Digitaliseringsdirektoratet
Norwegian Digitalisation Agency

Veileder for beskrivelse av informasjonsmodeller

Innmelding av feil og mangler:
Dersom du finner feil eller mangler i dokumentet, ber vi om at dette meldes inn på Github Issues. Dersom du ikke allerede har bruker på Github kan du opprette bruker gratis.

Status: under utarbeidelse
Versjon: forberedelse til 1.0
Publisert: under utarbeidelse
Oppdatert: 2021-06-29
Gjeldende versjon: https://data.norge.no/guide/veileder-modelldcat-ap-no/
Redaktørens utkast: https://informasjonsforvaltning.github.io/veileder-modelldcat-ap-no/

Innholdsfortegnelse

Innledning

Hensikt og avgrensning

Denne veilederen skal gi en hjelp til virksomhetene i offentlig forvaltning til å beskrive informasjonsmodellene sine i henhold til *Spesifikasjonen for beskrivelse av informasjonsmodeller (ModellDCAT-AP-NO)* og tilgjengeliggjøre dem i Felles datakatalog. Den gjelder altså først og fremst virksomheter som allerede har informasjonsmodeller som skal høstes, og ikke de som skal starte opp informasjonsmodellering.

Ulike virksomheter bruker ofte ulike modelleringsspråk og -verktøyer. Selv om noen etater forholder seg til de samme modelleringsspråkene og -verktøyene, kan bruken av disse og hvilke versjoner man anvender variere. Hensikten med ModellDCAT-AP-NO er å kunne beskrive virksomhetenes ulike informasjonsmodeller på en ensartet måte. Gjennom en lik tilnærming, er det enklere å finne, forstå, sammenligne og gjenbruke informasjonsmodeller i offentlig sektor.

Det er ikke et mål for denne veilederen å beskrive alle felter i ModellDCAT-AP-NO, heller ikke teknisk og normativt. For normative beskrivelser av alle feltene i spesifikasjonen for informasjonsmodeller, se ModellDCAT-AP-NO og valideringsreglene (shacl).

Målgruppe

Målgruppen for denne veilederen er deg som:

- skal bruke ModellDCAT-AP-NO for å beskrive eksisterende informasjonsmodeller i virksomheten din
- skal utvikle/tilpasse verktøystøtte i virksomheten din for beskrivelse av informasjonsmodeller og/eller publisering av disse i henhold til ModellDCAT-AP-NO.
- ønsker å få bedre forståelse av ModellDCAT-AP-NO
- ønsker å tilgjengeliggjøre informasjonsmodeller i virksomheten din i Felles datakatalog.
- ønsker bedre å finne, forstå, sammenligne og gjenbruke modeller beskrevet i henhold til ModellDCAT-AP-NO
- ønsker å bruke erfaring med og kunnskap om ModellDCAT-AP-NO inn i diskusjoner om relevante, internasjonale standarder

For å få best mulig innsikt i ModellDCAT-AP-NO og utbytte av veilederen, er det en fordel om du har litt kjennskap til Resource Description Framework (RDF).

Struktur

Veilederen består av følgende deler:

- **Kort om ModellDCAT-AP-NO** – gir en kort innføring i spesifikasjonen for beskrivelse av informasjonsmodeller.
- **Generelt** – generell del som beskriver egenskaper og vokabularer som brukes på tvers av

klasser.

- **Informasjonsmodell** – omhandler hvordan man beskriver informasjonsmodeller.
- **Modellinnhold** – tar for seg hvordan man beskriver modellelementer og egenskaper.
- **Modellkatalog** – omhandler hvordan man beskriver en modellkatalog
- **Informasjonsmodell og relasjoner til begreper, datasett og datatjenester** – beskriver hvordan man knytter informasjonsmodeller sammen med øvrige katalogressurser.

I tillegg er det laget et vedlegg, **Informasjonsmodellering enkelt forklart** som gir en kort innføring i hva informasjonsmodeller er og anbefalt metode for utforming av disse.

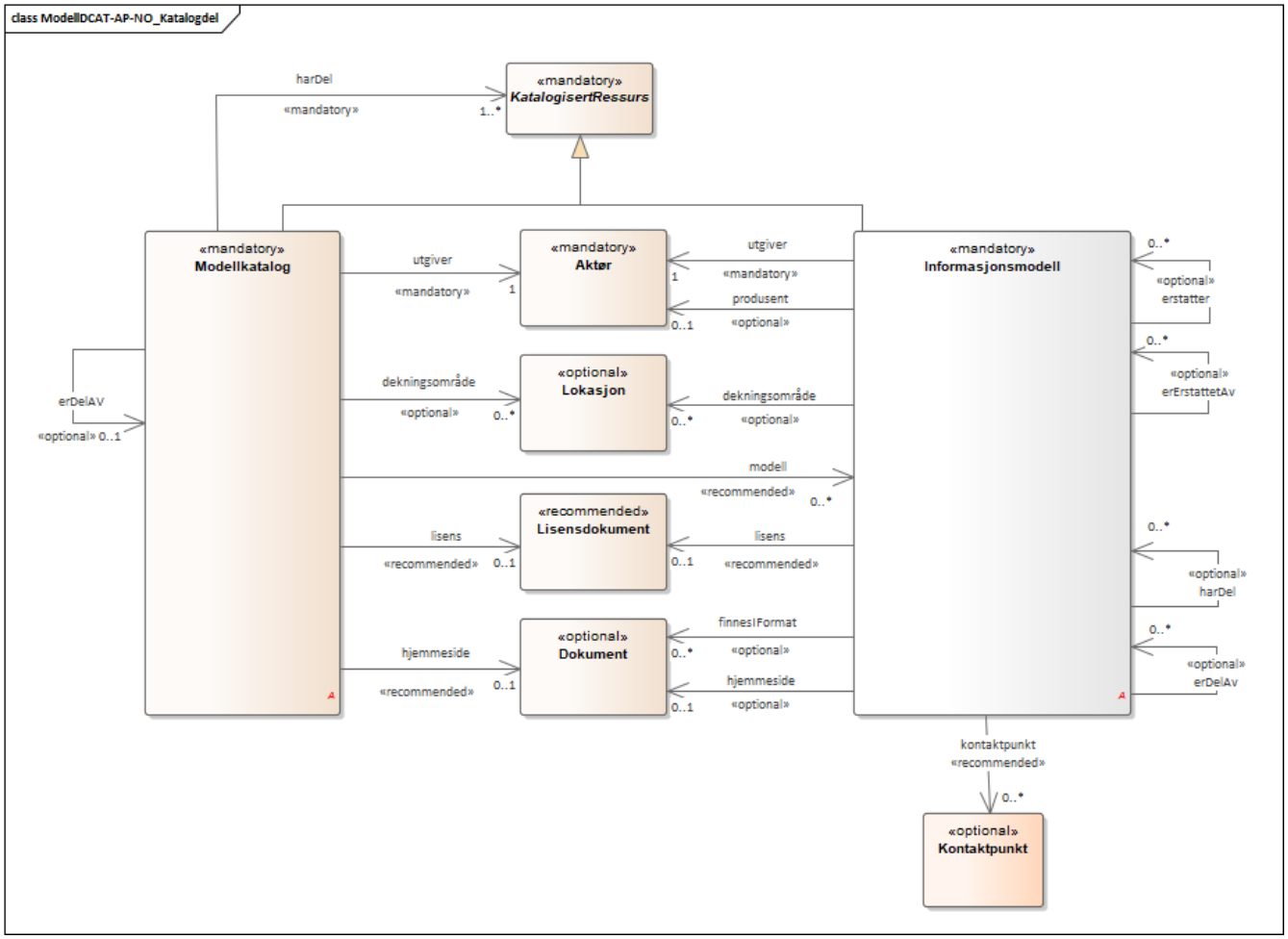
Navnerom som er brukt i veilederen

Prefiks	Navnerom	Forklaring/navn
adms	http://www.w3.org/ns/adms#	Asset Description Metadata Schema
dcat	http://www.w3.org/ns/dcat#	Data Catalog Vocabulary
dct	http://purl.org/dc/terms/	DCMI Metadata Terms
foaf	http://xmlns.com/foaf/0.1/	FOAF Vocabulary
modelldcatno	https://data.norge.no/vocabulary/modelldcatno#	Spesifikasjon for beskrivelse av informasjonsmodeller (ModellDCAT-AP-NO)
owl	http://www.w3.org/2002/07/owl#	OWL Web Ontology Language
xsd	http://www.w3.org/2001/XMLSchema#	XML Schema Part 2: Datatypes Second Edition

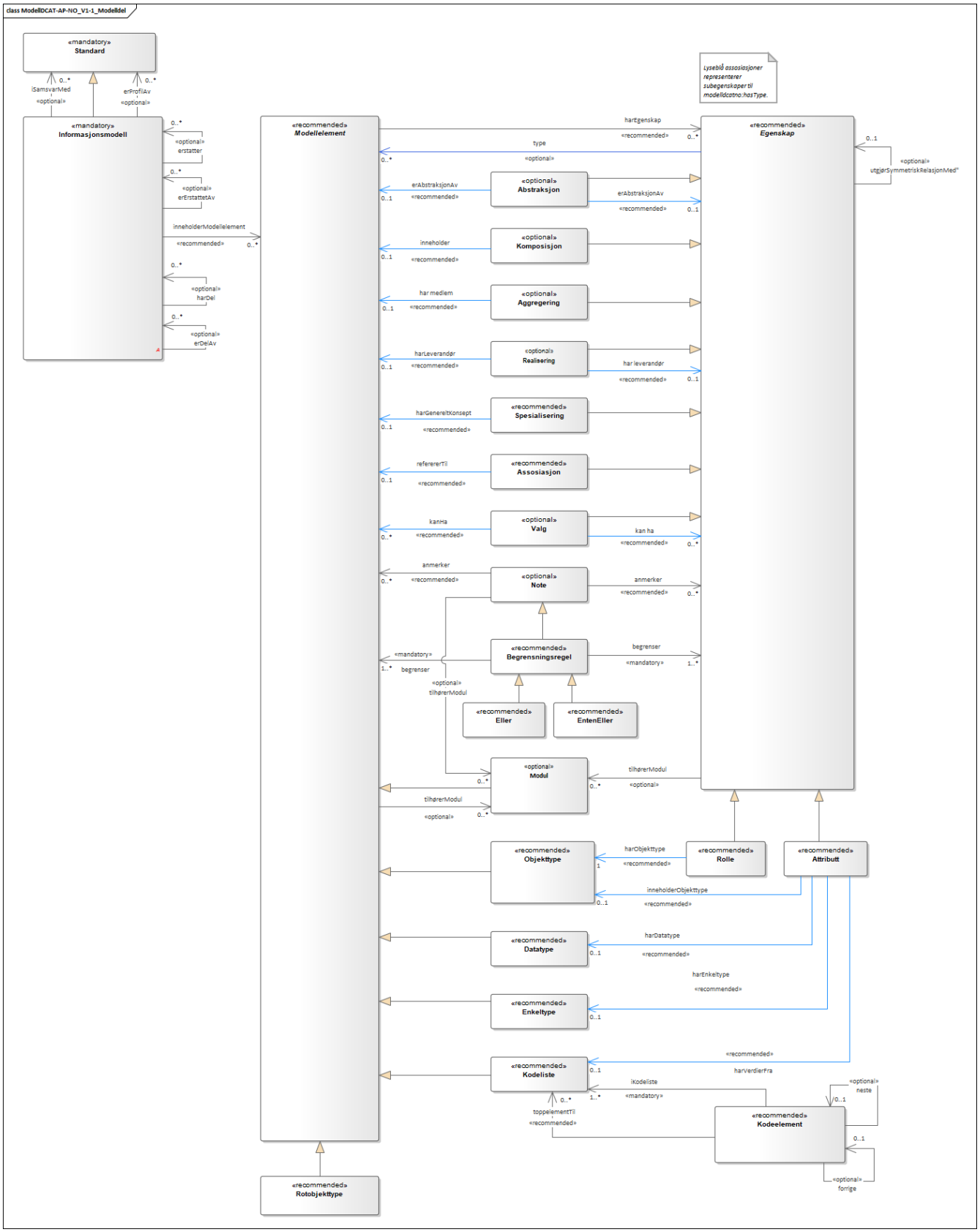
Kort om ModellDCAT-AP-NO

ModellDCAT-AP-NO er basert på *Standard for beskrivelse av datasett, datatjenester og datakataloger (DCAT-AP-NO)* og inngår i Rammeverk for informasjonsforvaltning.

Figurene nedenfor viser en forenklet modell av ModellDCAT-AP-NO beskrevet i UML. Egenskapene er utelatt, og kun de mest sentrale klassene og relasjonene er tatt med. Modellene viser kun de norske klasse- og relasjonsnavnene. For fullstendig oversikt, se spesifikasjonen for ModellDCAT-AP-NO.



Figur 1. Forenklet modell for ModellDCAT-AP-NO - Katalogdel



Figur 2. Forenklet modell for ModellDCAT-AP-NO - Modelldel

Spesifikasjonen består av to hoveddeler, en katalogdel som er basert på DCAT-AP-NO 2.0 (lysebrune klasser) og en utvidet del, modelldel (grå klasser), som beskriver en informasjonsmodell og dens innhold. Som datasett og datatjenester, er informasjonsmodell beskrevet som en subklasse til klassen Katalogisert ressurs (`dcat:Resource`).

I ModellDCAT-AP-NO er det definert noen overordnede klasser. Disse er:

- Katalogisert ressurs (`dcat:Resource`)
- Modellelement (`modelldcatno:ModelElement`)
- Egenskap (`modelldcatno:Property`)

Disse skal i en konkret bruk erstattes med en av de spesifikke subklassene.

Siden ModellDCAT-AP-NO er basert på DCAT-AP-NO 2.0, er spesifikasjonen beskrevet i RDF. Det betyr at modellen (målmodellen) som du skal transformere til, også må være i RDF.

Generelt

Om bruk av egenskap «identifikator» (dct:identifier)

Egenskapen `dct:identifier` brukes til å oppgi identifikatoren til subjektet (første leddet) i en RDF-trippel. Subjektet i en RDF-trippel er per definisjon en identifikator (URI). I en konkret realisering vil instanser av klassene (inkl. deres subklasser) Modellkatalog (`dcat:Catalog`), Informasjonsmodell (`modelldcatno:InformationModel`), Modellelement (`modellcatno:ModelElement`), Egenskap (`modelldcatno:Property`), Kodeliste (`modelldcatno:CodeListe`) og Kodeelement (`modelldcatno:CodeElement`) derfor få en «innebygd» identifikator. Det er med andre ord strengt tatt ikke nødvendig å ha en identifikator (`dct:identifier`) i tillegg til den «innebygde» identifikatoren.

Egenskapen `dct:identifier` er derfor satt som anbefalt og ikke obligatorisk i de nevnte klasser og deres subklasser, med unntak av for Modellkatalog (`dcat:Catalog`). `dct:identifier` er satt som obligatorisk for Modellkatalog (`dcat:Catalog`) for at ModellDCAT-AP-NO skal være kompatibel med DCAT-AP-NO, ettersom `dct:identifier` er obligatorisk for `dcat:Catalog` der.

Egenskapen `dct:identifier` trenger ikke å inneholde den samme identifikatoren som den innebygde URIen i en RDF-trippel, men når det er den samme identifikatoren, anbefales det at hele den innebygde URIen (subjektet i en trippel) kopieres til `dct:identifier` i en Modellkatalog (`dcat:Catalog`).

Eksempel i RDF Turtle:

```
@prefix dct:  <http://purl.org/dc/terms/> .
@prefix dcat: <http://www.w3.org/ns/dcat#> .
@prefix xsd:  <http://www.w3.org/2001/XMLSchema#> .

<https://examples.com/infomoc/exmodcat1> a dcat:Catalog ;
   dct:identifier "https://examples.com/infomod/exmodcat1"^^xsd:anyURI .
```

Identifikatoren bør utformes i henhold til *Standard for pekere til offentlige ressurser på nett*.

Om bruk av egenskap «utgiver» (dct:publisher) og «produsent» (dct:creator)

Det anbefales å bruke følgende mønster for URI til utgiver (`dct:publisher`) og produsent (`dct:creator`), der det siste leddet er organisasjonsnummeret:

```
<enInfoModell> dct:publisher <https://organization-
catalogue.fellesdatakatalog.digdir.no/organizations/974760673> .
```

Når det er behov for å oppgi at «eieren» for en informasjonsmodell som ikke er den samme som utgiveren (`dct:publisher`), anbefales det å bruke produsent (`dct:creator`).

Om bruk av egenskap «dekningsområde» (dct:spatial)

Til dekningsområde (`dct:spatial`), anbefales det å bruke EU sine kontrollerte vokabularer Continent, Country, og Place. Eksemplet under (i RDF Turtle) viser bruk av Country-vokabularet for å oppgi Norge (NOR):

```
<enInfoModell> dct:spatial
<http://publications.europa.eu/resource/authority/country/NOR> . # Norge
```

Det anbefales å bruke Kartverket sine kontrollerte vokabularer for fylke og kommune. Eksemplene under viser Oslo som fylke hhv. kommune.

```
<mod1> dct:spatial <https://data.geonorge.no/administrativeEnheter/fylke/id/173159> .
# Oslo som fylke

<mod2> dct:spatial <https://data.geonorge.no/administrativeEnheter/kommune/id/173018>
. # Oslo som kommune
```

Du kan finne fram til URIene her.

Om bruk av egenskap «lisens» (dct:license)

For lisens (`dct:license`) anbefales det å bruke EU sitt kontrollerte vokabular for lisens, Licence, hvis lisensen du bruker finnes på listen. I eksemplet under er det CC BY v. 4.0 som er brukt.

```
<enInfoModell> dct:license
<http://publications.europa.eu/resource/authority/licence/CC_BY_4_0> .
```

Informasjonsmodell

Hva er en informasjonsmodell?

Med informasjonsmodell (`modelldcatno:InformationModel`) mener vi en formell beskrivelse av informasjonen en virksomhet trenger å motta eller selv produsere for å utføre sitt daglige virke. Vi legger til grunn en vid tolkning av begrepet informasjonsmodell. Den omfatter modeller på ulike abstraksjonsnivåer (som konseptuelle, logiske og fysiske modeller) og modeller som beskriver ulike aspekter ved informasjon (som felles- og anvendelsesmodeller).

For ModellDCAT-AP-NO er det utarbeidet et vokabular for klassifisering av modeller. Dette er:

- **Konseptuell modell** er en form for kvalitativ modell som beskriver de viktigste konseptene innenfor et fagdomene og sammenhengen mellom disse.
- **Logisk modell** beskriver hvilke typer informasjon som inngår i en avgrenset sammenheng og hvordan de er logisk relatert, uavhengig av teknologi.
- **Fysisk modell** er en logisk modell som er utarbeidet for å beskrive datautveksling eller lagring av data for en bestemt løsning.
- **Fellesmodell** er en informasjonsmodell til felles bruk på tvers av virksomheter, forretningsområder og/eller applikasjonssegmenter.
- **Anvendelsesmodell** er en modell som er rettet mot et spesifikt anvendelsesområde i en avgrenset kontekst, og er sammensatt av elementer i en fellesmodell.

Hvordan beskriver du en informasjonsmodell?

Kun tittel (`dct:title`) og utgiver (`dct:publisher`, se Om … «utgiver») er obligatorisk for en informasjonsmodell. Det er med andre ord ikke påkrevd å ta med modellelementer i modellbeskrivelsen din, selv om dette er anbefalt. Hvis modellen din kun er tilgjengelig som f.eks. en bildefil, anbefales det å gjøre den tilgjengelig på en nettside og peke til denne ved bruk av dct:hasFormat. Det er også mulig å referere til en hjemmeside hvor modellen er nærmere beskrevet ved bruk av egenskapen `foaf:homepage`.

Hva som betraktes som større eller mindre endringer, er en faglig vurdering som denne veilederen ikke dekker. Når du ut fra den faglige vurderingen mener at endringene i informasjonsmodellen din er så store at det kan ha betydning for hvordan informasjonsmodellen skal forstås/brukes, bør du opprette en ny instans av `modelldcatno:InformationModel` for den nye versjonen av informasjonsmodellen din, slik at den nye versjonen også får en egen identifikator. Dette gjør at både den nye og den gamle versjonen av informasjonsmodellen din kan refereres til. Du bør også bruke egenskapen «Informasjonsmodell: erstatter» (`dct:replaces`) fra den nye versjonen til den gamle versjonen av informasjonsmodellen (ev. den motsatte egenskapen, «Informasjonsmodell: er erstattet av» (`dct:isReplacedBy`) fra den gamle til den nye versjonen av informasjonsmodellen). For mindre endringer kan du bruke egenskapene «Informasjonsmodell: versjon» (`owl:versionInfo`) og «Informasjonsmodell: versjonsnote» (`adms:versionNotes`) til å dokumentere endringene, uten at det blir opprettet en ny instans og dermed en ny identifikator.

Eksempel på en informasjonsmodell (i RDF Turtle)

Se eksempel 1 i RDF Turtle, som er en informasjonsmodell uten modellelementer, men med henvisning til hvor modellen finnes. Eksemplet viser også bruk av alle egenskapene (dvs. ikke bare obligatoriske).

Se eksempel 2 i RDF Turtle, som er en informasjonsmodell med et modellelement. Eksemplet viser bruk av kun obligatoriske egenskaper.

Hva gjør du når du skal gjøre din informasjonsmodell tilgjengelig?

To tilfeller:

- Uten modellelementer, men med lenke til hjemmeside der selve modellen finnes, se eksempel 1 ovenfor.
- Med modellelementer, slik at hele modellen blir tilgjengeliggjort for automatisk høsting til modellkatalogen, se eksempel 2 ovenfor.

Modellinnhold

Basis modellelementer og egenskaper

Objekttype: beskriver en klasse av objekter med felles egenskaper.

Datatype: beskriver en sammensatt verdistruktur uten identitet.

Attributt: beskriver en basisegenskap ved en objekttype eller datatype.

Enkeltype: beskriver verdidomenet for et attributt.

Rolle: relasjon som beskriver en rolle et objekt har overfor et annet.

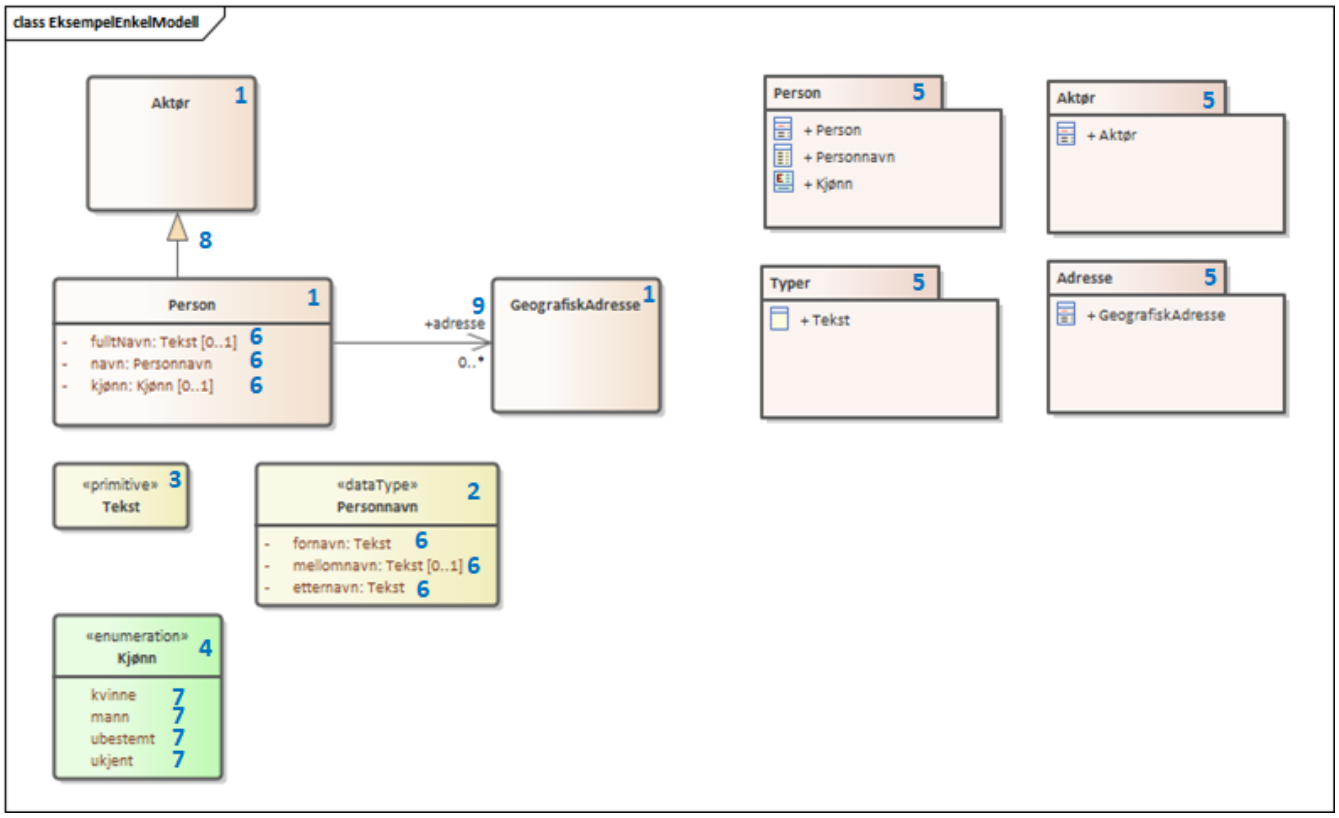
Kodeliste: beskriver et sett av lovlige verdier for et attributt.

Kodeelement: representere et navngitt og unikt element i en kodeliste.

Spesialisering: beskriver et arveforhold mellom modellelementer, hvor en subtype er en spesialisering av en mer generell type (supertype).

Modul: representerer en delmodell eller kategori som modellelementer og egenskaper i en modell kan grupperes under.

Figuren nedenfor viser en enkel informasjonsmodell med de mest grunnleggende modellelementene. Modellelementene og egenskap er nummerert etter type.



Figur 3. Eksempel - en enkel informasjonsmodell

Mapping til ModellDCAT-AP-NO:

Modellelement-/egenskapstype	**UML-representasjon**	**ModellDCAT-AP-NO**
1	Class	Objekttype (`modelldcatno:ObjectType`)
2	DataType	Datatype (`modelldcatno:Datatype`)
3	Primitive	Enkeltype (`modelldcatno:SimpleType`)
4	Enumeration	Kodeliste (`modelldcatno:CodeList`
5	Package	Modul (`modelldcatno:Module`)
6	Attribute	Attributt (`modelldcatno:Attribute`)
7	Literal	Kodeelement (`modelldcatno:CodeElement`)
8	Generalization	Spesialisering (`modelldcatno:Specialization`)
9	Role	Rolle (`modelldcatno:Role`)

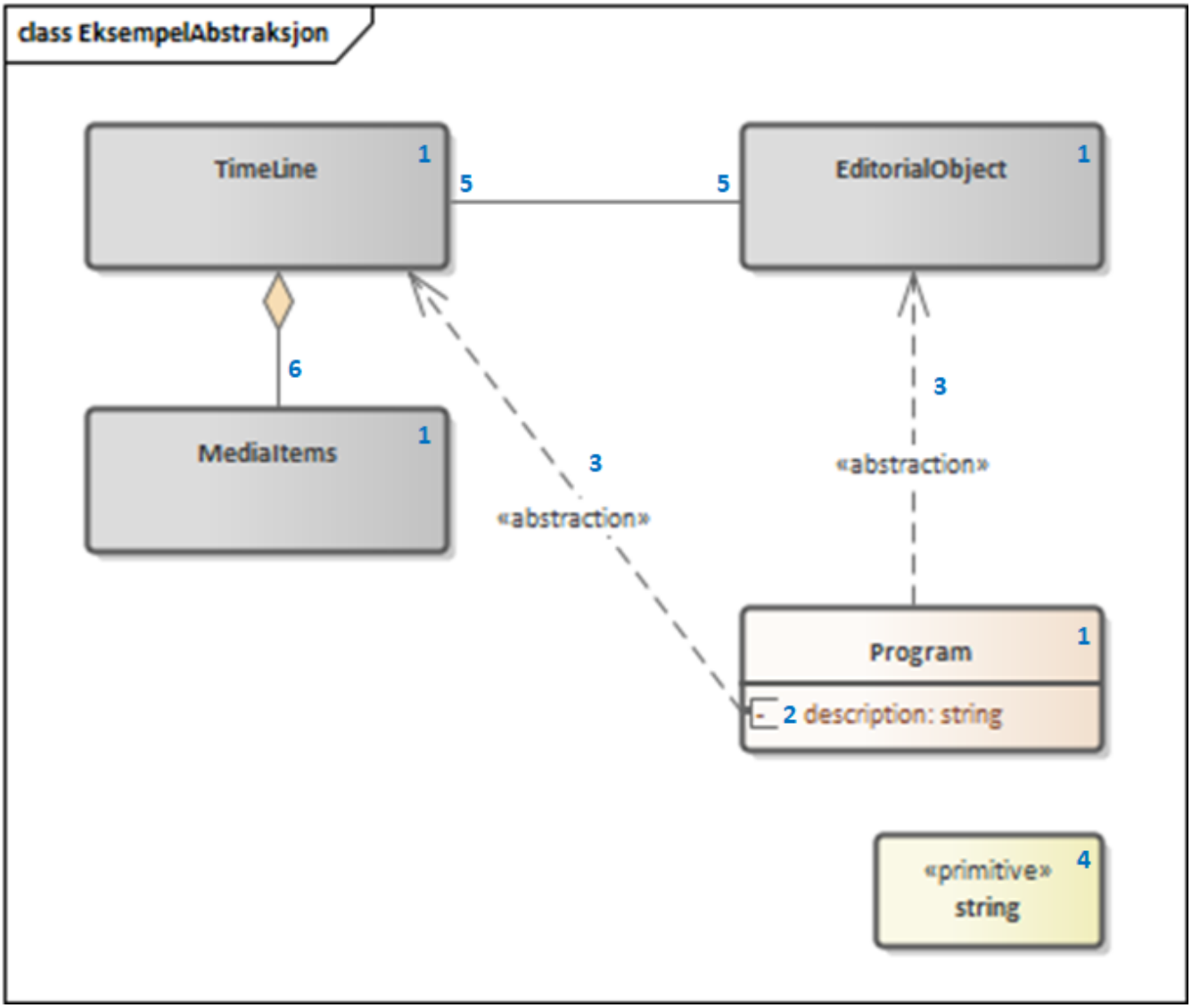
Se eksemplet i RDF Turtle.

Relasjoner

I modelleksemplet ovenfor har vi beskrevet basis relasjonstyper som spesialisering og roller. ModellDCAT-AP-NO muliggjør også beskrivelse av mer spesialiserte relasjonstyper, som abstraksjoner, assosiasjoner, komposisjoner, realiseringer og samlinger.

Abstraksjon

> Abstraksjon er en relasjon som settes mellom modellelementer eller egenskaper for å vise at disse representerer det samme konseptet, men på ulike abstraksjonsnivåer.



Figur 4. Eksempel - Abstraksjon

Eksempelet viser to former for abstraksjoner. De grå boksene i diagrammet kommer fra en standard utarbeidet av den Europeiske kringkastingsunionen (EBU). Editorial object representerer et redaksjonelt sammensatt produkt, et program eller en podcast. Timeline representerer tidsaksen i programmet og brukes til å vise hvordan de ulike Mediaitems, plater og prat, avvikles sekvensielt.

Så er det boksen Program. I diagrammet står den alene, men er nok en del av en større modell. For å vise slektskapet mellom Program og Editorial object så bruker man en abstraksjon, i form av en antydning at man her snakker om det samme, uten at man nødvendigvis har et samsvar i hvordan man har modellert med tanke på egenskaper og lignende.

Så har vi en annen abstraksjon: Program-klassen har en egenskap “description” som er en tekstlig beskrivelse av programinnholdet. Dette er jo samme type opplysning som vil framgå av konstruksjonen med Timeline og Mediaitems. Måten det er modellert på her er jo helt annerledes, og den grå modellen vil være mye mer detaljert, men “description” feltet vil fungere som en beskrivelse av det samme konseptet, dette slektskapet utrykker vi med en abstraksjon.

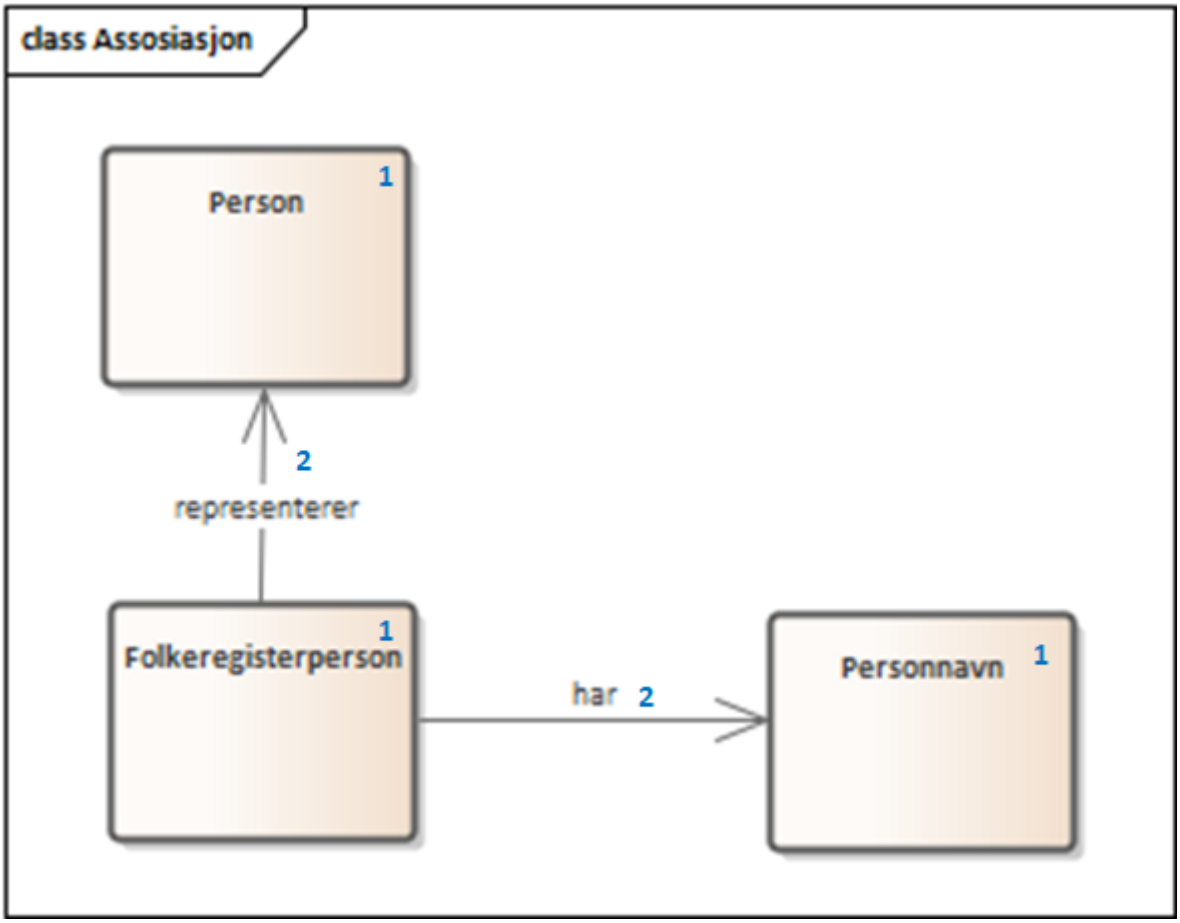
Mapping til ModellDCAT-AP-NO:

Modellelement-/egenskapstype	UML-representasjon	ModellDCAT-AP-NO
1	Class	Objekttype (`modelldcatno:ObjectType`)
2	Attribute	Attributt (`modelldcatno:Attribute`)
3	Abstraction	Abstraksjon (`modelldcatno:Abstraction`)
4	Primitive	Enkeltype (`modelldcatno:SimpleType`)
5	Role	Rolle (`modelldcatno:Role`)
6	Aggregation	Samling (`modelldcatno:Collection`)

Se eksempelet i RDF Turtle.

Assosiasjon

Assosiasjoner (`modelldcatno:Association`) beskriver enkle relasjoner mellom modellelementer, som er vanlig i mer konseptuelle modeller.



Figur 5. Eksempel - Assosiasjon

Eksempelet viser et enkelt utsnitt av begrepsmodellen for Folkeregisteret.

Mapping til ModellDCAT-AP-NO:

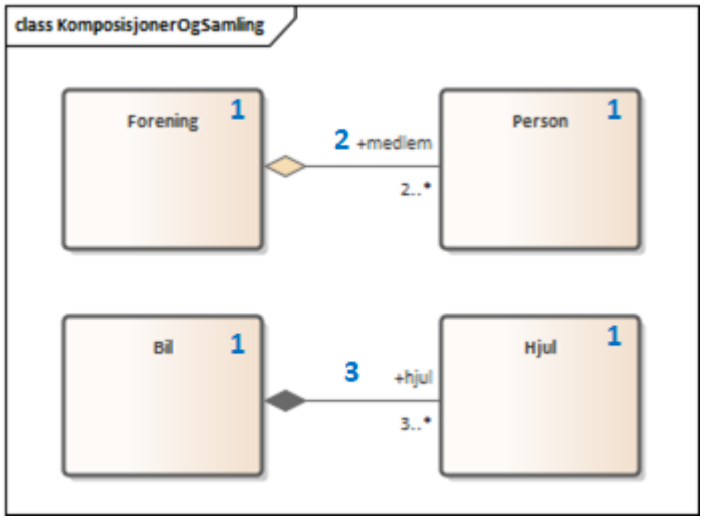
Modellelement-/egenskapstype	UML-representasjon	ModellDCAT-AP-NO
1	Class	Objekttype (`modelldcatno:ObjectType`)
2	Association	Assosiasjon (`modelldcatno:Association`)

Se eksemplet i RDF Turtle.

Samling og komposisjon

Samling: relasjon mellom to modellelementinstanser, hvor den ene instansen inngår som en del av en annen, og kan eksistere uavhengig av den andre.

Komposisjon: relasjon mellom to modellelementinstanser, hvor den ene instansen inngår som en del av en annen, og kan bare eksistere sammen med den andre.



Figur 6. Eksempel - Komposisjon og Samling

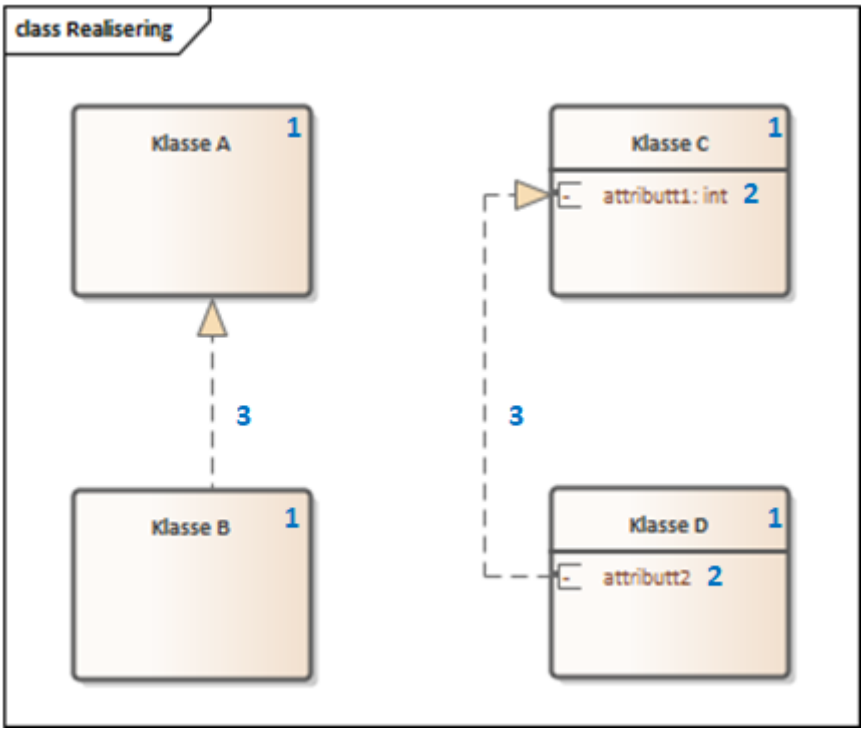
Mapping til ModellDCAT-AP-NO:

Modellelement-/egenskapstype	UML-representasjon	ModellDCAT-AP-NO
1	Class	Objekttype (`modelldcatno:ObjectType`)
2	Aggregation	Samling (`modelldcatno:Collection`)
3	Composition	Komposisjon (`modelldcatno:Composition`)

Se eksemplet i RDF Turtle.

Realisering

> Realisering er et forhold mellom modellelementer og/eller egenskaper, der det ene modellelementet/egenskapen (klienten, engelsk client) realiserer atferden som det andre modellelementet/egenskapen (leverandøren, engelsk supplier) spesifiserer. Flere klienter kan realisere atferden til en enkelt leverandør.



Figur 7. Eksempel - Realisering

Eksemplet viser hvordan man i SOSI-standardene for geografisk informasjon har definert SOSI-typer som en realisering av typer i INSPIRE. Her er ikke alle egenskapene fra standardene tatt med.

Mapping til ModellDCAT-AP-NO:

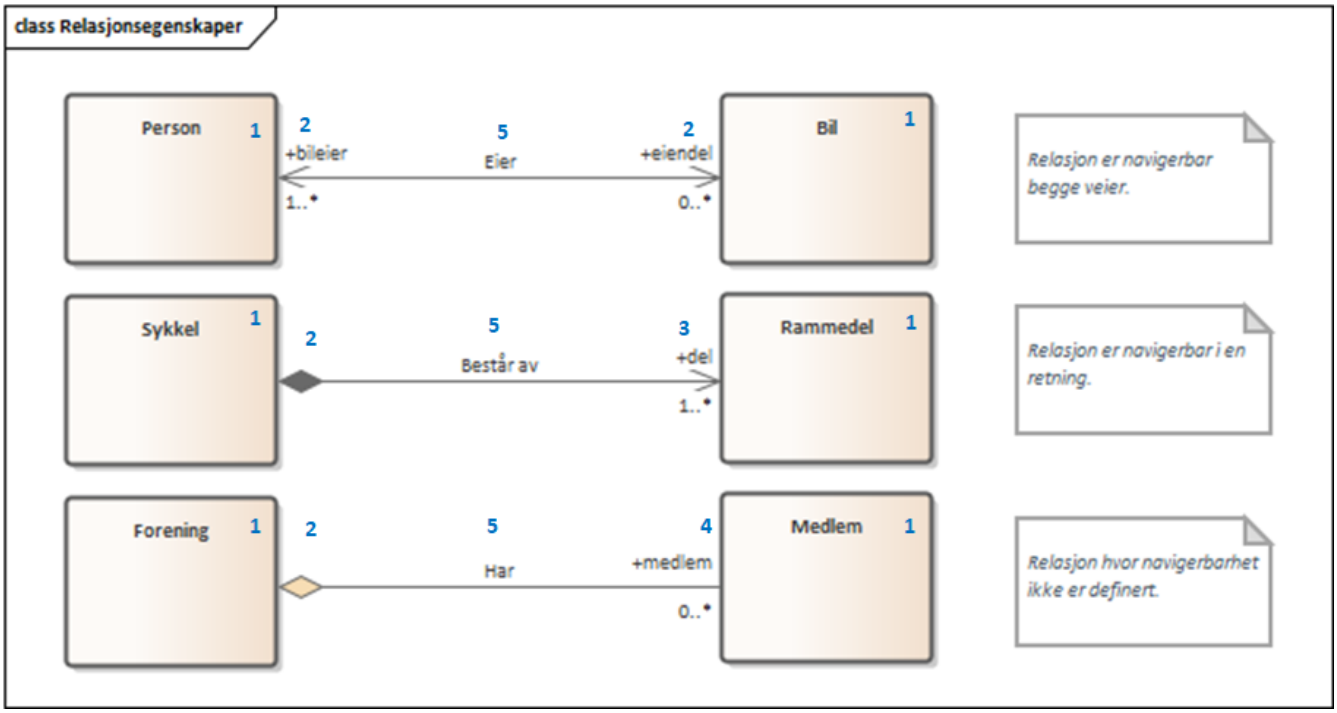
Modellelement-/egenskapstype	**UML-representasjon**	**ModellDCAT-AP-NO**
1	Class	Objekttype (`modelldcatno:ObjectType`)
2	Attribute	Attributt (`modelldcatno:Attribute`)
3	Realization	Realisering (`modelldcatno:Realization`)
4	DataType	Datatype (`modelldcatno:Datatype`)
5	Primitive	Enkeltype (`modelldcatno:SimpleType`)

Se eksemplet i RDF Turtle.

Relasjonsegenskap

> En relasjonsegenskap beskriver et symmetrisk forhold mellom to egenskaper (som f.eks. rolle, komposisjon og samling). Forholdet kan navngis.

I ModellDCAT-AP-NO er det ikke et eget modellelement som beskriver toveisrelasjoner, som f.eks. assosiasjon i UML. I stedet kan man knytte to og to egenskaper sammen, f.eks. roller, og navngi dette forholdet. Ved å spesifisere om egenskapene er navigerbar eller ikke, kan man angi leseretning på dette forholdet.



Figur 8. Eksempel - Relasjonsegenskap

Mapping til ModellDCAT-AP-NO:

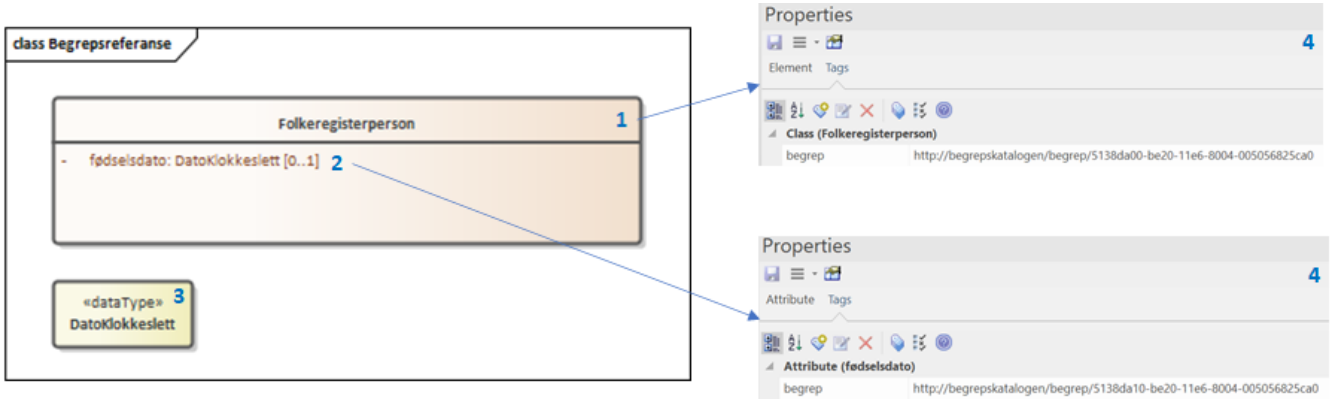
Modellelement-/egenskapstype	**UML-representasjon**	**ModellDCAT-AP-NO**
1	Class	Objekttype (`modelldcatno:ObjectType`)
2	Role	Rolle (`modelldcatno:Role`)
3	Composition + Role	Komposisjon (`modelldcatno:Composition`)
4	Aggregation + Role	Samling (`modelldcatno:Collection`)
5	Association, Composition, Aggregation	utgjør symmetrisk relasjon med (`modelldcatno:formsSymmetryWith`)

Se eksemplet i RDF Turtle.

Begrepsreferanse

En begrepsreferanse er en relasjon fra et modellelement/egenskap til et begrep, hvor begrepet det refereres til beskriver den semantiske betydningen av modellelementet/egenskapen.

For å kunne beskrive den semantiske betydningen til modellelementer, egenskaper og kodeelementer, kan disse knyttes til begreper.



Figur 9. Eksempel - Begrepsreferanse

Siden det ikke er en egen mekanisme i UML for å referere fra modellelementer til begreper, er det i eksemplet opprettet en tag, *begrep* som plassholder for begrepsreferanser. Begrepsreferansene peker her til Skatteetatenes begreper for folkeregisterperson og fødselsdato.

Mapping til ModellDCAT-AP-NO:

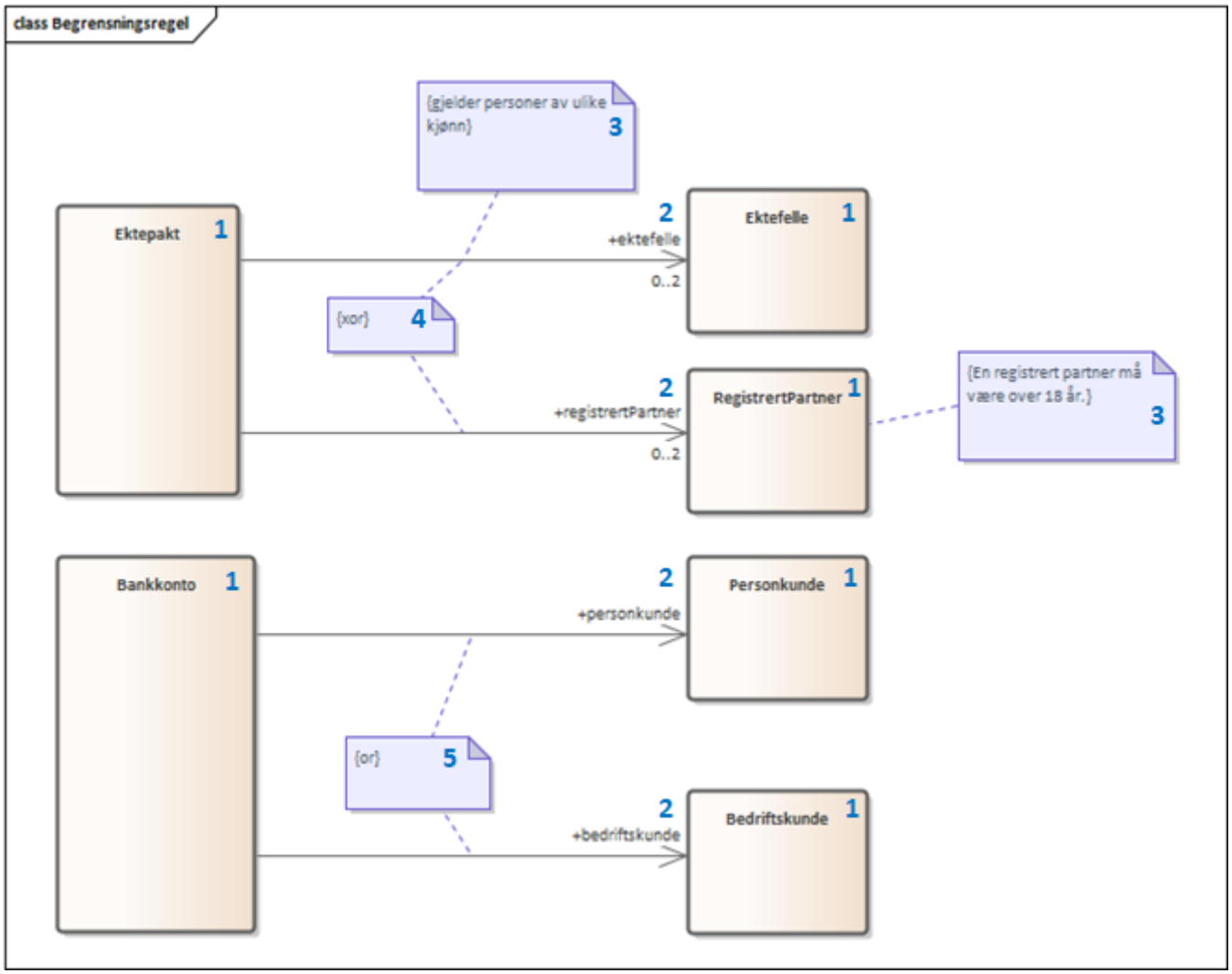
Modellelement-/egenskapstype	**UML-representasjon**	**ModellDCAT-AP-NO**
1	Class	Objekttype (`modelldcatno:ObjectType`)
2	Attribute	Attributt (`modelldcatno:Attribute`)
3	Primitive	Enkeltype (`modelldcatno:SimpleType`)
4	Tagged value	begrep (`dct:subject`)

Se eksemplet i RDF Turtle.

Begrensningsregel

En begrensningsregel beskriver hvilke begrensninger som gjelder for én eller flere egenskaper og/eller modellelementer.

ModellDCAT-AP-NO tillater å beskrive begrensninger på bruk av en eller flere modellelementer og egenskaper ved bruk av klassen Begrensningsregel (`modelldcatno:ConstraintRule`). Begrensningsuttrykk kan være en tekstlig beskrivelse, men også mer maskinelle lesbare uttrykk, som f.eks. Object Constraint Language (OCL). I tillegg er det definert to subklasser til Begrensningsregel, Enten eller (`modelldcatno:Xor`) og Eller (`modelldcatno:Or`).



Figur 10. Eksempel - Begrensningsregel

Mapping til ModellDCAT-AP-NO:

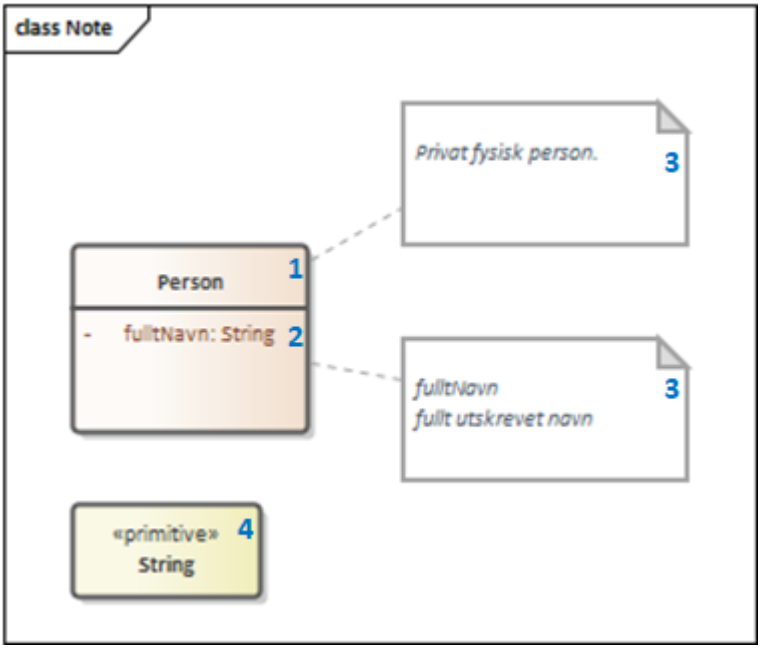
Modellelement-/egenskapstype	**UML-representasjon**	**ModellDCAT-AP-NO**
1	Class	Objekttype (`modelldcatno:ObjectType`)
2	Role	Rolle (`modelldcatno:Role`)
3	Constraint	Begrensningsregel (`modelldcatno:ConstraintRule`)
4	Xor	Enten eller (`modelldcatno:Xor`)
5	Or	Eller (`modelldcatno:Or`)

Se eksempel i RDF Turtle.

Merk at for klassen Begrensningsregel (`modelldcatno:ConstraintRule`) må minst én av egenskapene tittel (`dct:title`) eller begrensningsregel (`modelldcatno:constraintExpression`) ha en verdi. Dette gjelder imidlertid ikke for subklassene Enten eller (`modelldcatno:Xor`) og Eller (`modelldcatno:Or`), hvor kun egenskapen begrensning (`modelldcatno:constraint`) er påkrevd.

Note

En note (merkelapp) brukes til å beskrive en merknad, forklaring eller tilleggsopplysning til ett eller flere modellelementer og/eller egenskaper.



Figur 11. Eksempel - Note

Mapping til ModellDCAT-AP-NO:

Modellelement-/egenskapstype	**UML-representasjon**	**ModellDCAT-AP-NO**
1	Class	Objekttype (`modelldcatno:ObjectType`)
2	Attribute	Attributt (`modelldcatno:Attribute`)
3	Note	Note (`modelldcatno:Note`)
4	Primitive	Enkeltype (`modelldcatno:SimpleType`)

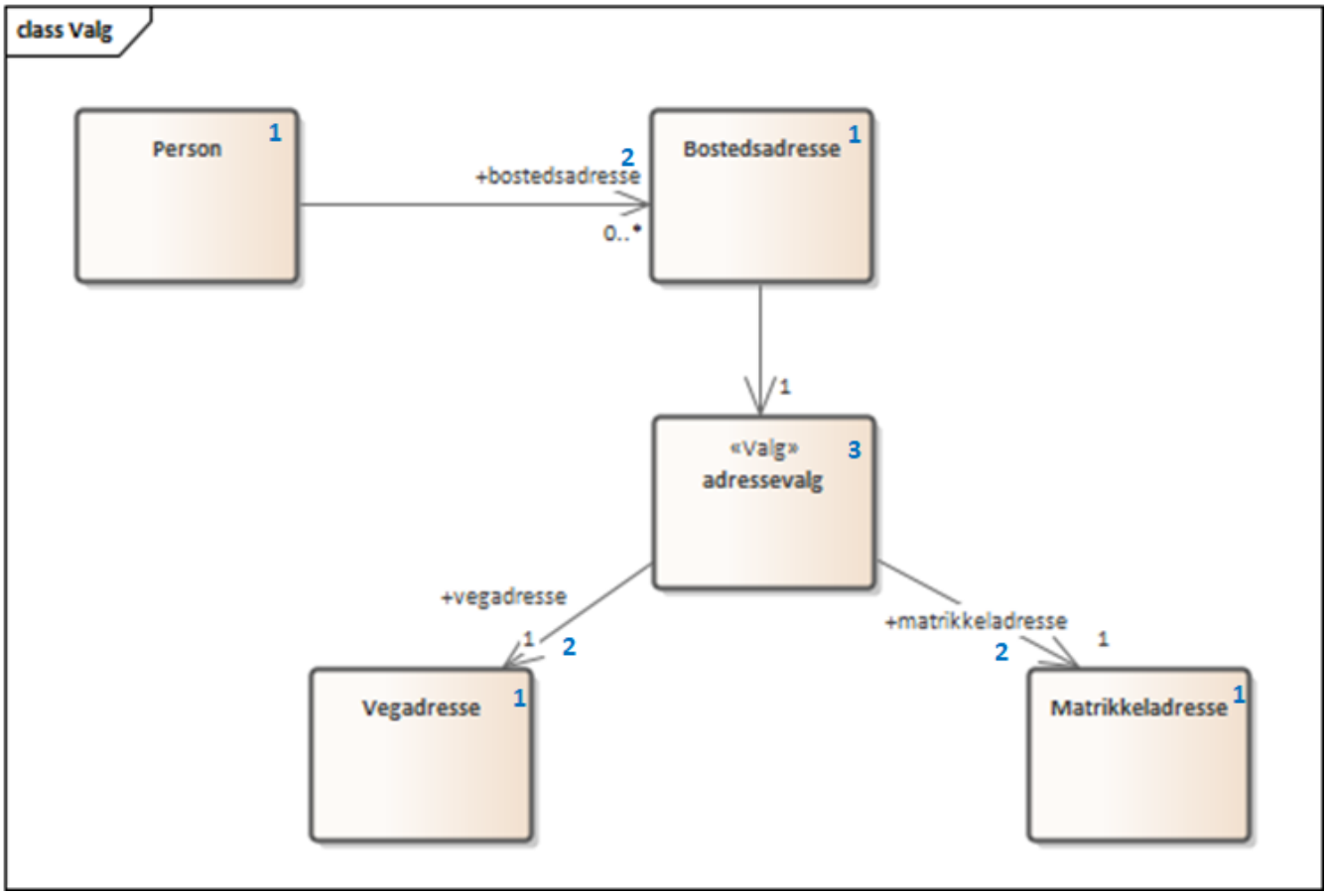
Se eksempel i RDF Turtle.

Valg (Choice)

> Valg er en egenskap som tillater at én egenskap eller modellelement av et sett av valgbare egenskaper og/eller modellelementer, kan inngå i det bærende modellelementet.

Nedenfor vises to eksempler på bruk av Valg (Choice), hvor det første tar for seg enkelvalg (single choice) og det andre flervalg (multiple choice). Valg (Choice) er et konsept som finnes bl.a. i XML Schema Definition (XSD). I eksemplene har vi benyttet UML-modeller. Siden Valg ikke er et eget element i UML klassediagram, har vi framstilt det som en klasse med stereotype «Valg». Selve valgene har vi også representert som XSD Choice.

Enkelvalg



Figur 12. Eksempel - Enkelvalg

Eksemplet viser at Person kan ha null til mange bostedsadresser. Valget *adressevalg* beskriver at objekttypen Bostedsadresse kan enten ha rollen *vegadresse* eller *matrikkeladresse*. At dette er et enkelvalg (simple choice), er beskrevet ved at UML assosiasjonen mellom Bostedsadresse og adressevalg har multiplisitet 1.

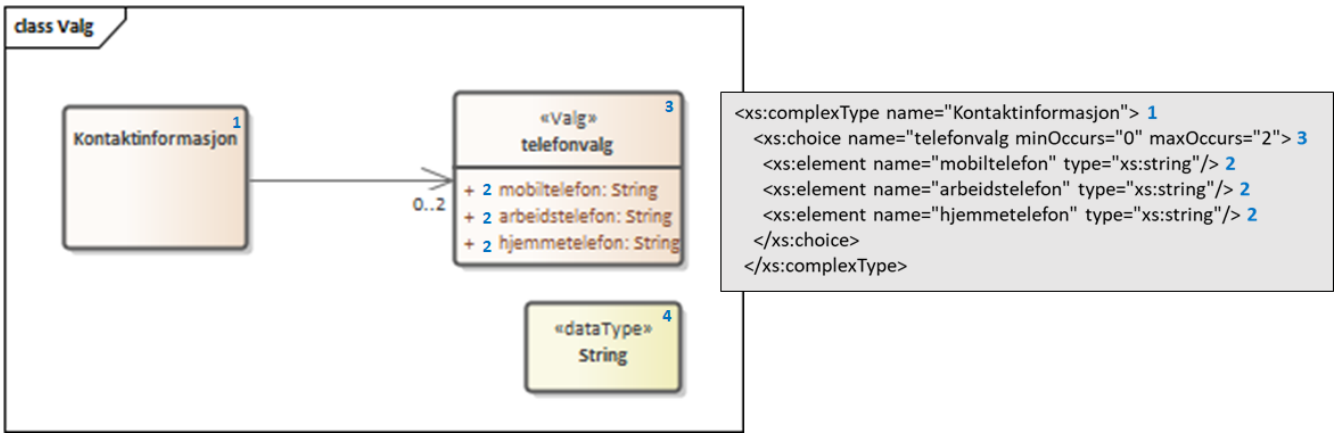
Mapping til ModellDCAT-AP-NO:

Modellelement-/egenskapstype	**UML-representasjon**	**ModellDCAT-AP-NO**
1	Class	Objekttype (`modelldcatno:ObjectType`)
2	Role	Rolle (`modelldcatno:Role`)

3	Class, stereotype «Valg»	Valg (`modelldcatno:Choice`)

Se eksempel i RDF Turtle.

Flervalg



Figur 13. Eksempel - Flervalg

Flervalg (multiple choice) brukes når et unikt valg kan foretas flere ganger. Kontaktinformasjon er i eksempelet knyttet til et Valg, *telefonvalg*, som representer et valg mellom ulike typer telefonnumre - mobiltelefon, arbeidstelefon og hjemmetelefon. Multiplisiteten 0..2 på assosiasjonsenden mellom Kontaktinformasjon og telefonvalg, beskriver at det kan forekomme opp til to unike valg. Det betyr at *Kontaktinformasjon* kan maksimalt bestå av to telefonnumre, som er av type mobiltelefon, arbeidstelefon og/eller hjemmetelefon. I XSD-representasjonen er dette angitt ved at xsd-elementet choice er tildelt verdier for minOccurs og maxOccurs.

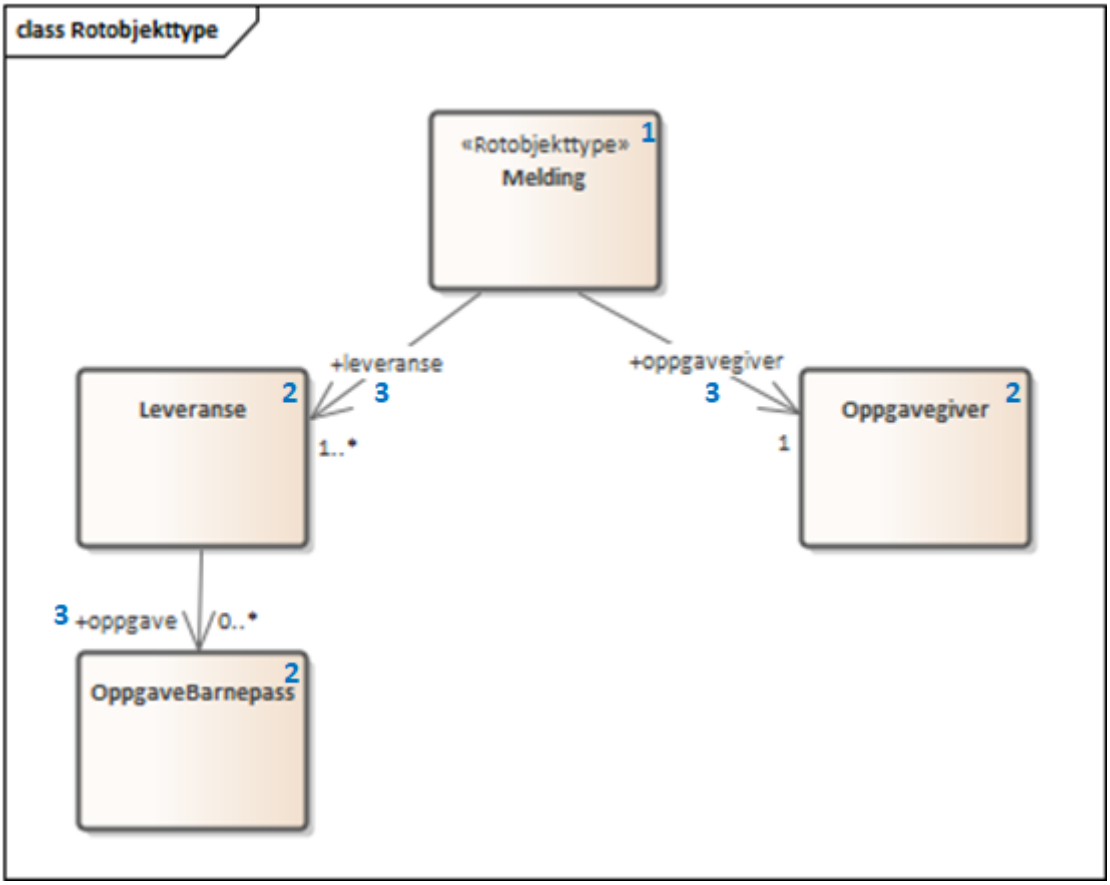
Mapping til ModellDCAT-AP-NO:

Modellelement-/egenskapstype	**UML-representasjon**	**ModellDCAT-AP-NO**
1	Class	Objekttype (`modelldcatno:ObjectType`)
2	Attribute	Attributt (`modelldcatno:Attribute`)
3	Class, stereotype «Valg»	Valg (`modelldcatno:Choice`)
4	Primitive	Enkeltype (`modelldcatno:SimpleType`)

Se eksempel i RDF Turtle.

Rotobjekttype

> En rotobjekttype representerer det overordnede objektet i en gruppe av objekter som er knyttet til hverandre i en hierarkisk struktur.



Figur 14. Eksmepel - Rotobjekttype

Eksempelet viser bruk av rotobjekttype. Rotobjektet er her representert som en klasse med stereotype «Rotobjekttype».

Mapping til ModellDCAT-AP-NO:

Modellelement-/egenskapstype	**UML-representasjon**	**ModellDCAT-AP-NO**
1	Class, stereotype «Rotobjekttype»	Rotobjekttype (`modelldcatno:RootObjectType`)
2	Class	Objekttype (`modelldcatno:ObjectType`)
3	Role	Rolle (`modelldcatno:Role`)

Se eksempel i RDF Turtle.

Mer om enkeltyper

Verdirestriksjon

I ModellDCAT-AP-N0 kan enkeltyper (modelldcatno:SimpleType) ha verdirestriksjoner. Til dette benyttes et utvalg av XML sine tegn- og tallrestriksjoner.

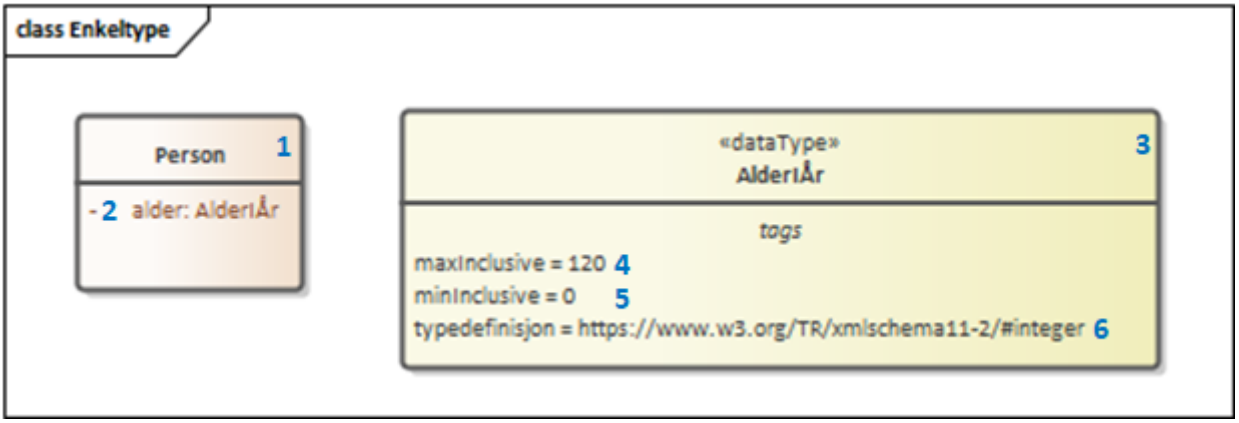
- xsd:fractionDigits

- xsd:length
- xsd:maxExclusive
- xsd:maxInclusive
- xsd:maxLength
- xsd:minExclusive
- xsd:minInclusive
- xsd:minLength
- xsd:pattern
- xsd:totalDigits

Typedefinisjoner

I ulike modeller kan det benyttes ulike standard ontologier eller bibliotek for primitive datatyper (enkeltyper), f.eks. typesett definert for XSD eller UML.

Ved bruk av egenskapen typedefinisjon (modelldcatno:typeDefinitionReference), kan man referere til ontologien eller biblioteket hvor datatypene er definert i form av en URI.



Figur 15. Eksempel - Typedefinisjon

Mapping til ModellDCAT-AP-NO:

Modellelement-/egenskapstype	**UML-representasjon**	**ModellDCAT-AP-NO**
1	Class	Objekttype (`modelldcatno:ObjectType`)
2	Attribute	Attributt (`modelldcatno:Attribute`)
3	Primitive	Enkeltype (`modelldcatno:SimpleType`)
4	Tagged value	maksimum inklusivt (`xsd:maxInclusive`)

5	Tagged value	minimum inklusivt (`xsd:minInclusive`)
6	Tagged value	typedefinisjon (`modelldcatno:typeDefinitionReference`)

Se eksempel i RDF Turtle.

Mer om moduler

Stereotyper

ModellDCAT-AP-NO har ikke et eget element for stereotyper slik man har i UML. Stereotype kan ses som en type gruppering, og moduler kan derfor brukes hvis man har behov for å representere dette.

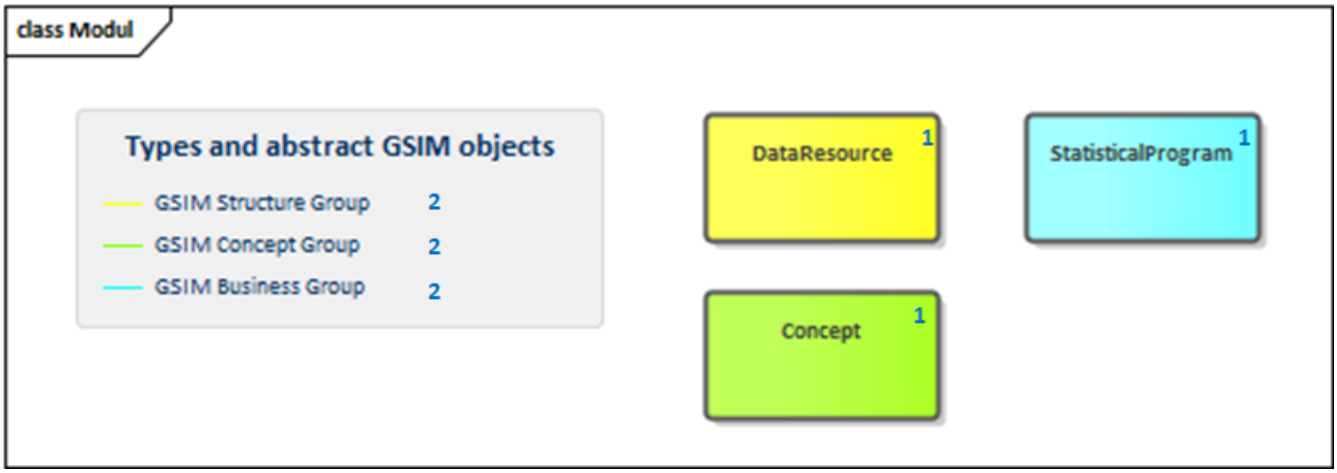
Figur 16. Eksempel - Modul (stereotyper)

Mapping til ModellDCAT-AP-NO:

Modellelement-/egenskapstype	**UML-representasjon**	**ModellDCAT-AP-NO**
1	Class	Objekttype (`modelldcatno:ObjectType`)
2	Stereotype	Modul (`modelldcatno:Module`)

Bruk av farger i diagrammer

Ofte grupperes modellelementer ved at de får ulike farger i diagrammer. I ModellDCAT-AP-NO kan dette representeres ved å bruke moduler.



Figur 17. Eksempel - Modul (farger i diagrammer)

Eksemplet er hentet fra SSBs logiske datamodell for statistikkinformasjon, som er basert på UNECE standard Generic Statistical Information Model (GSIM). Egenskaper på objekttypene vises ikke.

Mapping til ModellDCAT-AP-NO:

Modellelement-/egenskapstype	**UML-representasjon**	**ModellDCAT-AP-NO**
1	Class	Objekttype (`modelldcatno:ObjectType`)
2	Legend element	Modul (`modelldcatno:Module`)

Se eksempel i RDF Turtle.

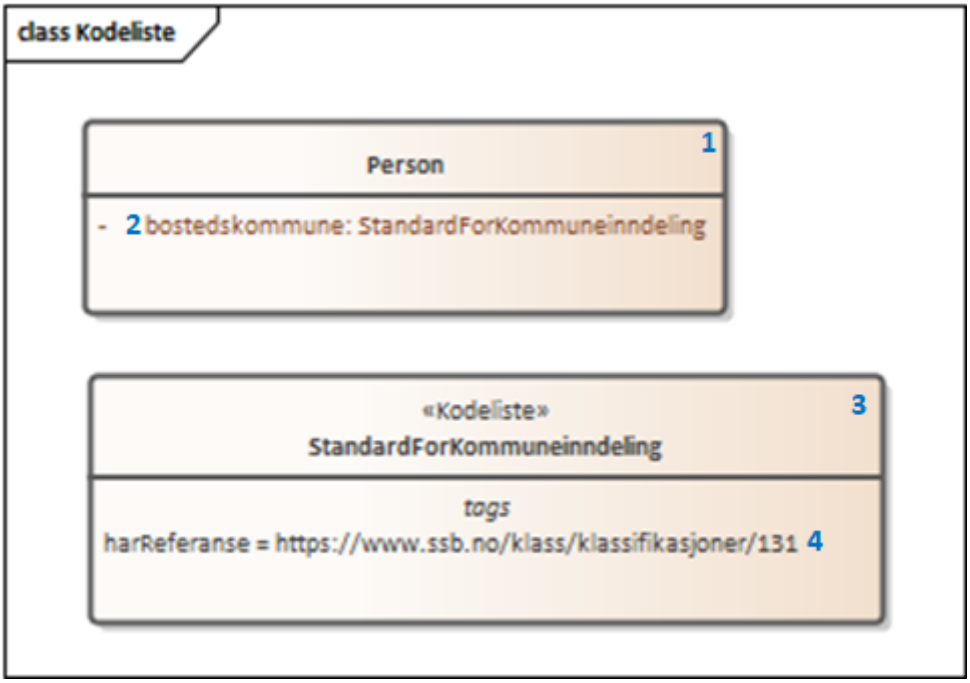
Mer om kodelister

Under Basis modellelementer og egenskaper har vi laget et eksempel på hvordan man kan beskrive en enkel enumerasjon som en kodeliste (`modelldcatno:CodeList`), og hvordan vi med egenskapen «har verdi fra» (`modelldcatno:hasValueFrom`) kan angi at et attributt (`modelldcatno:Attribute`) relaterer seg til en kodeliste.

Andre typiske brukstilfeller er:

- Kodeliste brukt i en informasjonsmodell, hvor kodeelementer ikke er beskrevet, men med referanse til en ekstern beskrivelse av kodelisten eller kodeverket med tilhørende kodeelementer.
- Kodeliste brukt i informasjonsmodell, med koder og kodetekst
- Kodeliste som egen informasjonsmodell
- Kodeliste som et datasett (åpne data)

Ekstern kodeliste



Figur 18. Eksempel - Ekstern kodeliste

I eksemplet er SSBs kodeliste «StandardForKommuneinndeling» benyttet for å beskrive verdidomenet til attributtet bostedskommune. I stedet for å legge inn alle kommunene som kodeelementer i modellen, henviser man til SSBs nettsider hvor beskrivelsen av kodelistene og kodeelementene ligger.

Mapping til ModellDCAT-AP-NO:

Modellelement-/egenskapstype	**UML-representasjon**	**ModellDCAT-AP-NO**
1	Class	Objekttype (`modelldcatno:ObjectType`)
2	Attribute	Attributt (`modelldcatno:Attribute`)
3	Class, stereotype «Kodeliste»	Kodeliste (`modelldcatno:CodeList`)
4	Tagged value	har referanse (`rdfs:seeAlso`)

Se eksempel i RDF Turtle.

Kodeliste med koder og kodetekst

Enumerasjoner representerer lister med kodeverdier. I ModellDCAT-AP-NO er det mulig å gi mer utdypende beskrivelser av kodeelementene i en kodeliste, som kodetekst, inklusjons-/eksklusjonsmerknader, frarådet kodetekst, definisjon m.m.

Se eksempel på en kodeliste i RDF Turtle.

Kodeliste som informasjonsmodell

I ModellDCAT-AP-NO er det mulig å beskrive en eller flere uavhengige kodelister i en egen informasjonsmodell. Ofte er det slik at man gjenbruker de samme kodelistene i ulike modeller. I stedet for å beskrive den samme kodelisten flere ganger, kan man beskrive den én gang i en egen informasjonsmodell. Dermed kan man referere til kodelisten fra modellene hvor den er benyttet.

Se eksempel på en kodeliste som en egen informasjonsmodell, i RDF Turtle.

Kodeliste som et datasett

En kodeliste kan ses på som en samling av data, og kan dermed beskrives som et datasett i henhold til DCAT-AP-NO.

For å angi at datasettet er en kodeliste, brukes `dct:type`:

```
<https://examples.com/infomoc/exdataset> a dcat:Dataset ;
   dct:type <http://publications.europa.eu/resource/authority/dataset-type/CODE_LIST>
.
```

Hvis kodelisten i tillegg er beskrevet som en informasjonsmodell i ModellDCAT-AP-NO, kan denne ses på som en distribusjon til datasettet. Du knytter da datasettet og informasjonsmodellen sammen ved bruk av `dcat:distribution`:

```
<https://examples.com/infomoc/exdataset> a dcat:Dataset ;
   dcat:distribution [ a dcat:Distribution ; dcat:accessURL
<https://github.com/Informasjonsforvaltning/modelldcat-ap-no/examples/testMod1> ] .
```

Modellkatalog

Hva bruker du en modellkatalog til?

Informasjonsmodeller fra en eller flere virksomheter samles i en modellkatalog. En modellkatalog kan ses som et omslag til eller innbinding av modellene. En modellkatalog skal ha én utgiver, og en og samme virksomhet kan være utgiver av flere modellkataloger.

Hvordan beskriver du en modellkatalog?

Som minimum skal følgende egenskaper ha verdier:

- beskrivelse (`dct:description`): En kort og presis beskrivelse av modellkatalogen skal gjøre det lett for andre å se hva det inneholder. Beskrivelse er et obligatorisk felt. Bør gjentas når beskrivelsen finnes i flere ulike språk/målformer.
- identifikator (`dct:identifier`): Obligatorisk (se Om … «identifikator»).
- tittel (`dct:title`): Kortfattet om katalogen. Angi, uten å liste, hvilke modeller den omfatter, f.eks. «Informasjonsmodellene til Digitaliseringsdirektoratet». Bør gjentas når tittelen finnes i flere ulike språk/målformer.
- modell (`modelldcatno:model`): Referanser til en eller flere informasjonsmodeller som er inkludert i modellkatalogen.
    - Kommentar: I UML-diagrammet i ModellDCAT-AP-NO står det at `dct:hasPart` er obligatorisk mens `modelldcatno:model` er anbefalt. `dcat:Resource` som `dct:hasPart` peker på er en abstrakt klasse og skal erstattes med instanser av subklassen `modelldcatno:InformationModel` eller `dcat:Catalog`. Bortsett fra tilfeller der man beskriver en modellkatalog bestående av andre modellkataloger, skal en modellkatalog ha minst én informasjonsmodell.
- utgiver (`dct:publisher`): Se Om … «utgiver».

Eksempel på en modellkatalog (i RDF Turtle)

Se eksemplet i RDF Turtle.

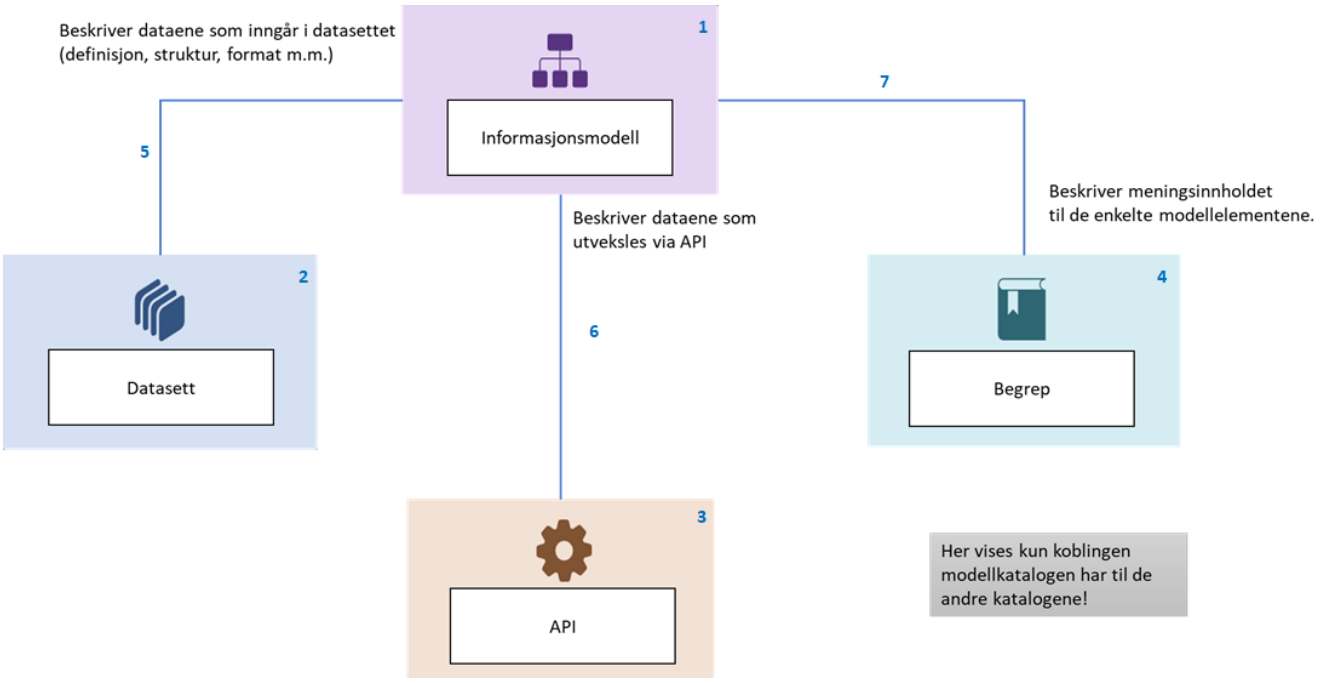
Hva gjør du når du skal publisere modellkatalogen din i Felles datakatalog?

Når du skal publisere katalogen din i Felles datakatalog, må modellene mappes iht. ModellDCAT-AP-NO til et format som kan høstes. Pr. i dag støtter Felles datakatalog følgende RDF-serialiseringer: RDF/XML, Turtle og JSON-LD.

Modellkatalogen må da legges ut som en nettressurs, f.eks. i Github. URIen til nettressursen registreres i Felles datakatalog, og katalogen høstes så automatisk.

Informasjonsmodell og relasjoner til datasett, datatjenester og begreper

Figuren nedenfor viser sammenhengen mellom informasjonsmodell, datasett, API (datatjeneste) og begreper:



Figur 19. Sammenheng mellom informasjonsmodell, datasett, datatjenester og begreper

Nummerering	ModellDCAT-AP-NO
1	Informasjonsmodell (`dct:InformationModel`)
2	Datasett (`dcat:Dataset`)
3	Datatjeneste (`dcat:DataService`)
4	Begrep (`skos:Concept`)
5	i samsvar med (`dct:conformsTo`)
6	i samsvar med (`dct:conformsTo`)
7	begrep (`dct:subject`)

Som vi har sett, kan vi referere fra informasjonsmodeller, modellelementer, egenskaper eller kodeelementer til begreper ved bruk av egenskapen begrep (`dct:subject`). Se beskrivelser og eksempler under Informasjonsmodell og Begrepsreferanser.

Når det gjelder et datasett eller datatjeneste, refererer man fra disse til en informasjonsmodell. I ModellDCAT-AP-NO er klassen Informasjonsmodell subklasse av `dct:Standard`. Ved å bruke egenskapen `dct:conformsTo`, kan et datasett eller en datatjeneste referere til en informasjonsmodell.

Eksempler i RDF Turtle:

Datasett til informasjonsmodell:

```
<https://examples.com/infomoc/exdataset> a dcat:Dataset ;
   dct:conformsTo <https://github.com/Informasjonsforvaltning/modelldcat-ap-
no/examples/testMod1> .
```

Datatjeneste til informasjonsmodell:

```
<https://examples.com/infomoc/exdataservice> a dcat:Dataservice ;
   dct:conformsTo <https://github.com/Informasjonsforvaltning/modelldcat-ap-
no/examples/testMod2> .
```

Referanser

- DCAT-AP-NO: Standard for beskrivelse av datasett, datatjenester og datakataloger (DCAT-AP-NO)
- Github-repository for ModellDCAT-AP-NO, https://github.com/Informasjonsforvaltning/modelldcat-ap-no
- ModellDCAT-AP-NO: Spesifikasjon for beskrivelse av informasjonsmodeller (ModellDCAT-AP-NO)
- RDF Turtle
- Resource Description Framework (RDF)
- SKOS-AP-NO: Forvaltningsstandard for tilgjengeliggjøring av begrepsbeskrivelser basert på SKOS
- Standard for pekere til offentlige ressurser på nett

Hjelpemidler

- Bibliotek for å mappe en modelldcatno-modell til RDF, Modelldcatnotordf. Kildekode: https://github.com/Informasjonsforvaltning/modelldcatnotordf.
- Valideringsregler (shacl), https://github.com/Informasjonsforvaltning/modelldcat-ap-no/blob/develop/shacl/modelldcat-ap-no_v1_SHACL-shapes.ttl
- Validator: https://data.norge.no/validator
- IDLab Turtle Validator, http://ttl.summerofcode.be

Akronymer

AP – Application Profile

DCAT – Data Catalog

FDK – Felles datakatalog

JSON – JavaScript Object Notation

JSON – LD – JSON for Linked Data

NO – Norwegian

RDF – Resource Description Framework

SKOS – Simple Knowledge Organisation System

URI – Uniform Resource identifier

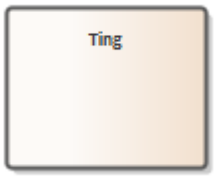
XKOS – eXtended Knowledge Organisation System

XML – eXtended Markup Language

Vedlegg - Informasjonsmodellering enkelt forklart

Modellering – et enkelt utgangspunkt

Vi lager modeller for å strukturere kunnskap. Vi deler opp informasjonen i tenkte enheter: ting. Disse tingene omtales litt forskjellig avhengig av tradisjon og sammenheng, noen sier klasser, noen sier konsepter og i ModellDCAT-AP-NO snakker vi om modellelementer. Uavhengig av hva vi omtaler det som, er klassen eller modellelementet en måte å samle informasjon om et konsept, noe konkret eller abstrakt.



Figur 20. Et konsept

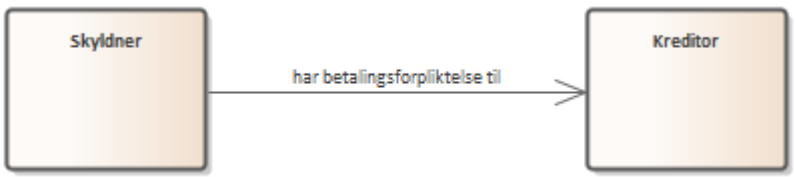
Denne informasjonen som beskriver «tingen» kan vi kalle egenskaper. Alt vi kan utrykke for å beskrive «tingen» er egenskaper, enten det er enkle fakta som en farge, *Enkeltype*, eller en struktur av opplysninger slik som en adresse, *Datatype*, som har flere felter, slik som gatenavn, husnummer, postnummer og poststed.

Figur 21. En datatype

En modell består som regel av mer enn ett konsept. Relasjonen mellom konsepter er i denne sammenhengen også en egenskap.

Hvilken type egenskap er litt avhengig av hva vi fokuserer på i modellen.

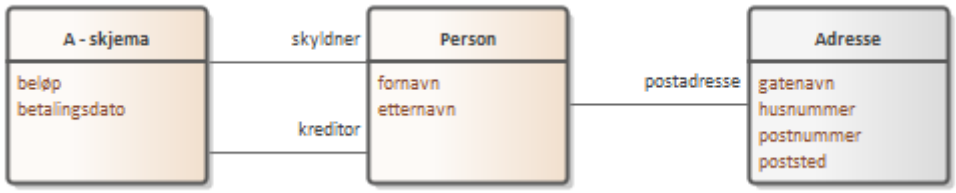
I en konseptuell modell der man f.eks. setter begreper i en sammenheng vil en kanskje kalle dette forholdet for en *Assosiasjon*.



Figur 22. En relasjon

Her kan man si at en skyldner kjennetegnes av at man som en av egenskapene har en betalingsforpliktelse til en kreditor.

I modeller som beskriver konkrete forhold, logiske eller fysiske modeller, er det vanlig å beskrive relasjonen mellom to objekter utfra hva det ene objektet betyr for det andre. Rettere sagt, hvilken rolle spiller B for A.



Figur 23. Modellelementer med relasjoner og roller

Vi kan tenke oss et skjema (A) der man i tillegg til enkle egenskaper som beløp og datoer ønsker å beskrive en person med navn og adresse. Person er her et selvstendig konsept eller en klasse som blir brukt av skjemaet, i tillegg er det hektet på adresseinformasjon som en egen klasse.

Skjemaet som skal beskrives, inneholder informasjon om en som skylder penger, derfor kan man tenke seg at dette beskrives som skjema (A), med to egenskaper som relaterer seg til en person, i det ene tilfelle skyldner, i det andre tilfelle kreditor. Dette er det samme forholdet som var en assosiasjon i den konseptuelle modellen, men som utrykkes som et rolle-forhold i den logiske eller fysiske modellen.

Ved hjelp av konseptet Modellelement, egenskapenes rolle, der ulike Modellelementer brukes for å beskrive ulike deler av informasjonen; og enkeltype og datatype for data om konseptet, kan man utrykke en datastruktur.

ModellDCAT-AP-NO har et større mulighetsrom, noe du kan se tidligere i denne veilederen, men som en enkel start er disse fire elementene nok til å beskrive datastrukturer.

Metode

Ulike arkitekter vil ha ulike svar på hva de gjør når de modellerer, men her er en metode som virker.

Finn interesseområdet og omfanget av modellen

Det er sagt at alt henger sammen med alt, men skal man lage en modell er det viktig å klargjøre akkurat hva modellen skal gi en oversikt og innsikt i, og hvem som skal bruke modellen.

Finn mulige gjenbrukbare modeller

Ikke gjør dobbeltarbeid, hvis du kan bruke ting du eller andre allerede har definert betyr det ikke bare mindre arbeid for deg som arkitekt, det betyr også mindre arbeid for de som skal omsette den konkrete modellen til en applikasjon. Gjenbruk av de samme basiskonseptene letter den gjennomgående forståelsen og sikrer at man beskriver de samme tingene på en ensartet måte.

Finn de viktige objektstypene

Også kalt klassene, tingene eller konseptene. Pek ut hva som er viktigst i modellen, og start med disse og bygg ut derifra.

De viktige konseptene kjennetegnes ofte ved at ord som betegner konseptet går igjen i beskrivelsen av systemet. Les igjennom sentrale dokumenter som beskriver feltet og vurder viktigheten av alle substantiver.

I en konseptuell modell vil ofte de objektene som har flest tilknyttede relasjoner være de viktigste.

La en gruppe mennesker som kjenner område skrive ned på gule lapper hva de forbinder med området som skal beskrives med ett ord. De sentrale objektene er ofte de som beskrives av flest lapper.

Finne sammenhengen mellom generelle og spesielle objekttyper

Noen konsepter er slik at det er en utdyping av et annet: loff er en type brød, da kan man si at brød er generelt og at loff er en spesialisering av brød. Med andre ord, vurder om en objekttype er en spesialisering eller en generalisering av en annen objekttype.

Definer egenskapene til objekttypene og beskriv tillatte verdier

Det er her informasjonen kommer inn i modellen. For hver objektstype, finn de egenskapene som beskriver objektet nærmere og legg dem til i form av egenskaper. Identifiser om egenskapen utrykkes på en spesiell måte, er det et heltall, en streng, og har det noen spesifiserte minimum og maksimum verdier. F.eks. at en gitt egenskap er et heltall mellom 10 og 50.

Beskriv forhold mellom objekttypene

Her beskrives forholdene mellom objektene i modellen. Avhengig av hvilken type modell det er kan man beskrive dette som en assosiasjon, en rolle eller noen av de andre relasjonstypene som er beskrevet senere i dette dokumentet. Forholdet mellom objekttypene i modellen gir struktur.

Beskriv verdidomener i form av kodelister

Dersom man vil beskrive en egenskap som noe mer spesifikt enn at egenskapen har et gitt format, men ønsker at feltet skal inneholde en konkret verdi fra en liste, da definerer man en kodeliste. Slike kodelister kan beskrive valgmuligheter som har en lokal gyldighet i en gitt modell, eller være større lister som brukes på tvers av ulike modeller. Dersom listene er store og har et globalt anvendelsesområde, bør de publiseres som egne dokumenter og kun bli referert til fra modellen. Små lokale kodelister kan defineres som en del av modellen.