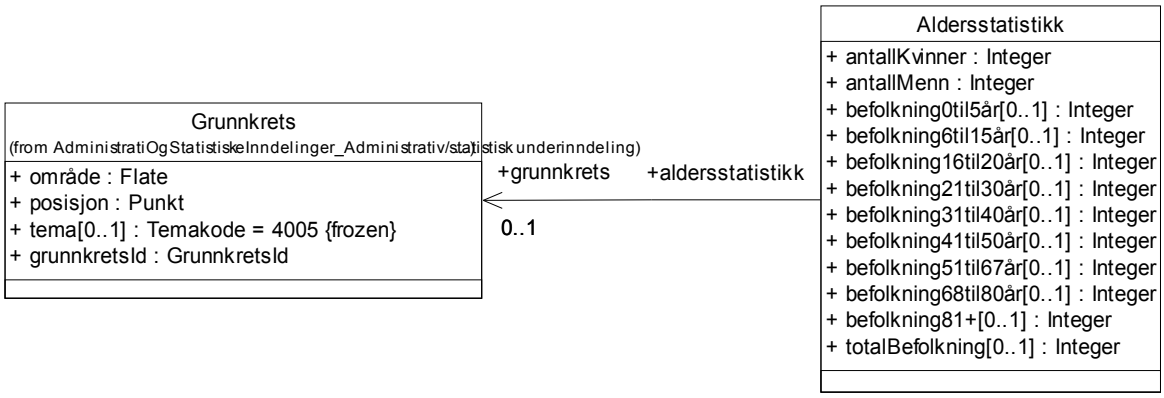
5.12 Koblede data

5.12.1 Bakgrunn

I mange tilfeller har man et ønske om å knytte ulike typer data til "allment kjente arealer" slik som kommune-, krets-, eiendoms-, steds-, stasjons- og adressebetegnelser. Eksempler på slike data er ulike typer samfunnsgeografisk informasjon, statistiske data og forurensningsdata.

5.13 Knytte data ved hjelp av assosiasjoner

For å unngå at visse objekttyper i objektkatalogen får en uhensiktsmessig mengde egenskaper som kan være nyttig i mange tilfeller bør egenskapene knyttes til objekttypene ved behov. Dette kan gjøres ved å definere egne objekttyper som har egne temaegenskaper definert i objektkatalogen, disse objekttypene er objekttyper på lik linje med andre objekttyper i SOSI men vil i mange tilfeller ikke ha noen eksplisitt stedfesting (i form av egne egenskaper som refererer til en geometritype). Skal en objekttype knyttes til en annen objekttype uttrykkes dette ved hjelp av en assosiasjon i UML-modellen. 5.13 under viser hvordan alderstatistikk kan knyttes til en Grunnkrets ved hjelp av en assosiasjon. Ved å bruke retning på assosiasjonen unngår vi å permanent knytte aldersstatistikk informasjon til grunnkrets og Grunnkrets beholder av den grunn sin opprinnelige spesifikasjon.



Figur 48.Alderstatistikk knyttet til objekttypen Grunnkrets

Vi kan altså lage objekttyper som holder ulike egenskaper som kan knyttes til objekttyper ved behov. Konklusjon Problemstillingen rører ved tre ulike deler av SOSI. Primært dreier dette seg om retningslinjer for modellering, videre har det med hvordan en skal utarbeide produktspesifikasjoner basert på SOSI objektkatalog og til sist ikke minst hvordan skal forholdene mellom ulike objekter realiseres i SOSI-formatet.

SOSI-objekt som er en supertype for alle objekttyper i SOSI inneholder en predefinert assosiasjon til seg selv. Denne kan brukes til å modellere forhold mellom alle objekttyper i SOSI om ønskelig, på den måten kan vi enkelt knytte ulike objekttyper opp mot hverandre i forskjellige modeller (les applikasjonsskjema for produktspesifikasjoner).

6 Bruk av farger i UML-modeller

Det er ingen krav om at farger skal benyttes. Dersom farger benyttes, så henvises det til eget dokument med angivelse av fargekoder. Farger bør kun benyttes dersom en vil illustrere visse aspekter ved modellen, ved strukturell informasjon må en benytte korrekt UML-notasjon (pakker o.l.) for å kunne overføre dette til andre systemer.

7 Dokumentasjon

ISO 19103 beskriver følgende (under 6.14 Documentations of models):
In addition to the diagrams, it is necessary to document the semantics of the model. The meaning of attributes, associations, operations and constraints needs shall be explained.
Documentation fields should be used extensively.

Regler:

- Alle klasser, egenskaper og assosiasjoner (roller) skal defineres i dokumentasjonsfeltet (i datamodelleringsprogrammet).

- Annen informasjon kan angis ved behov, slik som note, eksempel eller link til annen relevant informasjon.

8 Anvendelse av UML-modeller

I forbindelse med selve standardiseringsarbeidet benyttes UML-modeller i form av en grafisk notasjon.
For videre bruk benyttes modellene for implementasjon, generering av XML skjema, etc.

For å kunne utveksle modeller finnes det en egen standard, XMI. Ikke alle software-produsenter støtter denne standarden, men det er flere initiativer på gang for å bedre dette. Blant annet har OMG (Object Management Group) ute et forslag til aktivitet knyttet til UML 2.0 'Diagram Interchange'.

For å sikre at modellene kan overføres til XMI må dette taes hensyn til i modelleringssystemenes interne struktur. Dette er systemproprietært og må håndteres ulikt i de respektive systemer.

Et eksempel på dette er multiplisitet som legges inn som en del av egenskapsnavnet i Rational Rose. For å få med dette i XMI fila må dette også legges inn i Rational Rose sitt interne format.

Dette dokumentet tar ikke sikte på å gi noen detaljert rettledning i hvordan dette gjøres i ulike modelleringsverktøy, men anneks B gir noen eksempler på hvordan dette gjøres i Rational Rose.

Den kommende ISO 19136 GML-standarden vil inneholde regler for mapping fra UML-modeller til GML-skjema (XML-skjema). Dette vil påvirke hva vi benytter i våre modeller. Tilsvarende direkte mappinger til SOSI er også under vurdering.

Annex ASpesielle tilfeller(informativt)

A.1Introduksjon

Dette kapittel inneholder forslag til løsning på en rekke generelle problemstillinger som ikke er utredet tilstrekkelig. Når vi har fått en omforenet løsning, er intensjonen at dette skal løftes over til den normative delen av dokumentet.

A.2Spesielle situasjoner

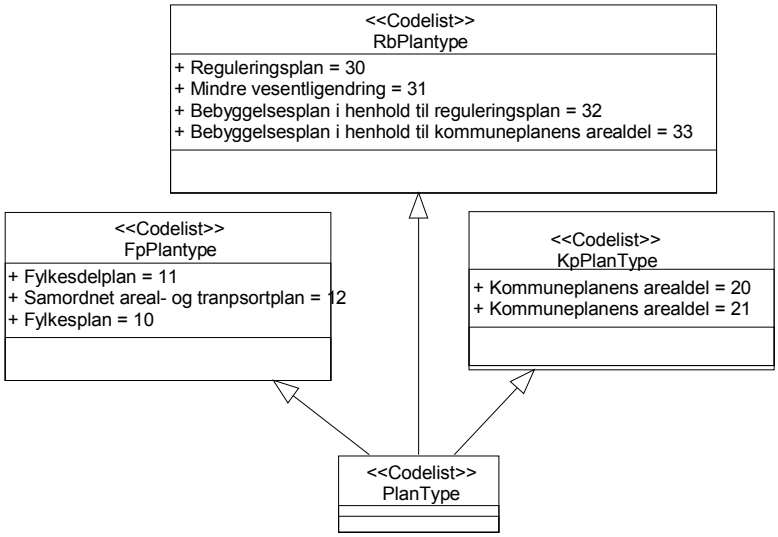
A.2.1 Begrensning av verdidomene

I enkelte sammenhenger er det behov for å angi innskrenkninger i verdiområde for en egenskap. Det er også behov for å kunne angi at kun enkelte verdier i en kodeliste er tillatt.

En korrekt syntaktisk løsning er å lage flere kodelister, og subtype denne til en kodeliste som da arver alle verdiene, dvs. det motsatte av 'subtyping' av de fullstendige kodelistene.

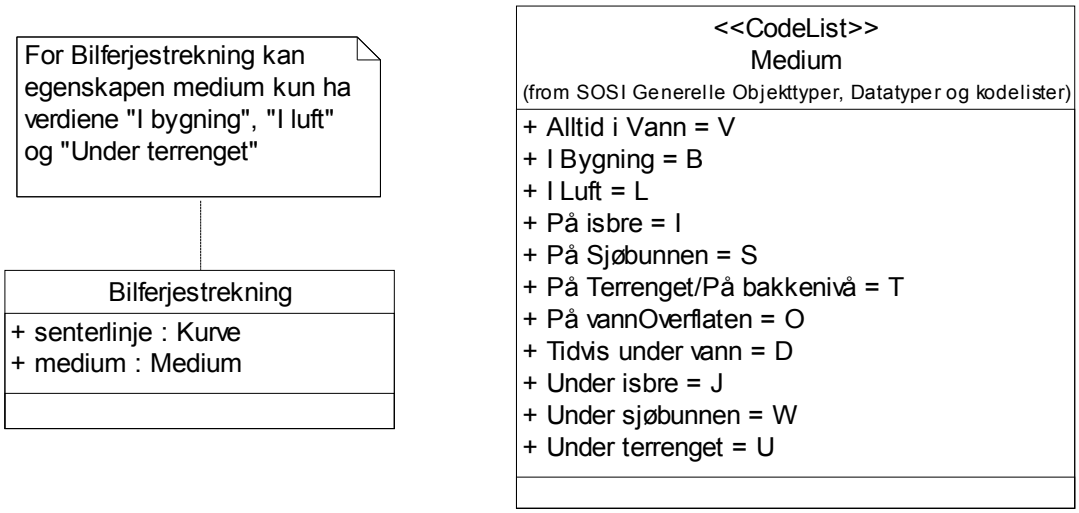
Her bør vi imidlertid være varsomme, da dette strider mot en anbefaling i ISO 19103:
ISO 19103: Multiple inheritance shall be used at a minimum, because it tends to increase model complexity.

Eksempel på oppdeling av verdidomene for plantypekoder, hvor noen verdier er lovlige for fylkesplan, noen for reguleringsplan og noen for kommuneplan. Dersom en har behov for å angi alle disse verdiene samlet, lages en ny kodeliste som arver fra de tre andre. Dette er det som kalles multippel arv, og er ikke anbefalt da det ofte øker kompleksiteten i modellen. Imidlertid kan dette gjøres for lettere å se 'mappingen' mot SOSI, som har intensiv bruk av slike kodelister.



Figur 49.Eksempel på 'subtyping' av kodelister

Alternativet til 'subtyping' er å angi begrensning i verdiområde i en Note. 8 viser objekttypen Bilferjestrekning med egenskapen medium. Via en note indikeres det hvilke verdier fra kodelisten Medium som kan brukes av Bilferjestrekning.



Figur 50.Eksempel på innskrenkning av verdier ved bruk av note

Det må påpekes at dette kun er en note til modellen og ikke noe som ivaretas ved en eventuell mapping fra UML. Helst skal slike beskrankninger skrives som OCL uttrykk.

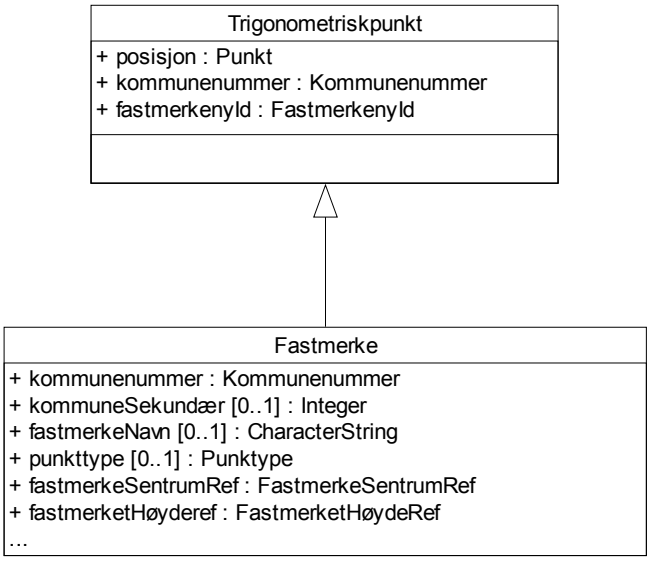
For mange slike noter forvansker modellen, og gjør det umulig å ivareta ved automatisk bruk.

Regler:

- 'Subtyping' av kodelister bør så langt som mulig unngås.
- Lager kodelister som dekker de verdidområder en har bruk for.
- Dersom en også trenger å benytte koder fra flere kodelister, 'subtypes' disse, jfr. 8
- Ved enkle verdibegrensninger kan dette også angis som en note i modellen, og må handteres manuelt i en implementasjon.

A.2.2 Avledninger

For avledede data er det viktig å modellere at disse er avledet fra primærdata, både rent praktisk med tanke på datafangst og vedlikehold. Hvordan modellerer vi at en objekttype i N50 for eksempel er avledet fra en tilsvarende objekttype i primærdata. 8 viser hvordan vi kan løse dette med generalisering/spesialisering for å gjøre modellen egnet for ulike anvendelser med tanke på detaljeringsgrad.



Figur 51.Eksempel på bruk av spesialisering som alternativ til avledning

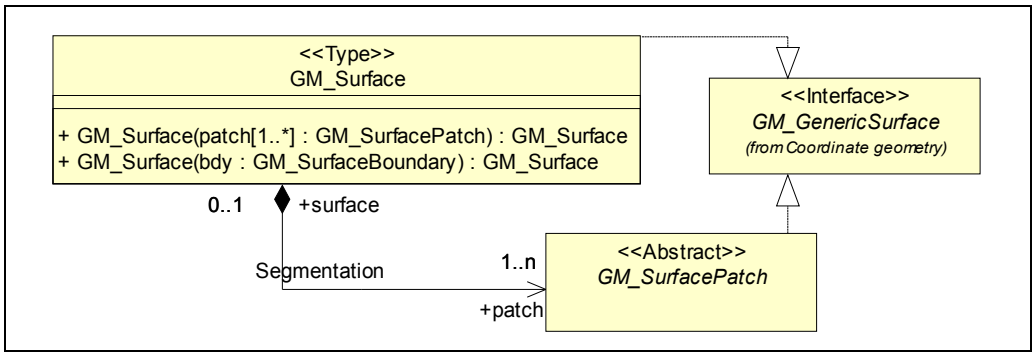
Regler:

- Ta hensyn til slik bruk på et tidlig stadium ved å innføre generalisering/spesialisering.
- Forholdet mellom opprinnelig og avledet objekttype angis med generalisering/spesialisering. (Relasjonen mellom Fastmerke og Trigonometriskpunkt).
- En note legges inn for tekstlig å beskrive selve avledningen.
- Multiplisitet angis. Ingen angivelse betyr 1.
- Egenskaper som avledes fra den opprinnelige klassen beskrives som avledet '/'. (Eks. høyde, diverse, geometri).

A.2.3 Spesielle stereotyper (definert i ISO 19103)

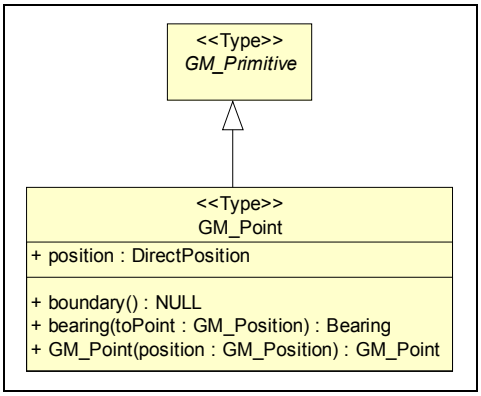
Dette kapittel omhandler stereotyper definert i ISO 19103 Conceptual Schema Language som kan anvendes i modellene. For modellering av geografiske objekter skal disse brukes med stor forsiktighet, dvs. at de er nødvendige kun ved spesielle behov og når man vet hva bruken av stereotypene innebærer. Skal en derimot modellere tjenester eller metamodeller, eller tilrettelegge for automatisk implementasjon, kan disse benyttes.

<<Interface>>
Spesifiserer en samling av operasjoner som er brukt for å spesifisere de tjenester som en klasse eller komponent bærer.



Figur 52.Eksempel på angivelse av interface

<<Type>>
Spesifiserer en abstrakt klasse som kun brukes til å spesifisere struktur og oppførsel til et sett objekter. Spesifiserer ikke implementasjon.



Figur 53.Eksempel på angivelse av <<Type>>

<<Control>>
Benyttes for en klasse hvis viktigste oppgave er å tilby en tjeneste uten å representere noen spesielle data.

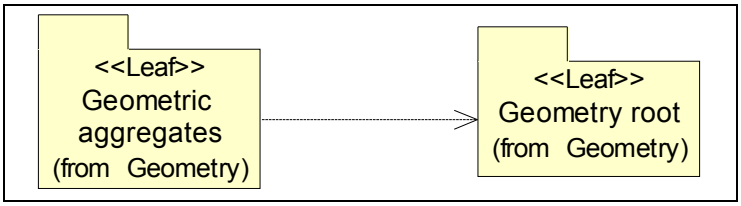
<<Entity>>
En entititet spesifiserer persistente informasjonsobjekter i en systemmodell.

<<Boundary>>
Benyttes for en klasse som representerer et eksternt grensesnitt for et system.

<<Exception>>
Spesifiserer en hendelse som er avvist av de underliggende prosesser

<<MetaClass>>
Benyttes for å angi en klasse hvor alle instanser i klassen er egne klasser. Typisk brukt i spesifikasjonen av metamodeller, hvor en klasse har som primæroppgave å holde metadata om en annen klasse. Benyttes ikke for vanlige applikasjonsskjemaer.

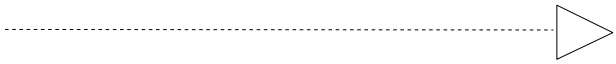
<<Leaf>>
En pakke som inneholder definisjoner, uten sub-pakker. Dvs. laveste nivå.



Figur 54.Eksempel på bruk av <<Leaf>>

A.2.4 Realisering (Realization)

En realisering beskriver et forhold mellom to modellelementer der det ene elementet spesifiserer en kontrakt/avtale som det andre elementet garanterer å utføre. Forholdet kan beskrives som en krysning mellom avhengighet og generalisering, og angis med stiplet linje med en stor åpen pil i den ene enden.



Bruk av realisering mellom objekttyper er forekommer sjelden, men gjør man dette har en realisering følgende semantikk:

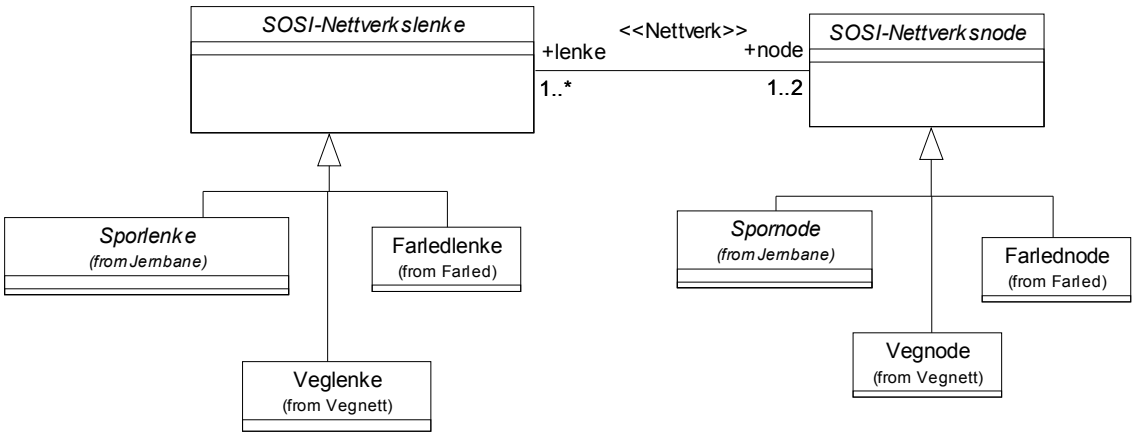
- Objekttypen på streksiden støtter minst alle operasjoner definert i objekttypen på pilsiden.
- Objekttypen på streksiden trenger nødvendigvis ikke å støtte/bruke egenskapene og assosiasjonene (strukturen) til objekttypen på pilsiden.

Som oftest brukes realisering for å spesifisere forholdet mellom ett interface og en klasse eller komponent. Ett interface spesifiserer en slags kontrakt som et sett operasjoner der en klasse eller komponent via en realisering garanterer å oppfylle kontrakten. Man sier at klassen/komponenten realiserer/implementer et interface.

Ved modellering av geografiske objekter bruker vi sjelden realisering og er en mekanisme vi skal være forsiktige med. Som nevnt kan den brukes mellom to objekttyper, men normalt vil brukes realisering ved modellering av applikasjoner der en spesifiserer komponenter og interface'ene de realiserer. Når vi modellerer geografiske objekter og rene informasjonsmodeller bruker vi ikke interface'er og realisering forholdet brukes dermed sjeldent. Skal en derimot modellere tjenester og applikasjoner er forholdet mellom et interface og komponenten(e) som realiserer interfacet sentralt.

A.2.5 Romlige forhold og modellering av nettverk

I flere av fagområdene innenfor SOSI arbeidet finnes det nettverksmodeller av varierende art. Under modelleringsarbeidet har det vært fremmet forslag om å komme opp med og enes om en felles overordnet nettverksmodell. Nå som vi har innført begrepet romlige assosiasjoner og stereotypen *Nettverk* kan vi se på hvordan en slik overordnet nettverksmodell kan se ut. 8 viser en enkel versjon av slik nettverksmodell.



Figur 55.Eksempel på mulig (enkel) overordnet SOSI nettverksmodell

Eksempelet viser to abstrakte objekttyper som er supertyper for alle node-lenke objekttyper i SOSI. Disse supertypene kan inneholde generelle egenskaper samt at de ivaretar det topologiske forholdet mellom en node og en lenke ved hjelp av en topologisk assosiasjon. Subtypene er her objekttyper tilhørende samferdsel som typisk har behov for en slik nettverksmodell. I kapittel 4.5.3 viser vi et eksempel på en jernbanemodell der Sporlenke og Spornode i 8 er detaljert beskrevet. Jernbanemodellen er et eksempel på hvordan en nettverksmodell basert på denne overordnede modellen kan beskrives.

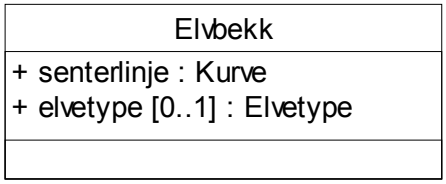
Annex BModellering i Rational Rose

B.1Introduksjon

UML er en grafisk presentasjon og det er det som vises i den grafiske modellen som er normativt. Flere systemleverandører har imidlertid mulighet for å legge inn informasjon i sin interne struktur, uten at det er noen grafisk visning av dette.

B.2Multiplisitet

Eksempelvis legges kardinalitet inn som en del av egenskapsnavnet for Rational Rose, mens det for Visio handteres separat. Disse forskjellene er imidlertid viktige med tanke på å bruke disse modellene til å generere XMI. Her må en innenfor hvert enkelt system legge inn nødvendig informasjon slik at det er mulig å generere korrekt XMI.



Figur 56.Angivelse av multiplisitet - grafisk notasjon

Annex CUtestående

C.1Områder som trenger videre vurdering

- Modellering med topologi-begreper fra ISO 19108 Temporal Schema
- Bruk av OCL for å beskrive restriksjoner/beskrankinger på modellene (constraints).
- Retningslinjer for modellering av tjenester
- Modellering av objekter som dekker flere behov. Eksempelvis kystkontur som både kan være bygningslinje samt vanlig kystkontur.
- Avledninger (ikke minst med tanke på N50, N250, etc).