

Interoperabla specifikationer

En utredning inom byggblock metadata

Matthias Palmér

(metadataexpert för Digg)

Ulrika Domellöf Mattsson

(byggblocksansvarig)

Deltagare i referensgruppen presenterar sig kort

Var och en tar 1-2 minuter att säga något om:

- Vem man är och vilken organisation man representerar
 - Tidigare erfarenhet av frågan om beständiga identifierare
 - Förväntan på deltagandet / projektet
 - Snabba förslag / tankar redan nu
- (kom ihåg 1-2 minuter)

Byggblock metadata

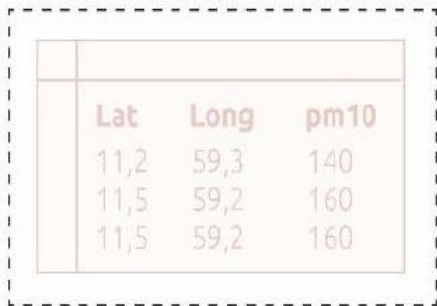
Myndigheten för digital förvaltning, Digg leder arbetet med att etablera en förvaltningsgemensam digital infrastruktur (Ena) för att information ska kunna utbytas på ett säkert och effektivt sätt.

- Samverkan
- Nationella grunddata (ramverk)
- Kompetensområden
- Förutsättningsskapande byggblock

<https://www.digg.se/ledning-och-samordning/ena---sveriges-digitala-infrastruktur>

Beskrivande metadata (Fokus på datamängden)

Titel: Luftkvalitet
Utgivare: Naturvårdsverket
Uppdat: Dagligen

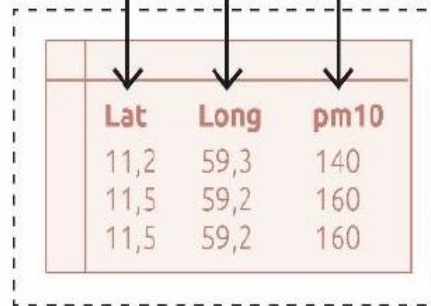


A dashed box containing a table with three columns: Lat, Long, and pm10. The table contains three rows of data. An arrow points from the descriptive metadata box above to the top of this table.

Lat	Long	pm10
11,2	59,3	140
11,5	59,2	160
11,5	59,2	160

Strukturella metadata (Fokus på datauttrycket)

Kolumn: lat
long
pm10
Label: latitud
longitud
partiklar
Datatyp: decimaltal
decimaltal
heltal



A dashed box containing a table with three columns: Lat, Long, and pm10. The table contains three rows of data. Three arrows point from the structural metadata box above to the column headers of this table: 'lat' to 'Lat', 'long' to 'Long', and 'pm10' to 'pm10'.

Lat	Long	pm10
11,2	59,3	140
11,5	59,2	160
11,5	59,2	160

Lite förenklat motsvarar beskrivande metadata FAIR principerna Findable och Accessible medan strukturella metadata motsvarar Interoperable och Reusable.

Syfte med referensgruppen

Syftet med referensgruppen är att arbeta fram en profil för att främja interoperabilitet och återanvändning av delar i en specifikation.

En specifikation kan innehålla en blandning av bakgrund, motivering och mer formella beskrivningar. Det är ett paket med olika delar/resurser, vissa riktade mot mänsklig konsumtion, andra maskinläsbara.

Genom att ge användare möjlighet att söka djupare i en datamängd, på den bakomliggande olika delarna möjliggörs interoperabilitet och återanvändning av dem.

Tidigare förstudie: <https://github.com/diggsweden/information-models-investigation>

Referensgruppsarbete

Material:

- Allt läggs up på Github
<https://github.com/diggsweden/interoperable-specifications>
- Bakgrundsmaterial, specifikation, exempel etc.
- Material på engelska då vi förutser Nordisk / Europeisk samverkan

Process:

- Ärendehantering på Github - mallar för styra kommunikationen
- Ändringsförslag (PR = Pull Requests)
- Referensgrupp cirka 4 möten - deltagares namn visas på github
- Första referensgruppsmötet måndag den 18/11 kl. 10

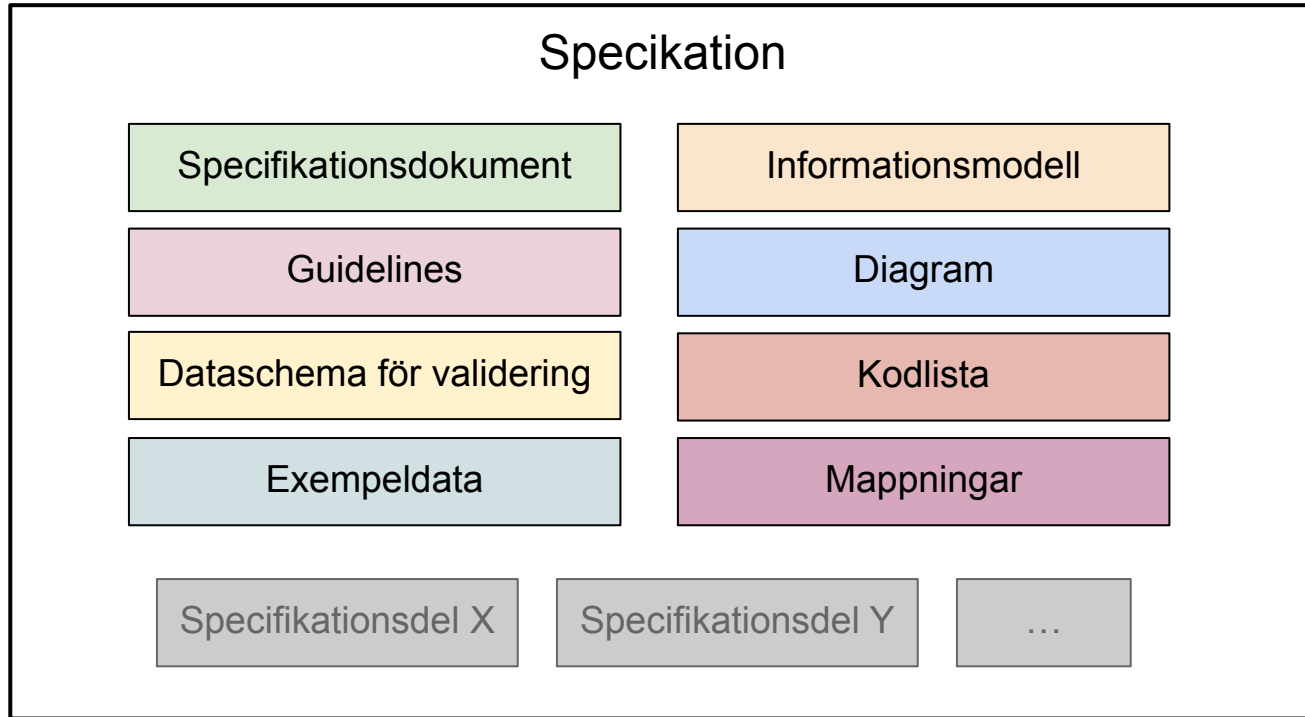
Bakgrund / behov

Syftet med interoperabla specifikationer är att stödja återanvändning mellan specifikationer

Förhoppningen är att:

- **Förenkla / effektivisera** hur man skapar nya informationsmodeller då man inte behöver börja från början
- **Förbättra kvaliteten** då återanvända delar redan är genomtänka och testade
- **Minska tiden för att lära sig** nya specifikationer då det finns mer överlapp
- Data som följer olika specifikationer kan vara **partiellt interoperabla**
- **En community** skapas kring interoperabilitet / återanvändning
- **Erfarenheter och kompetens** mellan organisationer kan delas

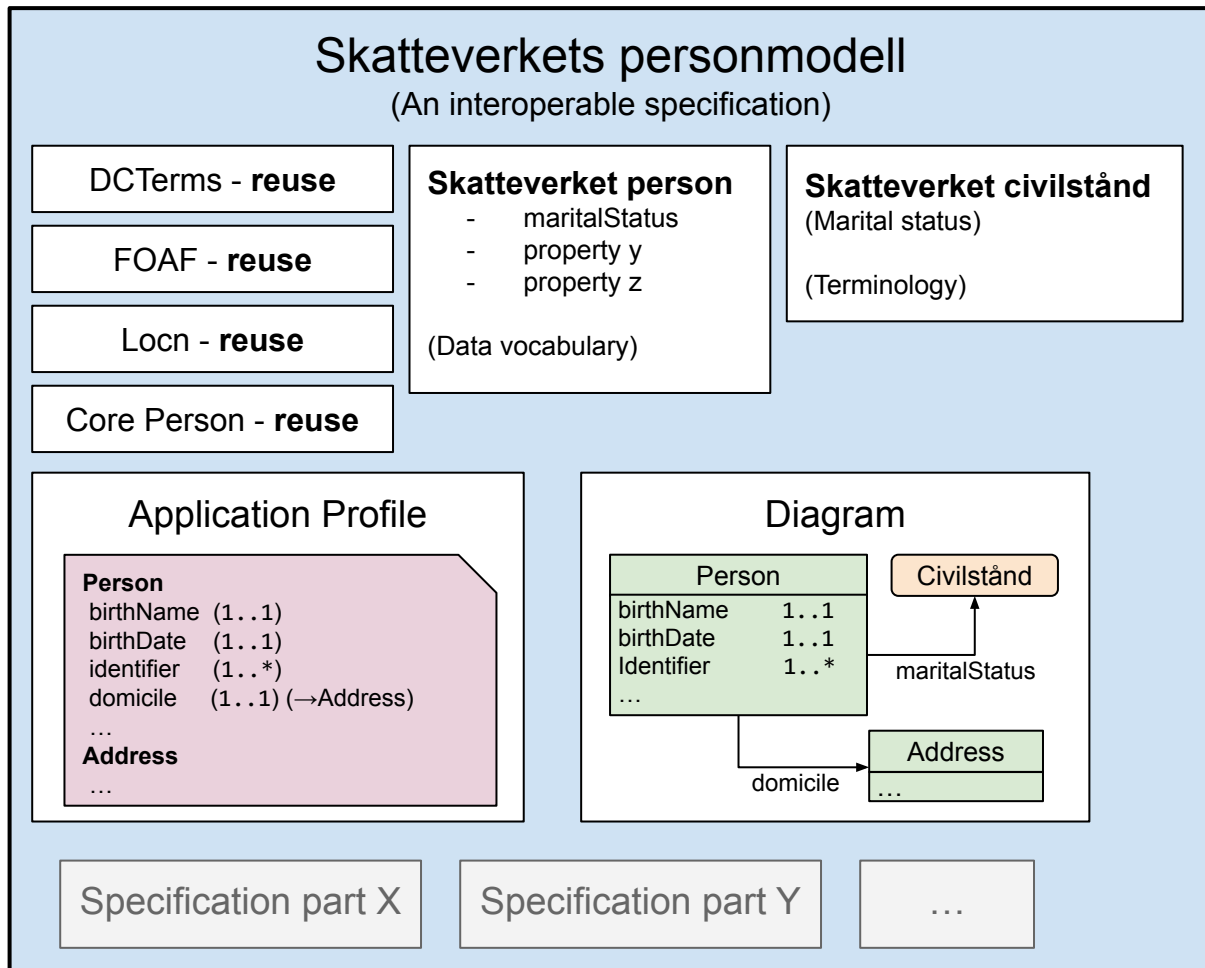
En specifikation är en behållare med resurser



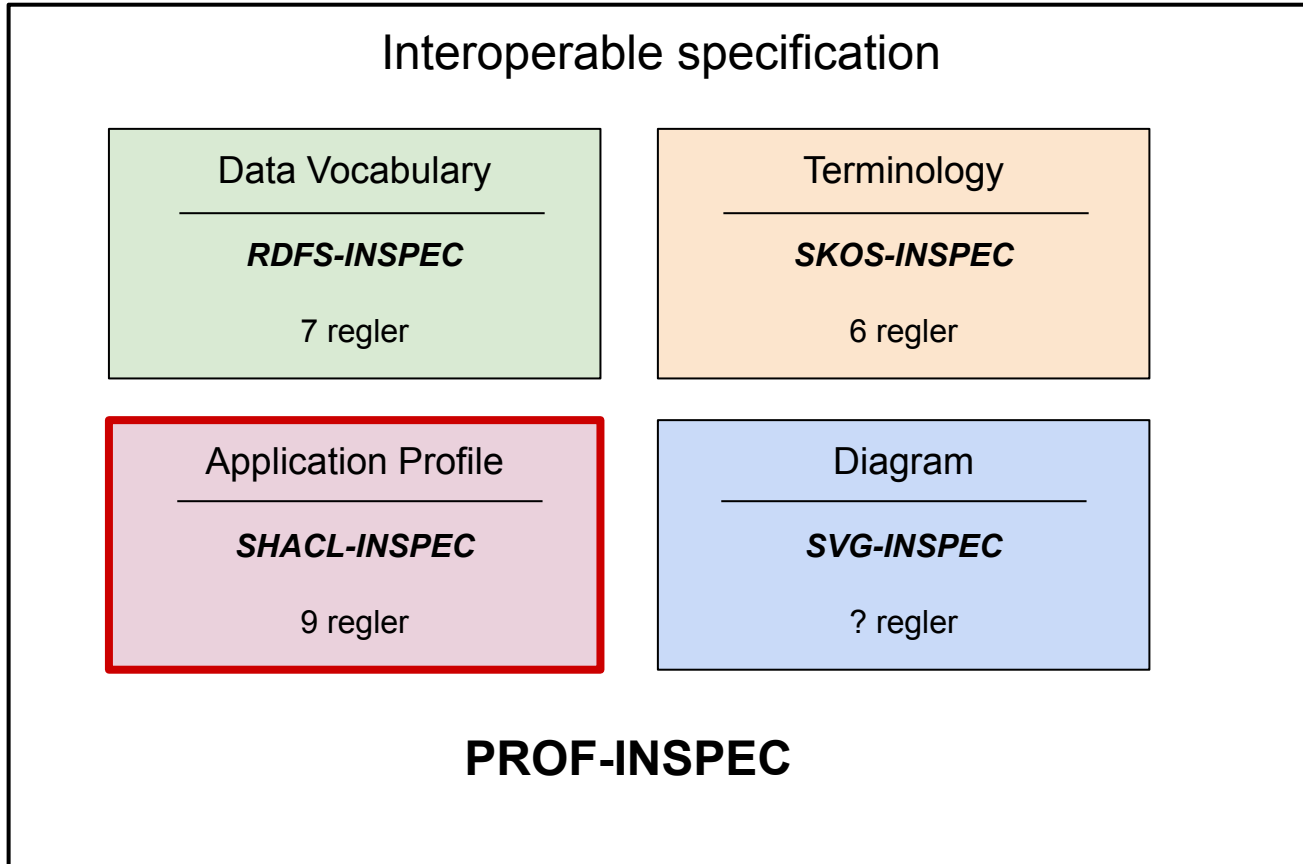
Skatteverkets personmodell

Observera att detta bara är ett exempel, inte en fullständig vy av hur Skatteverkets personmodell skulle kunna se ut.

Utgångspunkten är att den isåfall görs baserad på EUs Core Person vocabulary.

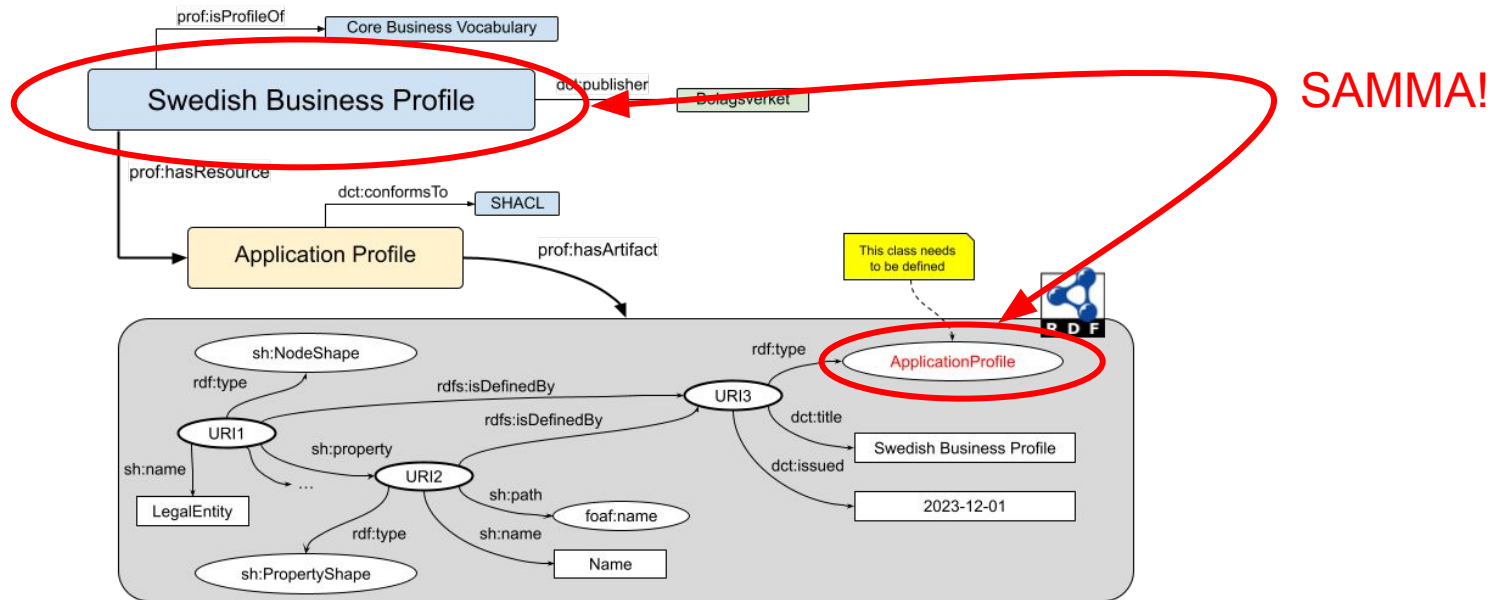


Fem profiler - extension "INSPEC"



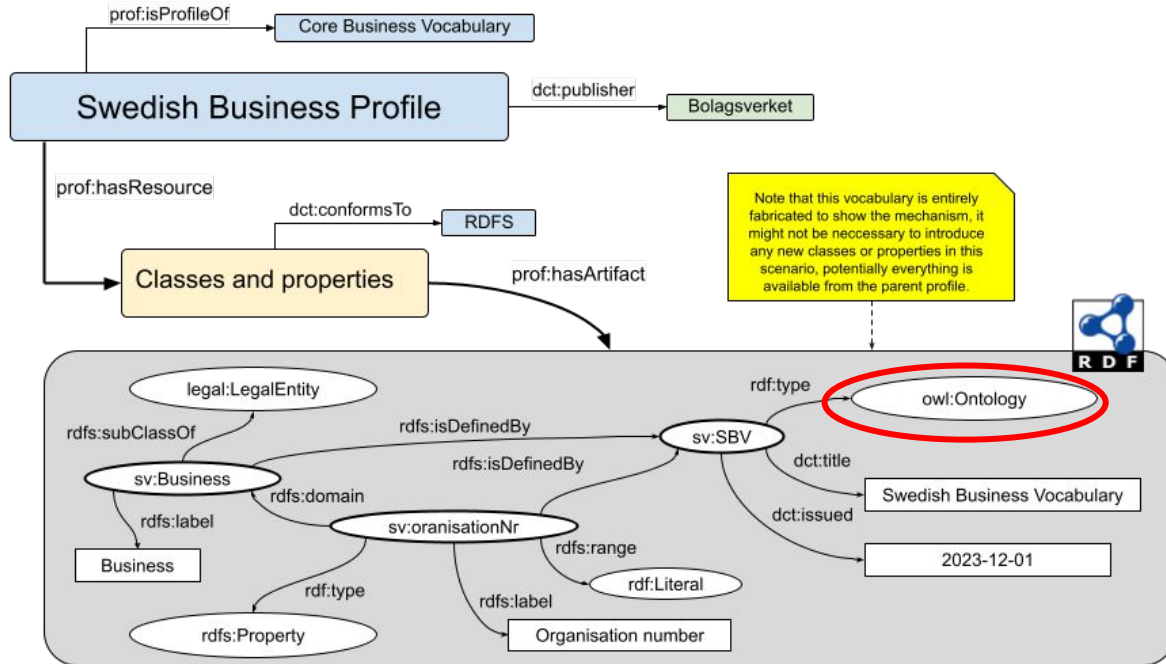
Centrala insikter sedan sist 1

- Vi kan låta Interoperabla specifikationen (prof:Profile) vara samma som applikationsprofilens centrala resurs, dvs samma URI bara rikare uttryck.



Centrala insikter sedan sist 2

- Data vokabulär kan använda owl:Ontology klassen

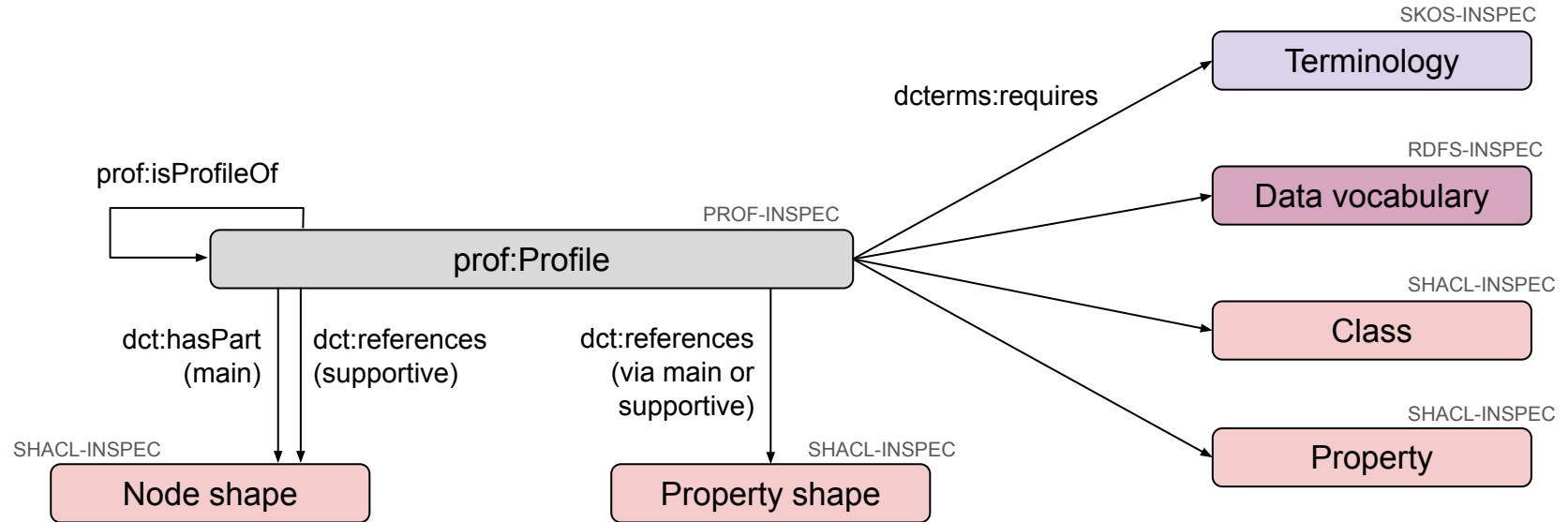


Centrala insikter sedan sist 3

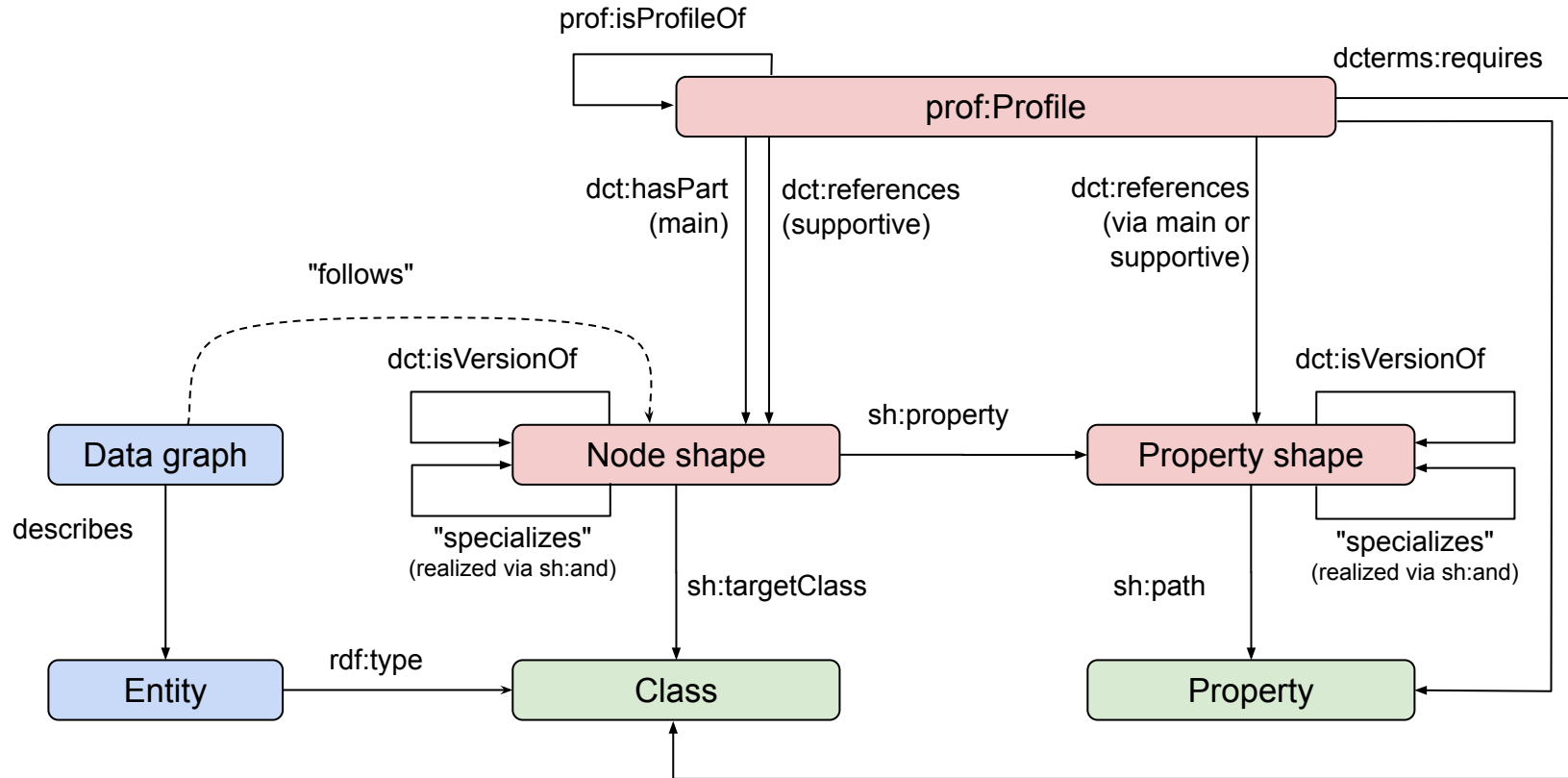
Vi klarar oss utan att introducera egna properties!

- Relationer från Dublin core
 - `dcterms:references` (för peka ut property shapes)
 - `dcterms:hasPart` (för markera main och supportive shapes)
 - `dcterms:requires` (för peka på klasser, properties, ontologier och terminologier)
 - `dcterms:isVersionOf` (för "based on")
- Prof relationer
 - `prof:isProfileOf`
 - `prof:isInheritedFrom`

Interoperable specifications berikas av sina delar



Application Profile uttryck i SHACL



Main and supportive Node Shapes

Note that there are node shapes of more technical nature that are excluded from the requirements below, all those must have a `sh:severity` set to `sh:INFO` or `sh:WARNING`. For example this covers node shapes pointed to via `sh:node` used to express more complex constraints such as indicating which terminology to choose concepts from.

The following information **MUST** be provided for a node shape:

- A stable identity in the form of a URI (subject position in triples)
- A label expressed via the property `sh:name`
- A list of property shapes via the property `sh:property` (see [section on defining order](#))

The following information **MAY** be provided:

- A class it corresponds to via the target declaration `sh:targetClass`
- A description / definition expressed via the property `sh:description`
- A usage note expressed via the property `vann:usageNote`
- A reference to another node shape it
 - "refines" via `sh:and` with a SHACL list containing the refined node shape (see [section on refinement](#)), OR
 - "is based on" via the `dcterms:isVersionOf` property (see [section on based on](#))

Property Shapes pointed to from main and supportive node shapes

Note that there are property shapes of more technical nature that are excluded from the requirements below. Property shapes are either excluded as they are referred to only from excluded node or from logical constraint components.

The following information MUST be provided for a property shape:

- A stable identity in the form of a URI (subject position in triples)
- A label expressed via the property `sh:name`
- The property it describes how to use via `sh:path`
- The value type to match against, `sh:nodekind` pointing to `sh:IRI`, `sh:BlankNode`, `sh:Literal` etc.

Property Shapes forts.

The following information MAY be provided:

- Express cardinality by:
 - `sh:minCount` `"1"^^xsd:integer` for mandatory
 - `sh:minCount` `"-1"^^xsd:integer` for preferred
 - `sh:minCount` `"0"^^xsd:integer` for preferred (can be left out)
 - `sh:maxCount` `"N"^^xsd:integer` for a maximum cardinality of `N`
- A description / definition expressed via the property `sh:description`
- A usage note expressed via the property `vann:usageNote`
- That a datatype is required on literals by using `sh:datatype` (if several datatypes are allowed, a construction with several property shapes with individual `sh:datatype` joined together via `sh:or` is necessary)
- That a language is required on literals by setting `sh:datatype` to `rdf:langString`
- Constraints on which literals that is allowed by:
 - An explicit list by using `sh:in` pointing to a SHACL list
 - A constraining pattern expressed by `sh:pattern` (Regular expression)
- Constraints on which URIs that is allowed by:
 - An explicit list by using `sh:in` pointing to a SHACL list
 - A constraining URI pattern expressed by `sh:pattern` (Regular expression)
 - Constrain to concepts in a terminology (see [section below](#))
 - Constrain to concepts in a concept collection (see section below (TODO))
 - Constrain to instances of a class by `sh:class` (if instances from several classes are allowed, a construction with several property shapes with `sh:class` joined together via a `sh:or` is necessary)
- A reference to another property shape it:
 - "refines" via `sh:and` with a SHACL list containing the refined property shape (see [section on refinement](#)), OR
 - "is based on" via the `dcterms:isVersionOf` property (see [section on based on](#))

Arv mellan property shapes

SHACL allows shapes to be combined via `sh:and` . This can be used to specialize an existing shapes with additional constraints or further restricting. E.g. consider the following property shape for the property `dcterms:publisher` where the range is `foaf:Agent` .

```
ex:ps1 a sh:PropertyShape ;  
  sh:label "Publisher" ;  
  sh:path dcterms:publisher ;  
  sh:nodeKind sh:URI ;  
  sh:minCount "1" ;  
  sh:class foaf:Agent .
```



we can further constrain it to the subclass `foaf:Organization` via the following construction:

```
ex:ps2 a sh:PropertyShape  
  sh:path dcterms:publisher ;  
  sh:class foaf:Organization ;  
  sh:and ( ex:ps1 ) .
```



Note that at a minimum we have to duplicate the `sh:path` property.