Strukturella Metadata - Referensgrupp 25 nov

Närvarande

Andrew Mercer (Riksantikvarieämbetet)
Cilla Öhnfeldt (Naturvårdsverket)
Erik Mossing (Bolagsverket)
Fredrik Emanuelsson (Riksarkivet)
Fredrik Klingwall (Kungliga biblioteket)
Fredrik Persäter (Lantmäteriet)
Kristofer Gäfvert (Lantmäteriet)
Marcus Smith (Riksantikvarieämbetet)
Matthias Palmér (Digg)
Mattias Ekhem (Digg)
Michalis Vassilas (Digg)
Olof Olsson (Svensk nationell datatjänst)
Patrik Wahlgren (SCB)
Stefan Jakobsson (Svensk nationell datatjänst)
Tomas Lindberg (SGU)
Ulrika Domellöf Mattson (Digg)

Bakgrund och referensgruppsarbete

Arbetet sker inom byggblock metadata, en del av Ena, Sveriges digitala infrastruktur och drivs av Myndigheten för digital förvaltning. Syftet med referensgruppen är att arbeta fram en profil för att främja interoperabilitet vid datadelning genom återanvändning av klasser, egenskaper och koncept. Klasser och egenskaper definieras så att de blir enklare att återanvända och att kombinera på olika sätt. Att beskriva hur de kombineras kallar vi en applikationsprofil.

Allt material kommer att publiceras till Github: https://github.com/diggsweden/interoperable-specifications

Materialet som tagits fram är på engelska är för att underlätta dialog med bland annat SEMIC (Europeiska kommissionens initiativ för att förbättra den semantiska interoperabiliteten) på EU-nivå. Referensgruppsdeltagarnas namn kommer att synas på Digg:s GitHub. Om ni inte vill att era namn ska synas på GitHub hör av er till Ulrika Domellöf Mattsson.

Tidigare förstudie finns även den på Github: https://github.com/diggsweden/information-models-investigation/tree/main

Repetition om interoperabla specifikationer (se även ppt)

Syftet med interoperabla specifikationer är att stödja återanvändning mellan specifikationer. Förhoppningen är bland annat att förenkla, effektivisera och förbättra kvaliteten i anslutning till att nya informationsmodeller skapas då myndighetsaktörer inte behöver uppfinna hjulet på nytt utan kan återanvända befintliga klasser.

En specifikation är en behållare med resurser. En samling av en mängd olika artefakter såsom informationsmodeller, guidelines diagram och exempeldata. Förr var specifikationer ofta i PDF-format (goda exempel, modell etc.).

En interoperabel specifikation i sin tur möjliggör återanvändning av klasser, egenskaper (attribut och relationer) samt begrepp. Klasser är i sammanhanget en kategorisering av data eller identitet med liknande egenskaper. Egenskaper som i UML betecknas som attribut (i RDFS talar man om properties). Man paketerar ofta klasser och properties i så kallade datavokabulärer. Termen kommer från RDFS (data vocabulary).

Mathias visade ett högnivåexempel (Se ppt), som inte ska ses som en fullständig vy, av hur Skatteverkets personmodell skulle kunna se ut som en interabel specifikation. Utgångspunkten i exemplet är att den baseras på EU:s Core Person Vocabulary. Specifikationen använder sig av redan etablerade vokabulär (DCTerms, FOAF, Locn och Core Person) istället för att Skatteverket börjar om från början. Skatteverket definierar enbart tillkommande delar såsom egenskap (Property) för martialStatus med tillhörande begrepp. Applikationsprofilen innehåller sedan martialStatus och Civilstånd och ingående relationer.

Återkoppling från referensgrupp, frågor och tankar

Digg fick återkoppling innan mötet om att tidsplanen att redan den 9 december ta ställning och fatta beslut kring en interoperabel specifikation var för snäv och att det blir svårt att ta ställning och fatta ett beslut med så pass kort varsel.

Nytt föreslag är istället att processen förskjuts så att den förslagna specifikationen skickas ut på remiss efter den 9 december och att referensgruppen återsamlas i vår för att gå igenom eventuella förbättringsförslag och synpunkter innan publicering. Den nya ordningen godkändes av referensgruppen.

Fördjupning om subklasser och varför OWL är problematiskt i sammanhanget (Se även ppt och Github)

Fortsättning på diskussionen från föregående referensgruppsmöte kring varför subklasser och OWL är problematiska när det kommer till interoperabla specifikationer.

Vi komplicerar återanvändningen av klasser om vi introducerar subklasser. Om vi skapar en subklass av en redan välkänd klass för att beskriva ytterligare properties (egenskaper) hur får vi då andra system/användare att förestå? Vi har i sammanhanget tre alternativ:

1. Länkade data – Slå upp med URI och förlitar oss att användaren upptäcker vilken klass den tillhör (ärver från). Användaren måste slå upp underklassen.
2. 2. Skicka med schema - Mer data att skicka och kräver extra tolkning
3. 3. Dubbeltypning – Beskriver både subklass och den överordnade klassen vid instanseringen. Enklast men kräver att avsändaren ger extra information.

Oavsett alternativ kommer de med en ökad kostnad för den konsumerande parten. När man introducerar subklasser ökar komplexiteten och ansvaret trycks så att säga neråt. Det blir med andra ord mycket jobb för att visa någon extra egenskap i form av en subklass.

OWL har två sätt att använda klasser och properties, antigen att skapa nya klasser och använda property restrictions eller att återdefiniera samma klasser och properties, vilket enbart fungerar i en stängd värld. Att definiera underklasser eller omdefiniera befintliga klasser för att hantera tekniska begränsningar kan leda till komplikationer och konflikter. Scenario – Bolagsverket skapar en egen klass av person för att lägga till en property (egenskap) vilket gör att Skatteverkets instans inte länge blir gällande då den i praktiken omdefinierat av Bolagsverkets instans.

När det kommer till interoperabla specifikationer har vi större fokus på properties, då de definieras oberoende av klasser. Properties återanvänds direkt inne i applikationsprofilen och sub- och egenskapsklasser undviks. Mathias drog parallellen till DCAT-AP och datamängder och datamängdsserier. Skillnaden mellan datamängder och datamängdsserier är liten och visas bäst med properites istället för subklasser.

SEMIC-stilguiden rekommenderar att endast introducera nya klasser och egenskaper vid semantiska anpassningar och att använda UML eller OWL2 för detta. För interoperabla specifikationer är det kanske att föredra att använda applikationsprofiler uttryckta i SHACL än att förlita sig på OWL2. Förslag till princip – Att använda sig av applikationsprofiler istället för subklasser och enbart subklasser om man ser ett informationsvärde.

För att förtydliga utmaningar med att använda subklasser och OWL har Mathias tagit fram följande text som vi vill ha synpunkter och inspel kring: https://github.com/diggsweden/interoperable-specifications/blob/main/docs/background.md#the-cost-of-subclassing

Detta är en viktig del i arbetet och de i referensgruppen med avvikande eller kompletterande åsikt uppmanas att läsa igenom stycket och höra av sig direkt till Mathias eller indirekt på Github. Här vill vi ha hjälp att förtydliga / förbättra innehållet.

Använd prof:isProfileOf för att indikera återanvändning av datavokabulär (

IsProfileOf relationer istället för reuse - Om vi istället använder prof:isProfileOf, måste vi länka mellan specifikationer oftare. Fördelen med återanvändningsmarkering (reuse) är att den informerar en kördare om att specifikationen inte ansvarar för att samla in en viss dataterminologi, vilket främst undviker dupliceringar. Det garanterar dock inte att någon annan har tillhandahållit terminologin.

Om vi istället använder prof:isProfileOf, innebär det att specifikationen redan har sett till att den aktuella dataterminologin har samlats in. Man kan inte peka på något som inte finns.. Om det inte finns får den som ska använda den lägga till relationen prof:isProfileOf. En nackdel är att vi kan behöva ladda upp många specifikationer för att få en komplett bild av en enskild specifikation.

Det finns ett ärende upplagt i Github – Ärendenummer 6

Övriga öppna ärenden på Github

Mathias gick snabb igenom övriga ärenden på Github

Svensk översättning av property (#3)

Egenskap är sannolikt den bästa översättningen av property.

Förtydliga skillnaden mellan ett begrepp och en klass (#5)

Det är inte ovanligt att konceptuella modeller (begreppsmodeller) används som utgångspunkt för att göra informationsmodeller. Kan skillnaden mellan ett begrepp och en klass göras tydligt? Hur ska vi hantera det faktum att SKOS inte stödjer begreppsmodeller med subklasser/delmängder?
Borde inte vara ett problem så länge vi definierar vad vi menar med begrepp och begreppsmodellering i det här sammanhanget.

En eller flera diagram per specifikation (#7)

Antingen behöver vi tillåta flera diagram, eller interaktiva. Tillämpningarna kan ju bli stora och komplexa. Även om det är en rimlig, kanske börja med att stödja ett diagram i ett inledningsskede?

Gör diagram interaktiva (#10)

Mathias föreslog att en visuell översikt i form av ett diagram är en bra start. Att göra diagrammet interaktivt, så att man kan klicka på klasser och enhetsprofiler för att få mer detaljerade beskrivningar, skulle vara mycket användbart. Det skulle också göra det möjligt att ha mindre detaljer i diagrammet, vilket skulle göra det mindre rörigt, eftersom man kan få mer information genom att kombinera det med de formella textuella definitionerna.

Förslaget inkluderar att skapa en regel för hur man gör en SVG-bild hyperlänkad och hur man kan bädda in information som datattribut i element. Detta skulle göra det möjligt för en webbapplikation att fånga upp interaktioner och lyfta fram motsvarande textuella definitioner. Det skulle vara fördelaktigt att kunna klargöra om det som klickades på var en klass, en egenskap, en enhetsprofil eller en egenskapsprofil.

Kan diagrammet skapas automatiskt utifrån våra maskinläsbara specifikationer? I så fall är det "bara" en tillämpningsfråga, och diagrammen kan återskapas om vi ändrar på hur vi vill presentera dem? Sannolikt svårt att åstadkomma i praktiken. Vi tänker oss något i stil med att det görs en export från EA och vi generera en SVG-bild.

I DDI-aliansen så autogenereras diagram men där väljer man att begränsa till ett diagram för klass som visar relationerna till relaterade klasser så grafen blir mindre. tex: https://ddialliance.github.io/ddimodel-web/DDI-L-3.3/item-types/VariableScheme/ idag är det png-format men man har även testat att jobba med svg-generering med Mermaid.

Lägga till en glossary (#8)

Kan behövas, även om många av begreppen är vedertagna.

Lägga till ett genomtänkt exempel (#9)

Viktigt för att förankra konceptet och öka förståelsen för det vi försöker göra med en profil för interoperabla specifikationer. Hittills har vi fokuserat på Skatteverket och informationsmodellen kopplad till grunddatadomänen Person. Ett annat förslag är att utgå från informationsmodell för den grunddatadomän som Bolagsverket ansvarar för - Företag.

Avslutande ord

Gå in i Github och kommentera om ni har synpunkter eller förbättringsförslag. Det går bra att skapa en ”fork” och skriva förslag där också om man vill. Vi lägger i vanlig ordning upp både presentation och mötesanteckningar på Github.